UK Atomic Energy Authority

ExCALIBUR

ExCALIBUR

Report on user layer design for UQ - M3.1.3

Abstract

The report describes work for ExCALIBUR project NEPTUNE around techniques for Uncertainty Quantification (UQ) that are expected to prove important for the project, arranged as they might appear in a workflow to optimise plant design. Firstly, efficient ways of identifying the major sources of uncertainty are identified, then secondly, techniques for working with this smaller number are described, including the production of surrogates. Third and lastly, UQ analysis is then continued with the surrogates used to predict distributions of expected outcomes, with emphasis upon producing optimal device designs, which are robust against say installation errors. Since NEPTUNE software may be required for a range of applications including comparison with theory, there is a discussion around how UQ might best be integrated in the context of NEPTUNE, prior to completion of research work external to UKAEA.

UKAEA REFERENCE AND APPROVAL SHEET

	Client Reference:	
	UKAEA Reference:	CD/EXCALIBUR-FMS/0024
	Issue:	1.00
	Date:	14 October 2020

Project Name: ExCALIBUR Fusion Modelling System.

Version 1.00

	Name and Department	Signature	Date
Prepared By:	Wayne Arter Ed Threlfall Joseph Parker BD	N/A N/A N/A	14/10/2020
Reviewed By:	Rob Akers	N/A	14/10/2020
Modified By:	Rob Akers Advanced Computing Dept. Manager BD	[illegible]	14/10/2020
Approved By:	Rob Akers BD	[illegible]	14/10/2020

1 Introduction

Uncertainty quantification (UQ) is a topic of interest to nuclear engineers (where it may feed into probabilistic risk assessment studies of safety), chemical engineers, systems engineers, control engineers, meteorologists, and probably many others, see the book by Smith [1]. This means that there is widespread literature available, but unfortunately it also means that the nomenclature is not standardised. This report will follow Smith's text as far as possible as to definitions. Smith defines UQ as: "The synergy between statistics, applied mathematics and domain sciences required to quantify uncertainties in inputs and Quantities of Interest (QoI) when models are too computationally complex to permit sole reliance on sampling-based methods." Implicit in Smith's definition is thus the production of a surrogate, which immediately highlights differences in nomenclature in that both the COSSAN [2] and Dakota [3] packages describe themselves as performing UQ analysis without necessarily using a surrogate. For NEPTUNE, surrogates will preferably have an underlying physics basis, and unless explicitly stated, numerical errors due to the discretisation and to round-off in finite precision computer arithmetic will be neglected.

The material on UQ in the present report will be ordered by application to the workflow in a generic, large scientific-modelling package as presented by Habib Najm at the Workshop on "Data Assimilation and Uncertainty Quantification at the Exascale", 24-25 September 2020 drawing extensively from ref [4]. The first stage of the workflow is to understand the behaviour of the full model, assumed to be dependent on a large number of parameters d, giving rise to the so-called 'curse of dimensionality', see Section 2.1. It is notable that Najm prefers to proceed to use only a small number of techniques of global sensitivity analysis, concentrating on sparse sampling and does *not* first examine sensitivity to small changes in input (ie. local sensitivity analysis). The second stage is to produce a surrogate dependent only on of order ten or so parameters, and the third is to use such surrogates to compute uncertainty in QoI, which may include solutions with optimal properties. It will of course be seen that sampling techniques remain important because thay are needed both for the first and third stages.

Before proceeding, it is worth remarking that two different kinds of uncertainty are distinguished, namely aleatory and epistemic uncertainties, less confusingly referred to as 'irreducible' and 'reducible' uncertainty. The first have been referred to as the "unknown unknowns", and the second as the "known unknowns", in that the latter could be determined simply by further measurement. Patelli et al [2] argue for more of a continuum of ignorance, because for example certain measurements may be available some but not all of the time.

This report will begin by describing the mathematical basis of methods used in UQ, ordered by the above stages in Section 2. It will proceed in Section 3 to outline how the mathematics might be implemented in the NEPTUNE software, and a brief summary is provided in Section 4.

2 Mathematics Background

2.1 Understanding Model Behaviour

This section describes techniques useful in global sensitivity analysis, specifically

- Polynomial Chaos Expansion (PCE)
- Sparse regression with application to PCE, including LASSO/Basis Pursuit (BP) algorithms
- Multifidelity Monte-Carlo (MFMC)
- Multilevel Monte-Carlo (MLMC) and Multi-Index Monte-Carlo (MIMC)

The 'curse of dimensionality' The need for special techniques arises from the need to treat problems typically with a large number d of parameters that give rise to the so-called 'curse of dimensionality'. The curse defeats simple uniform sampling strategies where supposing that 10 samples are taken for each parameter, the number of samples required is 10^d. This fact motivates use of Monte-Carlo (MC) sampling which has costs independent of d but gives rise to a slowly reducing error$\propto 1/\sqrt{N}$ in the calculation of averages, where N is the number of samples.

To calculate more accurate statistics, use is made of Quasi-Monte-Carlo (QMC) sampling, see the appended Section 5.1, and sparse (quadrature) sampling Section 2.2.1 which can achieve errors$\propto (\ln N)^d/N$. Even in this expression, the power-law dependence on d restricts use of the technique to values of d of order ten or so, more precise values depending on the cost of a single sample calculation, hence the need for techniques of sparse regression to identify a restricted set of parameters.

2.1.1 Polynomial Chaos Expansion (PCE)

Polynomial Chaos Expansion (PCE), also known as the Wiener chaos expansion [5], is a method for determining the evolution of uncertainty in a dynamical system where there is imperfect knowledge of the system parameters. Uncertain parameters and variables are supposed to depend on a normally distributed random variable θ, and are then expressed using a Hermite expansion. For example, density $n(\mathbf{x}, t)$, usually a function of position $\mathbf{x}$ and time t, would gain an additional θ-dependence and be written

$$n(\mathbf{x}, t, \theta) = \sum_{j=0}^{N_H} n_j(\mathbf{x}, t) H_j(\theta), \tag{1}$$

where $n_j(\mathbf{x}, t)$ are deterministic coefficients and H_j are the Hermite polynomials. Hermite polynomials are used as these are orthogonal with respect to a Gaussian (*i.e.* normal distribution) weight,

$$\langle H_n | H_m \rangle = \frac{1}{2^n n!} \int_{-\infty}^{\infty} H_n(\theta) H_m(\theta) \frac{e^{-\theta^2}}{\sqrt{\pi}} \, \mathrm{d}\theta = \delta_{nm}, \tag{2}$$

where δ_{nm} is the Kronecker which is 1 if $n = m$ and 0 otherwise. Statistical properties, which are moments of n with respect to θ, are then known simply in terms of the expansion coefficients. For example, the expected value of n is

$$\mathbb{E}(n) \equiv \int_{-\infty}^{\infty} n(\mathbf{x}, t, \theta) \, \mathrm{d}\theta = n_0(\mathbf{x}, t). \tag{3}$$

Moreover, by taking the inner product of the model equations with each Hermite polynomial in turn, one obtains a set of moment equations for the evolution of the coefficients n_j. For example, taking moments of the equation for the conservation of density,

$$\frac{\partial n}{\partial t} + \frac{\partial (nu)}{\partial x} = 0, \tag{4}$$

yields the $N_H + 1$ equations

$$\frac{\partial n_k}{\partial t} + \sum_{i=0}^{N_H} \sum_{j=0}^{N_H} e_{ijk} \frac{\partial (n_i u_j)}{\partial x} = 0, \tag{5}$$

with $e_{ijk} \equiv \int_{-\infty}^{\infty} H_i H_j H_k \, \mathrm{d}\theta$ being (easily-computable) integrals. Equation (5) also appears in the Equations document [6]. Solving (5) allows one to reconstruct $n(\mathbf{x}, t, \theta)$ via (1), which gives the density along with a quantification of its uncertainty through θ.

This method may be generalised to include a set of random variables $\{\theta_i\}$, in which case an expansion analogous to (1) is made using tensor Hermite polynomials.

The Hermite expansion approach is appropriate only for θ a normally distributed random variable (or certain similar types, like log-normal distributions). For other distributions, convergence of the expansion (1) with N_H can be extremely slow. However using non-normal distributions is often desirable; for example, it might be known that errors are non-negative, or uniformly distributed. To allow modelling with non-normal distributions, Xiu [7] introduced Generalised Polynomial Chaos (gPC). In gPC, the random variable θ is generalised to be a random variable with any distribution function f. The Hermite polynomial expansion is then replaced with an expansion in the set of orthogonal polynomials whose weight function is f. For example, if θ is uniformly distributed, the expansion is in Legendre polynomials. The gPC theory also supports discrete random variables, such as the Poisson and the binomial distributions; a list of random variables and their corresponding orthogonal polynomials are given in [7, Table 2.1].

The method described above is *intrusive*, that is, if one only has software to solve the original problem (4) for n, then non-trivial changes are likely required in order to solve (5) for n_k. An alternative non-intrusive approach may be derived as follows. Suppose the solution is sampled at a set of N random points $\{\theta_j\}_{j=1}^{N}$, and construe n as a distribution in probability space, viz.

$$n(x, t, \theta) = \sum_{j=0}^{N} n_{sj}(x, t) \delta(\theta - \theta_j). \tag{6}$$

Equating (6) with (1) in the weak sense, there obtains

$$\left\langle \sum_{j=0}^{N} n_{sj}(x, t) \delta(\theta - \theta_j) \middle| H_k(\theta) \right\rangle = \left\langle \sum_{j=0}^{N_H} n_j(x, t) H_j(\theta) \middle| H_k(\theta) \right\rangle, \tag{7}$$

where $\langle\cdot|\cdot\rangle$ is the inner product defined in (2), so that

$$n_k(x,t) = \frac{1}{2^k k! \sqrt{\pi}} \sum_{j=0}^{N_H} n_{sj}(x,t) H_k(\theta_j) e^{-\theta_j^2}. \tag{8}$$

This expression for n_k may be calculated using only knowledge of n from solving the original equation (4).

2.1.2 Sparse linear regression

Linear regression Suppose a data set $\{y_i, x_{1i}, \ldots, x_{Pi}\}_{i=1}^N$ to consist of N observations of the dependent variable y and the d independent variables $\mathbf{x}$, supposed related by the model $y = \mathbf{x}\boldsymbol{\beta}$, where $\boldsymbol{\beta}^T = (\beta_1, \ldots, \beta_d)$ is a vector of unknown coefficients of the model. The classical linear regression problem determines $\boldsymbol{\beta}$ by minimising the L_2-norm error,

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^d} \|y - \mathbf{x}\boldsymbol{\beta}\|^2. \tag{9}$$

When there are more observations than parameters, $N > d$, this has the well-defined solution $\boldsymbol{\beta}^* = (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T y$.

Sparse linear regression In some applications there are many more observations than parameters $N \gg d$, and moreover expect many of the observation points x_j to be irrelevant for determining y. In this case, expect many of the coefficients β_i would be expected to be zero, but will not known *a priori* which can be neglected. Many approaches have been developed to incorporate sparsity into linear regression. This section focusses on

- Sequential Least-Squares' Thresholding (SLSQT)
- Best Subset Selection
- Ridge Regression
- LASSO

The last three of these may all be formulated as regularised least-squares optimisation, *i.e.* least-squares subject to an additional inequality constraint. For simplicity, the following discussion is restricted to the case when the independent variables $\mathbf{x}_i$ are orthonormal, $\langle\mathbf{x}_i|\mathbf{x}_j\rangle = \delta_{ij}$, which applies when the independent variables are the basis functions from a polynomial chaos expansion.

Sequential Least-Squares' Thresholding (SLSQT) is simply described as a sequence of least-squares' fits in which after each fit regression coefficients that drop below some researcher-fixed threshold are set and remain zero thereafter. This has proved very successful for Brunton et al [8].

Best subset selection A natural formulation for seeking a sparse coefficient vector $\boldsymbol{\beta}$ is to restrict β to having a small number of non-zero entries. The linear regression problem (9) is modified to

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^d} \|y - \mathbf{x}\boldsymbol{\beta}\|_2^2 \quad \text{subject to} \quad \|\boldsymbol{\beta}\|_0 \leq C. \tag{10}$$

where $\|\boldsymbol{\beta}\|_0$ is the "0-norm" which simply counts the number of non-zero elements of $\boldsymbol{\beta}$. This is known as "Best Subset Selection" as the constraint on β restricts the nonzero coefficients to a k-dimensional subspace of the original problem.

Equation (10) may be written in Lagrangian form,

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^d} \left(\frac{1}{N}\|y - \mathbf{x}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_0\right), \tag{11}$$

which is solved by

$$\beta_j = \mathcal{H}_{\sqrt{N\lambda}}(\beta_j^*) = \beta_j^*\mathcal{I}\left(|\beta_j^*| > \sqrt{N\lambda}\right) \tag{12}$$

where $\mathcal{I}$ is the indicator function that is unity if its argument is true, and zero otherwise, and $\boldsymbol{\beta}^* = (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\boldsymbol{\beta}$ is the solution to the ordinary least-squares problem. Here $\mathcal{H}_\alpha$ is the "hard thresholding function", which sets coefficients to zero once they are sufficiently small, but leaves other coefficients untouched.

This approach is however computationally infeasible. This is because the "0-norm", the $d \to 0$ limit of the d-norm, $\|\boldsymbol{\beta}\|_d = \left(\sum_i \beta_i^d\right)^{1/d}$, is not actually a norm (as, for example, $\|2\boldsymbol{\beta}\|_0 \neq 2\|\boldsymbol{\beta}\|_0$). This means one cannot minimise $\|\cdot\|_0$ except by exhaustive search. Therefore other methods replaces the "0-norm" with d-norms that are computationally tractable.

Ridge regression In ridge regression, also known as L_2 regularisation, the "0-norm" might be replaced with the Euclidean 2-norm,

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^p} \left(\frac{1}{N}\|y - \mathbf{x}\boldsymbol{\beta}\|_2^2 + \lambda\|\boldsymbol{\beta}\|_2^2\right), \tag{13}$$

This approach, a variant of Tikhonov regularisation, mitigates the problems due to dependencies among the variables x_i. However, because $\nabla\|\mathbf{x}\|_2^2 = -\mathbf{x}/\|\mathbf{x}\|_2^2$ is invariant under a rescaling of $\mathbf{x}$, Equation (13) has the solution

$$\beta_j = (1 + N\lambda)^{-1}\beta_j^*. \tag{14}$$

Evidently, this solution does not increase the sparsity of the solution and so does not reduce the number of model parameters, leading to consideration of use of the 1-norm in Equation (13).

LASSO The LASSO method, also called basis pursuit denoising or L_1 regularisation, and indeed 'compressed sensing' (CS), turns out to yield the favourable features of Best Subset Selection and Ridge Regression. In LASSO, the “0-norm” is replaced with the 1-norm, $\|\boldsymbol{\beta}\|_1 = \sum_i |\beta_i|$, in equation (10),

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^d} \left(\frac{1}{N} \|y - \mathbf{x}\boldsymbol{\beta}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_1 \right). \tag{15}$$

Although not obvious, it may be shown using the Karush-Kuhn-Tucker conditions for constrained optimisation that the 1-norm is also minimised by values of $\boldsymbol{\beta}$ containing exact zeros; moreover, smaller values of C in (10) yields $\boldsymbol{\beta}$ with more zeros. The 1-norm, being a norm, is also computationally tractable.

The solution to (15) is

$$\beta_j = \mathcal{S}_{n\lambda}\left(\beta_j^*\right) = \beta_j^* \max\left(0, 1 - \frac{N\lambda}{|\beta_j^*|}\right), \tag{16}$$

where $\mathcal{S}_\alpha$ is the “soft thresholding function”. This solution both shifts coefficient values towards zero, and sets the smaller values to zero, increasing the sparsity.

2.1.3 Multifidelity Monte-Carlo

Multifidelity modelling In conventional modelling, normally a “high-fidelity” model is produced which captures as faithfully as possible the system being modelled. Such models can be computationally expensive, motivating the use of low-fidelity, or reduced, models which approximate the same system but that, for example, describe simplified physics, or use a lower resolution grid. Reduced models are much cheaper, but may be much less able to predict the behaviour of a physical system. Moreover it may be difficult to certify the model by quantifying uncertainties.

Multifidelity modelling is an approach between these two extremes. It uses both high-fidelity and (possibly multiple) low-fidelity models in an attempt to place guarantees on the properties of the solution.

Monte-Carlo sampling In Monte-Carlo sampling, there is an uncertain input (random variable Z) to a high-fidelity model, $f^{(1)}$. The outputs from the model $f^{(1)}(Z)$ are also uncertain, so the goal is to calculate statistics, such as the expected value of the output, $s = \mathbb{E}(f^{(1)}(Z))$. Estimators for this statistic are made by making N evaluations of the model, that is, taking N realisations z_i from Z and computing

$$\hat{s} = \bar{y}_N^{(1)} = \frac{1}{N}\sum_{i=1}^{N} f^{(1)}(z_i). \tag{17}$$

If the work of evaluating the model once is W_1, then the Monte-Carlo process costs NW_1 work.

Multifidelity Monte-Carlo (MFMC) In multifidelity Monte-Carlo, the single high-fidelity model $f^{(1)}$ supplemented with a collection of $K-1$ lower-fidelity “surrogate” models, $f^{(2)}, \ldots, f^{(K)}$, where fidelity decreases with increasing index. Model i is evaluated n_i times, where higher fidelity models are evaluated fewer times, $n_1 \leq n_2 \leq \ldots \leq n_K$. Define mean estimators $\bar{y}_n^{(j)} = \frac{1}{n}\sum_{i=1}^{n} f^{(j)}(z_i)$ (from calling the jth model n times), and from these construct the MFMC estimator for s,

$$\hat{s} = \bar{y}_{n_1}^{(1)} + \sum_{i=2}^{K} \alpha_i \left(\bar{y}_{n_i}^{(i)} - \bar{y}_{n_{i-1}}^{(i)} \right). \tag{18}$$

Compared to (17), this requires many fewer evaluations of the high-fidelity model ($n_1 \ll N$), but additional evaluations of the surrogate models. In (18), the brackets term expresses the change in mean estimator of $\bar{y}^{(i)}$ that results from calling model i more times relative to model $i-1$, the model that is one-level higher-fidelity. The estimator is unbiased, and moreover gives the freedom to choose the model evaluations n_1, n_2, ..., n_K and the coefficients α_2, ..., α_K.

The cost of evaluating $\hat{s}$ is $\sum_{i=1}^{K} W_i n_i$. Because surrogate models are cheaper to compute ($W_1 > W_i$ for all $i > 1$) and many fewer evaluations of the high-fidelity model ($n_1 \ll n$) are anticipated, this cost is significantly less than the cost NW_1 of the conventional Monte-Carlo calculation. Peherstorfer *et al.* [9] used this approach to obtain the same accuracy as a high-fidelity MC simulation, but with a reduction in runtime of four orders of magnitude.

Multi-Level Monte-Carlo (MLMC) Multi-level Monte-Carlo (MLMC) is a more efficient way of estimating averages and other statistics than by means of random or Monte-Carlo (MC) sampling. MLMC is usually associated with the name of Mike Giles [10]. Suppose that a model or surrogate problem may be solved with a range of spatial and/or temporal resolutions. The basic idea is that sample solutions computed with different levels of resolution may be combined to produce means and other statistics (giving estimates of uncertainty) as accurately as though all solutions had been obtained using the finest level.

In more detail, for the example of a steady solution obtained with different spatial levels of resolution [11], the MLMC algorithm is

1. Construct a hierarchy of meshes, spacings h_l, where $l = 0$ is the coarsest level and $l = L$ is the finest.

2. For each l, draw a level-dependent number N_l of samples from the parameters/fields to be sampled, eg. $\mathbf{U}$ denoting a single solution depending on d spatial variables alone.

3. Solve the model or surrogate problem N_l times at each level, giving an ensemble of solutions $\mathbf{U}_l^k,\ k = 1, \ldots N_l$,

4. Estimate the mean (other moments of the distribution follow similarly) as

$$\mathbb{E}^L[\mathbf{U}] = \sum_{l=0}^{L} \mathbb{E}_l[\mathbf{U}_l - \mathbf{U}_{l-1}] \tag{19}$$

   using simple averages $\mathbb{E}_l$ with $\mathbf{U}_{-1} = \mathbf{0}$

Large gains occur when $N_l = N_0 4^{lp}$ where the solution scheme is of order $p \leq (d+1)/2$. A refinement of MLMC, referred to as Multi-Index Monte-Carlo (MIMC) [12] allows for h_l to be different by factors of 2 in different coordinate directions. As illustrated in Figure 1, the differences of function values in Equation (19) on two isotropic meshes may be replaced by differences between values on several different anisotropic meshes.

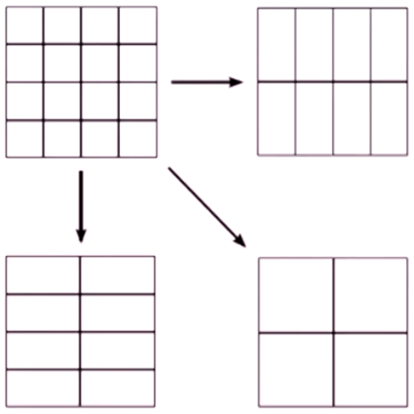

Figure 1: Different meshes for use in MLMC and MIMC in 2-D. In MLMC, only the bottom right and top left would be differenced in the Equation (19), whereas all four might be differenced in MIMC, depending on choice of ‘index set’.

Note that Najm *et al*’s work [13] often focusses on quantities of interest (QoIs) Q rather than the whole solution. They also point out that the error in the standard deviation is not available in closed form and its calculation involves a complicated approximation which they describe [13, $\S$ III.B.1].

2.2 Producing Surrogates

By this stage, the parameters that contribute most to the uncertainty have been reduced to a more manageable number, so that a global examination can be made of this remainder. Relevant techniques employed during this stage are

- Sparse quadrature sampling, especially adaptive
- Forward UQ
- Sobol Analysis

2.2.1 Sparse Quadrature Sampling

The idea of sampling on sparse grids seems to have supplanted QMC sampling (see Appendix Section 5.1) for many applications, presumably because it can be made to adapt to the functional dependence of outputs suggested by the sampling, unlike QMC or indeed Latin hypercube sampling

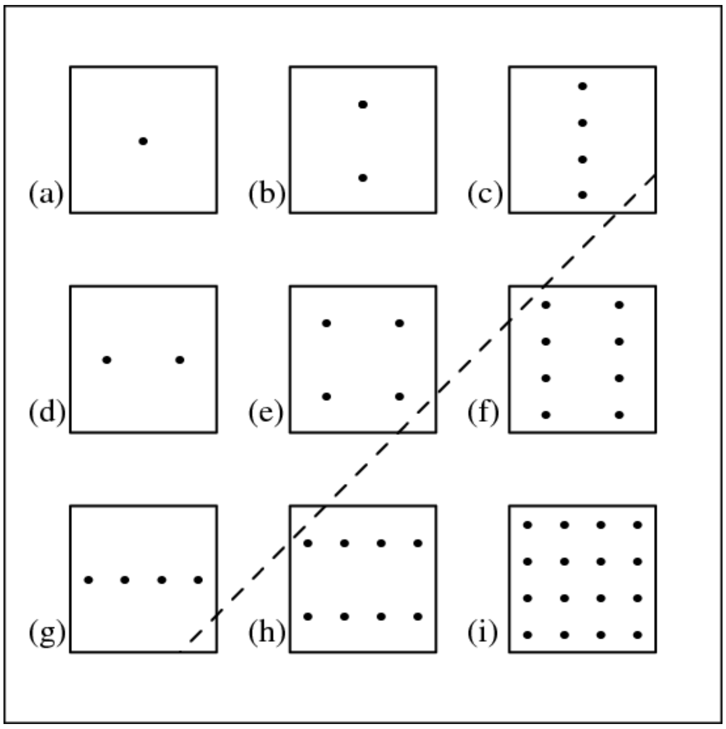

Figure 2: Sparse grid sampling. The point sets lying below the diagonal drawn dashed are omitted.

(see Appendix Section 5.2). The review article [14] is remarkable for the wide range of references on the subject of sparse sampling, not just in application to quadrature (evaluating integrals of functions). The key idea is best illustrated graphically, see Figure 2, and is that provided the integrand is well behaved, it is sufficient to have sample or quadrature points concentrated along the coordinate axes in the centre of the domain. (There is a variant which includes samples at the edge of the domain of integration.) The resulting set of sample points is shown in Figure 3(a). The locations of the points in the higher order sets are always determined by bisecting the positions of those in the lower order sets.

Provided that integrands are sufficiently smooth, integrals can be proved [14] to converge at a rate

$$\mathcal{O}(\log(N)^{d-1}/N) \tag{20}$$

where d is the number of dimensions over which the integral is performed. Indeed, by careful choice of point sets, excluding some which lie above the equivalent of the diagonal in Figure 2 for greater sampling density than shown there, it is possible to achieve convergence of the integral at a rate

$$\mathcal{O}(N^{-1}) \tag{21}$$

There is the important caveat that this rate is achieved only in the 'energy' norm:

$$\parallel Q \parallel_E = \sqrt{\left(\int \sum_{i=1}^{d} \left(\frac{\partial Q}{\partial x_i}\right)^2 d\mathbf{x}\right)} \tag{22}$$

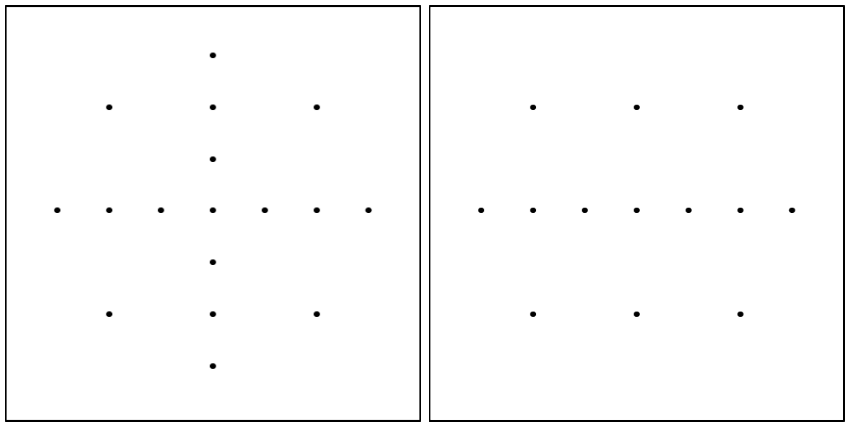

Figure 3: On the left is an example of isotropic sparse grid sampling. The diagram on the right indicates how the vertical direction may be more coarsely sampled than the horizontal, by omitting further the point set labelled (c) in Figure 2.

and in other norms the d-dependent rate Equation (20) is found. Nonetheless this implies that the number of samples needed to characterise the parameter space is

$$\mathcal{O}(N \log(N)^{d-1}) \tag{23}$$

If singularities are present, then sparse grids can be locally adapted [14, end § 4], most easily using the fact their construction by bisection naturally leads to error estimates based on the difference between the solutions on the mesh before and after subdivision. As explained in ref [15] sparse grids can also be dimension adaptive, placing more points in some coordinate directions than others, as illustrated in the right diagram of Figure 3.

2.2.2 Forward UQ

“Forward UQ” is conceptually the easiest kind of uncertainty quantification, in that it is assumed that the uncertainties are known at the outset. Thus in the case of a physics-based problem, all the important physical processes are regarded as having been identified, so that they can be parameterised. These parameters will in turn have known uncertainties, specified via probability distributions denoted $p(x)$.

The commonest distribution is the Gaussian, as a consequence of the Central Limit Theorem, which states that the probability distribution for the sum of an increasing number of independently and identically distributed random variables with appropriately normalised finite mean and variance tends to the “Normal” (ie. Gaussian) distribution. In practice the theorem is found to apply to most measurements, implying it is accurate for numbers less than ten, as the measurement chain from experiment to observer typically involves the compounding of a relatively small number of errors. If parameters are expressed in terms of bounds on their extreme values, this of course corresponds to uniform distributions. “Interval analysis” [16] includes this case, but also the Dempster-Shafer structure, whereby uncertainty is described by overlapping parameter intervals with different uniform probabilities.

It is straightforward, given a random number generator (RNG) that is equally likely to return any number ξ_i within the unit interval $[0, 1]$ to scale the outputs to be uniformly distributed in an arbitrary finite interval $[a, b]$, and relatively straightforward to arrange so that numbers are output with a given frequency distribution. Both commercial (such as matlab^{TM}) and opensource packages (eg. Python NumPy Generator [17]) are available. They rely on the fact that the cumulant (ie. definite integral) of a probability distribution is uniformly distributed, so that if the Cumulant $P(x)$ is defined as

$$P(x) = \int_{-\infty}^{x} p(x')dx' \tag{24}$$

then samples $x_i = P^{-1}(\xi_i), i = 1, 2, \ldots, N$ become distributed according to $p(x)$ as N becomes large.

Forward UQ most easily proceeds by constructing an ensemble of realisations of the detailed models at parameters sampled according the specified distributions. A distribution for each QoI may be estimated using the ensemble, from which follows an average and a standard deviation estimate for the QoI. Thus it is customary to talk of uncertainty as being propagated through the system.

2.2.3 Sobol

The use of adaptive sparse grids is a well-established procedure, see ref [18] to perform the Sobol ANOVA-like analysis. The approach described by Sobol [19] (based on a Russian paper from 1990) is based on a function-theoretic result which applies to a function which must at least be continuous. The 'Sobol' decomposition generalises ANOVA, where ANOVA stands for 'Analysis of Variance', a family of statistical methods for quantifying how the outputs of a system depend on its inputs, usually based on linearity assumptions.

Suppose that the function is $f(\mathbf{x}) = f(x_1, x_2, \ldots, x_d)$ where $\mathbf{x} \in I^d$ (i.e. $0 \le x_k \le 1, k = 1, \ldots, d$), then the Sobol decomposition is

$$f(x_1, \ldots, x_d) = f_0 + \sum_{k=1}^{d} f_k(x_k) + \sum_{j=1}^{d-1} \sum_{k=j+1}^{d} f_{jk}(x_j, x_k) + \cdots + f_{12\ldots d}(\mathbf{x}) \tag{25}$$

where f_0 is a constant and the integral of each term in the sums is zero, i.e.

$$\int_0^1 f_{j_1 j_2 \ldots j_r}(x_{j_1}, x_{j_2}, \ldots, x_{j_r})\, dx_{j_k} = 0, \quad 1 \le k \le r \tag{26}$$

There is the expectation that this latter property ensures that the term of higher order in r become negligible. The applicability to statistical analysis is evident when it is realised that Equation (25) may be interpreted as an expansion for the joint probability distribution $f(x_1, \ldots, x_d)$ since, integrating over the variables x_{j_r} as appropriate and using Equation (26) gives

$$\begin{aligned} f_0 &= \mathbb{E}(f) \\ f_i(x_i) &= \mathbb{E}(f|x_i) - f_0 \qquad (27) \\ f_{ij}(x_i, x_j) &= \mathbb{E}(f|x_i, x_j) - f_0 - f_i(x_i) - f_j(x_j) \qquad (28) \end{aligned}$$

so that the numbered equations give successively the effect of variation of one variable, two-way interactions, etc. Similarly, squaring Equation (25) and integrating gives relations for the variances

$$V_i(x_i) = \mathrm{Var}_{x_i}\left(\mathbb{E}_{x_{k\neq i}}(f|x_i)\right) \tag{29}$$

$$V_{ij}(x_i, x_j) = \mathrm{Var}_{x_{ij}}\left(\mathbb{E}_{k\neq i,l\neq j}(f|x_i, x_j)\right) - V_i - V_j \tag{30}$$

($k \neq i$ denotes that the expectation $\mathbb{E}$ is computed by integrating over all the x_k except for x_i) which normalised by the variance $\mathrm{Var}(f)$ are equal to the Sobol sensitivity indices S_i, S_{ij}, ..., respectively.

As noted by Huan et al [4, § 3.2], the Sobol indices may be expressed directly as sums of squares of the coefficients of a PCE. It will be evident that S_i gives a normalised measure of the sensitivity of the distribution of f to the parameter x_i, so that small S_i justifies neglect of x_i in subsequent analysis.

2.3 Efficient Use of Surrogates

Once a PC surrogate has been constructed, it can be used in Bayesian inference on model parameters and optimisation under uncertainty. The relevant mathematical techniques are

- Surrogate models in general.
- Bayesian inference, with application to model parameters
- Optimisation under uncertainty

2.3.1 Surrogate Models

The current trend for forward UQ (Section 2.2.2) in large-scale computational models, where the aim is to quantify the statistics of simulation outputs based on the assumed statistics of uncertain model parameters, has led to a need to perform large numbers of simulations in order to chart high-dimensional input parameter spaces (see, for example [20]). Brute-force application of Monte-Carlo using full accuracy *in-silico* models is impractical due to the requirement to perform possibly millions of simulations, if each is required for one sample since the convergence rate of MC scales as $\frac{1}{\sqrt{N}}$ for N samples. One solution is to view the simulation (regardless of its microscopic physics and myriad degrees of freedom) as a black box that takes input parameters and produces outputs, and then replace the full simulation by a *surrogate model* whose purpose is to reproduce the outputs of the simulator with an adequate degree of fidelity and at low cost. Surrogate models (also called response surfaces, metamodels, or emulators) must be calibrated using past knowledge of the simulator and by a limited number of full simulations for judiciously-chosen input parameter sets.

Modern methods for constructing surrogate models divide fairly neatly into regressions against sets of random variables, and machine learning based approaches. A few examples of both types follow (further information can be found by following the links in [21]).

Kriging also known as **Gaussian process regression**, involves the assumption that the underlying uncertainty in the sample data follows a Gaussian process, that is, a stochastic process $Z_x; x \in X$ such that for every finite set of indices $x_1, ..., x_K$

$$Z_{x_1,...,x_K} = (Z_{x_1}, ...Z_{x_K}) \tag{31}$$

is a multivariate Gaussian variable. An equivalent definition is the statement that there exist real-valued σ_{jk}, μ_j for any $s_1, ..., s_K \in \mathbb{R}$ such that the expectation is

$$\mathbb{E}\left(\exp\left(i\sum_{k=1}^{K} s_k Z_{x_k}\right)\right) = \exp\left(-\frac{1}{2}\sum_{j,k}\sigma_{jk}s_j s_k + i\sum_{j}\mu_j s_j\right). \tag{32}$$

The μ_j are the means of the process and the $\sigma_{jk} = C(x_i, x_j)$ the covariances. The latter are typically chosen from a standard dictionary of functions. Common choices correspond to well-known stochastic processes and include Gaussian white noise,

$$C(x_i, x_j) = \sigma^2 \delta(x_i - x_j); \tag{33}$$

Ornstein-Uhlenbeck, which is used to describe the velocity of a particle under the influences of damping and Brownian motion,

$$C(x_i, x_j) = \exp\left(-\frac{|x_i - x_j|}{L}\right); \tag{34}$$

and Matérn,

$$C(x_i, x_j) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\sqrt{2\nu}\frac{x_i - x_j}{L}\right)^{\nu} K_\nu\left(\sqrt{2\nu}\frac{x_i - x_j}{L}\right) \tag{35}$$

which has a number of desirable properties, for example the ability to model a process that is discontinuous at a chosen order of derivative (K_ν is the modified Bessel function of the second kind). In the above Equations(33) and (35), σ, L and ν are examples of hyperparameters, which are typically determined by use of Bayesian methods.

In *simple kriging* the mean μ is known in advance and the estimator for the value of the process at point $\mathbf{x_0}$ is constructed as a weighted linear combination of the existing set of N samples

$$Z^*(\mathbf{x_0}) = \mu + \sum_{\mathbf{i=1}}^{\mathbf{N}} \mathbf{w_i}\left(\mathbf{Z}(\mathbf{x_i}) - \mu\right); \tag{36}$$

minimisation of the prediction variance leads to a simple expression (actually a least-squares estimator) for the weights in terms of the matrix of covariances between the sample points $C_{ij} \equiv C(x_i, x_j)$ and the vector of covariances between the estimation point and the sample points $D_i \equiv C(x_0, x_i)$ as $\mathbf{w} = \mathbf{C^{-1}D}$.

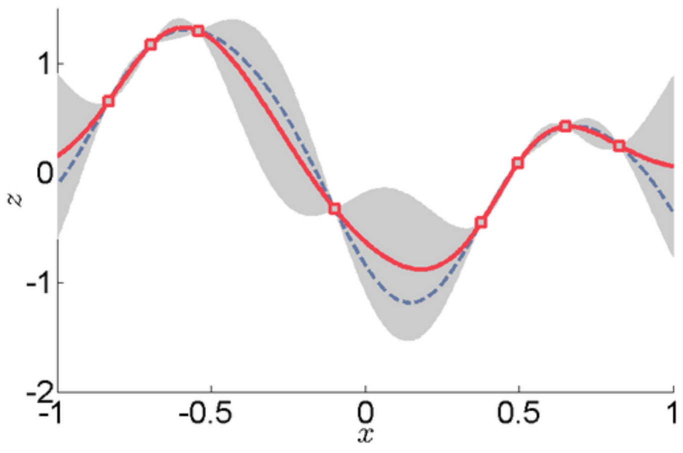

Figure 4: Kriging estimator (red curve) and its associated standard deviation (grey). The dotted line shows a plausible spline fit that nevertheless departs from the Maximum Likelihood kriging fit. Taken from [22].

In the case of a constant unknown mean, one has *ordinary kriging* (note it still involves the assumption that the mean is constant), and the formulae above are supplemented by a condition that the estimator be unbiased (this is simply the constraint that the weights sum to unity), which can be incorporated using a Lagrange multiplier. The variance of the estimator can be computed, leading to an intuitive plot that resembles a string of sausages (Figure 4). There are successively more involved kriging varieties based on further generalisation, for example general polynomial models of the underlying trend. Note that there is also gradient-enhanced kriging, which uses an estimate of the local gradient to reduce the variance of a kriging estimator.

Generalised Polynomial Chaos (gPC) involves expanding model outputs Y in terms of orthogonal polynomials of random variable inputs X (orthonormality being defined with respect to the measure associated to the distribution of the X - e.g. Hermite polynomials in the normal-distributed case). Construction of the surrogate is then reduced to the task of determining the coefficients in the expansion (which obviously is truncated at some polynomial order); this can be accomplished by means of least-squares analysis. Once the coefficients are available, the truncated PCE can be sampled at negligible cost and thus the full probability density function of the output is available, as is access to the sensitivity analysis via Sobol indices which represent the portion of the output variance due to each input parameter or combination of parameters (see Section 2.1 for more details).

Artificial Neural Networks (ANNs) are models consisting of arrays of interconnected nodes (artificial neurons) that react to inputs in a non-linear way. The neurons (nodes) and connecting edges have *weights* associated to them that control the output for a given set of inputs: these weights are dynamically adjusted in response to training feedback taking the form of an error measure, a process called *unsupervised learning*. Such general models abandon any attempt to match the model to the specific problem, with specificity emerging as a result of what is usually extensive training.

The input to a typical neural network is a vector of elements x_k, which are combined by a series of linear filters to give inputs to the *hidden unit* neurons

$$h_j = \sum_k w_{jk} x_k, \tag{37}$$

the output of which is a function of the input

$$X_j = g(h_j), \tag{38}$$

where the *activation functions* g are nonlinear. Typically chosen to suppress large outputs, they are frequently sigmoid in form; one such choice is $\tanh(\beta h)$. The topology of the network often corresponds to a number of 'layers' followed by a final set of output nodes, see [23], [24, § 6.7].

Such a generality of models provides potential surrogates for a wide range of problems including situations where the model exhibits discontinuities in its derivatives. The selection of an appropriate member of the *neural network zoo* [25] is a further challenge. Both the 'zoo' and ref [24, § 6.7] give brief indications as to which ANN topologies might be most appropriate. Within a given topology, it of course desirable to identify the number of minimum number of neurons needed, although this is probably less critical in application to NEPTUNE.

Reduced-order models (ROMs) are commonly produced using projection methods in which the dynamics of the full forward problem are projected onto a reduced-dimensional subspace. The latter is generally assembled using proper orthogonal decomposition or reduced basis methods, and using a set of full simulation outputs ('snapshots'). The success of these methods is connected with the merits of 'diagonalising' the problem to a set of uncorrelated degrees of freedom and identifying the components that have the biggest eigenvalues, a procedure known generically as principal components analysis. As with the methods outlined above, the reward is a more parsimonious way of calculating the model dynamics, see [26]. These models will be discussed in more detail as part of the Milestone Report M2.5.2 scheduled for August 2021.

2.3.2 Bayesian Inference as Applied to Model Parameters

The general problem of modeling a physical system involves the need to fit a model of some kind to existing data: there is a model architecture, m, with a set of adjustable parameters θ, as well as some measured data x, and the goal is to choose (or, in statistics jargon, *infer*) θ so as to provide the best agreement to x. This is also referred to as "Inverse UQ", to be contrasted with "Forward UQ" where the input distributions are given and the aim is to estimate the the uncertainties in the outputs.

The Bayesian formulation of this inverse problem relies on evaluating the θ that are most likely, given the model, the measured data, and, crucially, our prior beliefs about what parameters are plausible - leading to *conditional probability*. The fundamental tool for performing this sort of 'turnaround' of conditional probabilities is *Bayes' rule*, which gives us the following quantity to maximise, called the *posterior distribution*:

$$\max\left(p(\theta|\mathbf{x},m)\right)=\max\left(\frac{p(\mathbf{x}|\theta,m)p(\theta|m)}{p(\mathbf{x}|m)=\int p(\mathbf{x}|\theta,m)p(\theta|m)\mathbf{x}d\theta}\right), \tag{39}$$

the right-hand side of which (whose term $p(\mathbf{x}|\theta,m)$ represents the probability of having obtained the data $\mathbf{x}$ given parameters θ and model m; $p(\theta|m)$ represents advance belief about which values of the parameters are reasonable - the Bayesian *prior*; the denominator being essentially a normalising factor) can be expressed illustratively as

$$\max\left(\frac{likelihood\times prior}{evidence}\right). \tag{40}$$

A full Bayesian model such as this involves, potentially, a great amount of work. Not only must a high-dimensional normalisation integration be performed, but there are *two* optimisation loops, one over the set of parameters, and also a maximisation over possible model architectures as well. To reduce the workload, the decision might be taken to use only one model architecture, leading to the problem

$$\max\left(p(\theta|\mathbf{x})\right)=\max\left(\frac{p(\mathbf{x}|\theta)p(\theta)}{p(\mathbf{x})}\right). \tag{41}$$

This is called *Maximum A Posteriori* (MAP) estimation (note that it uses the *modal* value - the maximum of the distribution). An additional simplification is to note that the normalisation integral in the denominator can be disregarded as it does not affect the position of the maximum.

Further assuming a uniform prior leads to the simplest kind of model estimation, *Maximum Likelihood* (which one might also say is not really Bayesian at all):

$$\max\left(p(\theta|\mathbf{x})\right)=\max\left(p(\mathbf{x}|\theta)\right). \tag{42}$$

Maximum Likelihood leads to the familiar and much-used least-squares error measure, which is the Maximum Likelihood estimator for data with normally distributed errors (another detail is that the sums of squared normal variables obey the chi-squared distribution ubiquitous in elementary goodness-of-fit testing). Model fitting using least-squares is achieved by using the matrix pseudo-inverse $\left(M^{\dagger}M\right)^{-1}M^{\dagger}$ to derive (few) model parameters from (many) data points, actually a case of singular value decomposition (SVD). There are other worthwhile tricks e.g. writing strictly positive parameters as the square of something else before fitting (this can be used to avoid getting e.g. negative fitted variance hyperparameters). For more details, see, for example, [23].

2.3.3 Optimisation Under Uncertainty (OUU)

The problem of Optimisation Under Uncertainty (OUU) is an expanding area in the wider field of Operational Research, and also in engineering where it may be the case that simple deterministic

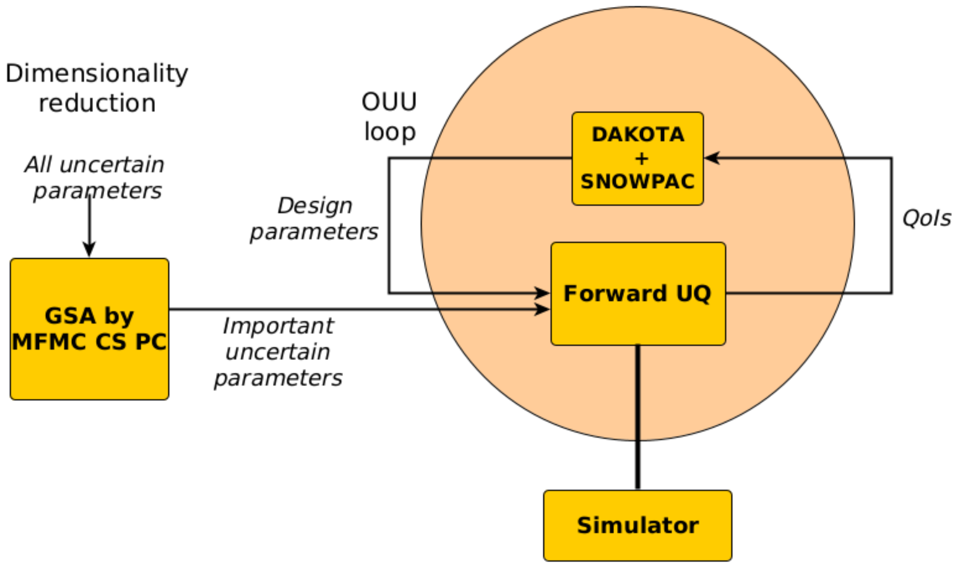

Figure 5: Workflow for optimisation under uncertainty (after [27]).

optimisations turn out not to be sufficiently robust when say devices are built with the tolerances typical of many construction methods. Obtaining solutions which provide the optimal value for a given ‘objective function’ normally involves many simulation runs, which now have to cover the set of parameters characterising uncertainty, thereby *‘inheriting and magnifying all the difficulties of forward UQ’* ([27]). It is clear that this motivates the use of lower fidelity and surrogate models to the fullest practical extent.

Najm and coworkers [13] have integrated the well-known DAKOTA package for UQ [3, 28] with the package SNOWPAC (Stochastic Nonlinear Optimisation with Path-Augmented Constraints) to perform OUU, see the example in Figure 5. The workflow begins on the left, with an offline model reduction using global sensitivity analysis (GSA), compressed sensing techniques (CS) applied to polynomial chaos expansions (PCE) and multi-level Monte-Carlo (MLMC) as described in the preceding Section 2.1 and Section 2.2. The inclusion of uncertainty means that the objective function and the constraints have to become functions of the parameters θ_i. In ref [13] the objective function is eg. taken to be a robustness measure given by a linear combination of the expectation value and the standard deviation i.e. $\mathbb{E}(Q) + c\mathrm{Var}[Q]$, for some constant $c > 0$. The optimisation driven by the DAKOTA plus SNOWPAC combination, incorporates forward uncertainty quantification using MLMC or adaptive sparse quadrature multi-level multi-fidelity (ASQ-MLMF) modeling. Note the ‘simulator’ used during the optimisation is likely to be a surrogate model.

3 Outline Implementation

The work of ref [4] is important for NEPTUNE in that simultaneous use is made of both a 2-D and a 3-D fluid model, referred to as MFMC (Multi-Fidelity Monte-Carlo, Section 2.1.3). Samples from different planes are used to produce a low-fidelity 2-D model with stochastic parameters as described in ref [4, § 4.1], using Bayesian Inference as described in Section 2.3.2. There is an important physical underpinning that the flow modelled by Huan et al [4] is primarily in a single direction, variation in which is neglected to produce the 2-D model. This is encouraging for NEPTUNE, as similar arguments relating to variation along the magnetic field are used to justify use of 2-D edge codes for the tokamak scrape-off layer (SOL).

Indeed, for SOL application, it is conceivable that multi-fidelity work involving 1-D fluid models may be efficient, when it is remembered that the so-called 'Onion-skin' models [29], have found a good deal of utility in SOL physics. The 'onion' is the plasma imagined to consist of separate skins, ie. layers delineated by equilibrium flux surfaces. Typically experiment provides boundary conditions at the wall for transport along the flux tubes of which each such layer may be supposed to be composed. Thanks to the axisymmetry of the magnetic field it is only necessary to consider a 1-D problem for each tube. The solutions of these 1-D problems for the different layers are then combined to produce a 2-D 'onion-skin' solution for which agreement to within $20\,\%$ of an explicitly 2-D fluid model is obtained in many cases [29].

The 3-D fluid models of NEPTUNE approximate in turn 5-D gyro-averaged or 6-D full phase-space dynamics. These more complex models may be needed particularly when the plasma collisionality is low, but then there is a change in physical emphasis in that the influence of the boundaries may be directly transmitted throughout the low collisionality region. For the spatially 5-D/6-D models, even at higher collisionality, theoretical analysis produces series of correction terms to be added into the fluid models. These are in addition to other physical effects such as plasma-neutral particle interaction, radiative loss etc., all adding up to a substantial additional challenge for usage of MFMC in NEPTUNE.

The works of Najm and collaborators [4, 13] have delivered a workflow primarily aimed at producing at an optimal design under uncertainty:

1. Global sensitivity analysis, to identify key parameters, see Section 2.1
2. Forward UQ using adaptive sparse sampling, see Section 2.2
3. PCE surrogate for inverse UQ and OUU, see Section 2.3

Whilst the above workflow has very sophisticated components, it does not cover all NEPTUNE applications. For instance, the physicist seeking to gain better understanding of the SOL may not be immediately interested in a particular optimisation. Such usage might require only relatively simple scans in one or two parameters to compare with theory, for which a local sensitivity approach (ie. examining the effect of small changes to other parameters one at a time) could provide adequate UQ.

More demanding is to improve understanding of the often highly nonlinear dynamics of the SOL, where it is possible for small-scale structures such as internal layers or local hot-spots to form. An example of the former might be the radiative fronts which develop as the plasma detaches from the first walls, where it could conceivably be critical for the numerical solution to resolve these fronts. One approach to reduce the uncertainty as to whether the discrete representation is adequate is ‘bifurcation-tracking’ whereby a branch of solutions is followed from where it emerges by initial linear instability on well-characterised physical scales, into the nonlinear regime. The physicist user may thus, after examining surrogate behaviour, want NEPTUNE to produce many more higher fidelity solutions. These solutions could usefully provide feed-back that the parameters of the surrogate require modification, and hence this user’s particular workflow might become very involved as it seeks to ‘home in’ on detachment processes.

Hence there are different potential workflows aimed at optimising designs, improving physical understanding, and indeed to produce optimal surrogates for use in device control [24]. The final decisions as to what should be implemented in NEPTUNE, await the production of the Milestone Reports M2.4.2 and M2.5.2 scheduled for August 2021.

4 Summary

This report has arranged the statistical and applied mathematical techniques of UQ according to a likely NEPTUNE workflow for device optimisation. Other workflows have been considered, see Section 3 where a need is recorded to treat code validation against theoretical models including comparison with experiment.

Other authors arrange the subject of UQ rather differently, thus the textbook of Smith [1] treats additional topics under the headings of “Model Calibration” and “Parameter Selection”, which loosely correspond to Stages 1 and 2 respectively, “Uncertainty Propagation” which corresponds to Stage 2 and “Model Discrepancy” which corresponds to Stage 3. Regarding the COSSAN software [2], there is a toolbox arrangement. Thus sampling (grouped with interval/subset methods under the heading of “Reliability”), meta-modelling, sensitivity and optimisation are treated under separate headings. Under each heading there is found a range of possible tools or methods, partly to provide a check, but also because newer methods may be worth exploration, or because the best method may be problem dependent. For instance, for sparse system identification, SLSQT (see Section 2.1.2) proved very successful for Brunton et al [8], but selecting an appropriate threshold value for SLSQT may not as easy in other applications. Each software tool may employed in different combinations with the others at different stages, eg. optimisation may be used both in Stage 2, to fit surrogate models better, and in Stage 3 in conjunction with sampling to treat uncertainty, to improve a design.

The nub of UQ is finding an affordable, good surrogate for the full model. The applicability of a lower-dimensional surrogate model often results from the fact that a physical system, in normal operation, behaves ‘smoothly’ ie. the sensitivity of its output function to changes in input parameters is rather low. The work of Trefethen [30, 31, 32] indicates that approximately $80\,\%$ of functions encountered in practice (most likely meaning related to use of the matlab^{TM} software) may be efficiently and accurately approximated as sums of products of functions of a single variable, suggesting they are representable by say, of order 100-$1000d$ samples if there are d parameters. Nonlinear systems exhibiting bifurcation self-evidently do not behave smoothly, particularly when deterministic chaos arises as an infinite sequence of bifurcations, and lack of smoothness also may arise due to a failure of numerics, such as the overfitting of functions by polynomial interpolation known as the Runge phenomenon.

Unfortunately there appears to be no easy way to determine whether an arbitrary output function can be approximated succinctly over the whole range of interest. A good approach is to start by examining local approximations and seeking to patch them together to cover the whole ranges, so called ‘Machine Learning ROMs’ [24, § 12.7]. Techniques of this kind, eg. also ‘Reservoir Computing’ [33] are currently the subject of much research, for which it is expected that the timing of the appearance of the Milestone Reports M2.4.2 and M2.5.2 reports will be such as to permit them to provide a more complete description.

Acknowledgement

The support of the UK Meteorological Office and Strategic Priorities Fund is acknowledged.

5 Appendices

5.1 Quasi-Monte-Carlo

The principal source for Quasi-Monte-Carlo methods is the book by Niederreiter [34], which however is highly mathematical in tone. This section should also serve as an informal introduction to Niederreiter's book.

In 1-D, it is easy to show that the discrepancy (the error made in calculating the size of an object based on the sampling) is at least $\epsilon_N \propto 1/N$ when the sample points $\mathbf{x}_l$ are uniformly distributed. The key fact is that there exist sets of points ('low discrepancy sequences') for which the discrepancy does not increase very much as the number of dimensions increases.

The simplest of these sets to describe is that due to Halton. In 1-D, it is identical to a van der Corput sequence, which involves generating numbers on the unit interval, using the reversed bit patterns of the positive integers. It is best illustrated by example. Thus 2 has the binary representation 10, so the 2nd element in the van der Corput sequence is $.01_2$ or $1/4$, $3 = 11_2$ so the 3rd element in the van der Corput sequence is $.11_2$ or $3/4$, $4 = 100_2$ so the 4th element in the van der Corput sequence is $.001_2$ or $1/8$. Hence the first 7 elements of the van der Corput sequence are (in eighths)

$$4, 2, 6, 1, 5, 3, 7 \tag{43}$$

It will be seen that there is a kind of fractal pattern about the above distribution. Van der Corput sequences may be defined for any prime b, by representing the integers in the base b, then using the reversed representation as above to generate values on the unit interval. The Halton sequence in 2-D contains pairs of numbers, the first in the lth pair being the lth element from a van der Corput sequence with base 2 and the second being the corresponding element in a base 3 van der Corput sequence. In fact any two distinct primes could be used, and the generalisation to many dimensions should be obvious.

The discrepancy of the Halton sequence in 2-D is bounded by a formula which may be approximated as

$$\epsilon_N = A_2 (\ln N)^2 / N \tag{44}$$

where $A_2 = 0.66$. It will be seen that this is smaller than the Monte-Carlo value of $\epsilon_N = 1/\sqrt{N}$ for $N > 1000$. It is therefore also competitive with uniform sampling on a rectangular lattice of N 2-D points, which in general gives an error proportional to lattice spacing, ie. $1/\sqrt{N}$.

The explanation for the superior performance of the Halton sequence is illustrated by comparison between Figure 6 and Figure 7. Both show 100 2-D vectors on the unit square, in fact the last hundred of a length 2000 vector sequences starting at zero. In Figure 6, the components of the random vectors are generated using the `ranlux` random number generator (luxury level of 3 where 4 is the highest) supplied by F. James [35]. Figure 7 plots the 2-D Halton sequence generated with base-2 and base-3 van der Corput sequences, using a modified version of software due to J. Burkardt [36] based on ref [37]. It will be seen that, comparing the number of points which are close together, the random vectors are much less uniformly distributed than the Halton sequence.

The Halton sequence is important because it can be defined without setting a definite number of samples N_a in advance, hence it can be used just like sequences from standard Monte-Carlo

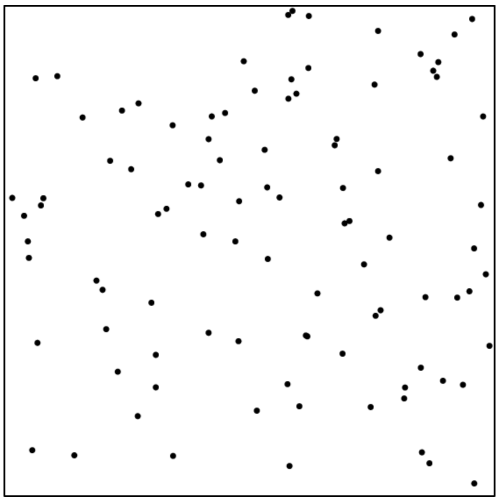

Figure 6: Two-dimensional vectors plotted by position in the unit square. The `ranlux` random number generator was used to produce 200 coordinates, paired to make 100 vectors.

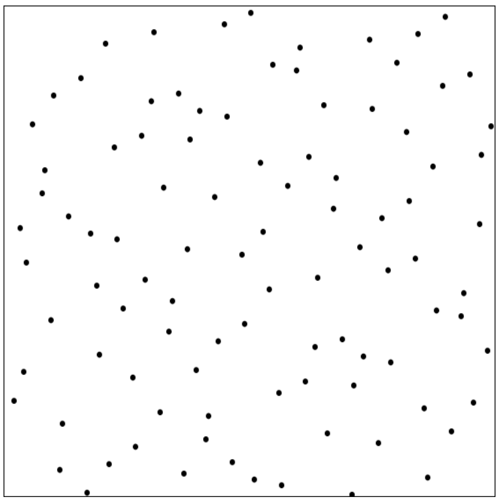

Figure 7: Two-dimensional vectors plotted by position in the unit square. The `halton` quasi-random number generator was used to produce 100 vectors.

random number generators. This contrasts with the Hammersley sequence, where the first component of each vector is l/N_a, then other components contain van der Corput sequences. However, much of Niederreiter's book is concerned with cases where N_a is specified in advance, when it transpires for example that A_2 may be reduced to as little as 0.26, using a (t, s) sequence.

The definition of (t, s) sequences is quite involved. The main idea is to generate the quasi-random sequences as sets of vectors of dimension s (hence the s in the name) rather than as in Halton where the components of the vectors are computed independently as 1-D van der Corput sequences. The t denotes the fact that, instead of each integer being represented in base b prior to bit reversal, it is represented in base b^{t+1}. Hence, in the (t, s) notation, van der Corput sequences are $(0, 1)$ sequences, and anyway apparently $t = 0$ has optimally small error.

Returning to Halton sequences, the expression for discrepancy in d-D is

$$\epsilon_N = A_d(\ln N)^d/N \tag{45}$$

where the values of A_d are tabulated in Niederreiter's Table 4.4. [34, p.96]. The first few are $A_d = 0.66, 0.82, 1.26, 2.62,$ for $d = 2, 3, 4, 5$. At higher d, A_d increases rapidly so that for example, $A_{12} = 16\,800$ and $A_{20} = 6.62 \times 10^{10}$, whereas the corresponding value C_d for the best (t, s) sequence *decreases* almost as rapidly ($C_6 = 0.0186$). This is the benefit gained from the greater complexity of calculating the (t, s) sequences.

Despite the theory, much practical experience with QMC as reported and performed by Morokoff *et al* [38, 39] suggests that QMC frequently performs little better than MC when d is large. Morokoff *et al* considered Halton sequences and (t, s) sequences in base $b = 2$ (referred to as Sobol sequences) and in other prime number bases (Faure sequences). It seems that for the variety of methods tested, the number of sample points required for the asymptotic discrepancy estimates to be realised can increase exponentially with dimension d, so the curse of dimensionality remains. There is evidence however that in some problems, where only a relatively small number of dimensions is apparently important, QMC performance can be greatly improved [40]. One important point to make is that Morokoff *et al* tended to explore more those QMC rules which had the best asymptotic behaviour. Their results do not exclude the possibility that the simpler techniques of lattice rules, see ref [34, $\S$5.1], might have better behaviour in practice, especially if the lattice basis $\mathbf{g}$ is well chosen. Note that NAG uses the simple Korobov rule, in which the components of $\mathbf{g}$ are given as

$$g_k = a^{(k-1)}(\mod p) \tag{46}$$

for specially chosen real $a > 1$ and prime number p.

5.2 Latin Hypercube Sampling

One of the cheapest sampling strategies is stratified sampling. The easiest variant to describe, viz. Latin hypercube [41] is illustrated in Figure 8, after Steinberg as quoted in ref [42, $\S$4]. The sample space is gridded, but now sample points are required to lie within the grid squares, and in fact their precise location within a square in determined randomly. The selection of grid-square is also random, but heavily constrained by the fact that no other sampled square should have the same discrete co-ordinate values in any co-ordinate. For example, in Figure 8, the fact that there is a sample point in the square three along from the left in the bottom row, discrete coordinate $(3, 1)$,

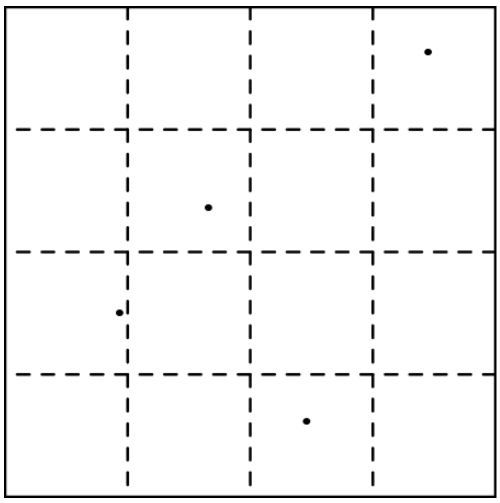

Figure 8: Latin hypercube sampling.

means that no sample will be placed in cells with coordinates $(3, j)$ or $(i, 1)$ for any i or j. If there are N_1 cells in each coordinate, the total number of samples needed is also N_1. This is very cheap, but the method is known to become unreliable when the parameters' variations are correlated.

References

[1] R.C. Smith. *Uncertainty Quantification: Theory, Implementation, and Applications*. SIAM, 2014.

[2] E. Patelli, M. Broggi, M. de Angelis, and M. Beer. OpenCossan: An efficient open tool for dealing with epistemic and aleatory uncertainties. In M. Beer, S.-K. Au, and J.W. Hall, editors, *Vulnerability, Uncertainty, and Risk : Quantification, Mitigation, and Management*, pages 2564–2573. American Society of Civil Engineers, 2014. DOI 10.1061/9780784413609.258.

[3] B.M. Adams, M.S. Ebeida, M.S. Eldred, J.D. Jakeman, L.P. Swiler, J.A. Stephens, D.M. Vigil, T.M. Wildey, W.J. Bohnhoff, K.R. Dalbey, J.P. Eddy, , K.T. Hu, L.E. Bauman, and P.D. Hough. DAKOTA, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 6.0 User's Manual. Technical report, Sandia Technical Report SAND2014-4633, Sandia National Laboratories, Albuquerque, NM, 2014.

[4] X. Huan, C. Safta, K. Sargsyan, G. Geraci, M.S. Eldred, Z.P. Vane, G. Lacaze, J.C. Oefelein, and H.N. Najm. Global sensitivity analysis and estimation of model error, toward uncertainty quantification in scramjet computations. *AIAA Journal*, 56(3):1170–1184, 2018.

[5] N. Wiener. The homogeneous chaos. *American Journal of Mathematics*, 60(4):897–936, 1938.

[6] W. Arter. Equations for EXCALIBUR/NEPTUNE Proxyapps. Technical Report CD/EXCALIBUR-FMS/0021-1.00-M1.2.1, UKAEA, 2020.

[7] D. Xiu. *Generalized (Wiener-Askey) Polynomial Chaos*. PhD thesis, Brown University Providence, RI, 2004.

[8] S.L. Brunton, J.L. Proctor, and J.N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937, 2016.

[9] B. Peherstorfer, K. Willcox, and M. Gunzburger. Optimal model management for multifidelity monte carlo estimation. *SIAM Journal on Scientific Computing*, 38(5):A3163–A3194, 2016.

[10] M.B. Giles. Multilevel Monte-Carlo methods. *Acta Numerica*, 24:259–328, 2016.

[11] S. Mishra, C. Schwab, and J. Šukys. Multi-level Monte-Carlo finite volume methods for uncertainty quantification in nonlinear systems of balance laws. In H. Bijl, D. Lucor, S. Mishra, and C. Schwab, editors, *Uncertainty quantification in computational fluid dynamics*, pages 225–294. Springer, 2013.

[12] A.-L. Haji-Ali and R. Tempone. Multilevel and multi-index monte carlo methods for the mckean–vlasov equation. *Statistics and Computing*, 28(4):923–935, 2018.

[13] G. Geraci, F. Menhorn, X. Huan, C. Safta, Y.M. Marzouk, H.N. Najm, and M.S. Eldred. Progress in scramjet design optimization under uncertainty using simulations of the hifire direct connect rig. *Conference paper; AIAA Scitech 2019 Forum*, 2019. AIAA paper 2019-0725.

[14] H.J. Bungartz and M. Griebel. Sparse grids. *Acta Numerica*, 13:147–269, 2004.

[15] T. Gerstner and M. Griebel. Dimension–adaptive tensor–product quadrature. *Computing*, 71(1):65–87, 2003.

[16] E. Patelli, D.A. Alvarez, M. Broggi, and M. de Angelis. Uncertainty management in multidisciplinary design of critical safety systems. *Journal of Aerospace Information Systems*, 12(1):140–169, 2015.

[17] NumPy Random Generator. `https://numpy.org/doc/stable/reference/random/generator.html#numpy.random.Generator`, 2020. Accessed: October 2020.

[18] M. Hegland. Adaptive sparse grids. *ANZIAM Journal*, 44(E):C335–C353, 2003.

[19] I.M. Sobol. Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates. *Mathematics and Computers in Simulation*, 55(1-3):271–280, 2001.

[20] B. Sudret and S. Marelli. Surrogate Models for Uncertainty Quantification: an Overview. `https://www.researchgate.net/publication/317396793_Surrogate_models_for_uncertainty_quantification_An_overview`, 2017. European Conference on Antennas and Propagation (EUCAP, 2017).

[21] Surrogate model wiki. `https://en.wikipedia.org/wiki/Surrogate_model`, 2020. Accessed: October 2020.

[22] Kriging wiki. `https://en.wikipedia.org/wiki/Kriging`, 2020. Accessed: October 2020.

[23] N. Gershenfeld. *The Nature of Mathematical Modeling*. Cambridge University Press, 1999.

[24] S.L. Brunton and J.N. Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. CUP, 2019.

[25] Neural network zoo website. `https://www.asimovinstitute.org/neural-network-zoo/`, 2020. Accessed: October 2020.

[26] Model order reduction wiki. `https://en.wikipedia.org/wiki/Model_order_reduction`, 2020. Accessed: October 2020.

[27] H. Najm. Uncertainty Quantification in Computational Models of Large Scale Physical Systems. `https://www.osti.gov/servlets/purl/1593073`, 2018. Seminar, NRC Institure of Marine Engineering, Rome, Italy.

[28] DAKOTA website. `https://dakota.sandia.gov`, 2020. Accessed: October 2020.

[29] P.C. Stangeby, J.D. Elder, W. Fundamenski, A. Loarte, L.D. Horton, R. Simonini, A. Taroni, O.F. Matthews, and R.D. Monk. Code-code comparisons of DIVIMP's 'onion-skin model'and the EDGE2D fluid code. *Journal of nuclear materials*, 241:358–362, 1997.

[30] A. Townsend and L.N. Trefethen. Continuous analogues of matrix factorizations. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 471(2173):20140585, 2015.

[31] B. Hashemi and L.N. Trefethen. Chebfun in three dimensions. *SIAM Journal on Scientific Computing*, 39(5):C341–C363, 2017.

[32] L.N. Trefethen. John von Neumann Prize Lecture. Off-the-cuff remarks, July 6–17, 2020. The Second Joint SIAM/CAIMS Annual Meeting (AN20).

[33] J. Pathak, A. Wikner, R. Fussell, S. Chandra, B.R. Hunt, M. Girvan, and E. Ott. Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 28(4):041101, 2018.

[34] H. Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*. Society for Industrial Mathematics, 1992.

[35] F. James. RANLUX: a Fortran implementation of the high-quality pseudorandom number generator of Luscher. *Computer Physics Communications*, 97(3):357–357, 1996.

[36] J. Burkardt. The halton quasirandom sequence. URL http://people.sc.fsu.edu/˜jburkardt/f_src/halton/halton.html.

[37] B.L. Fox. Algorithm 647: Implementation and relative efficiency of quasirandom sequence generators. *ACM Transactions on Mathematical Software (TOMS)*, 12(4):362–376, 1986.

[38] W.J. Morokoff and R.E. Caflisch. Quasi-random sequences and their discrepancies. *SIAM Journal on Scientific Computing*, 15:1251, 1994.

[39] W.J. Morokoff and R.E. Caflisch. Quasi-Monte Carlo integration. *Journal of Computational Physics*, 122(2):218–230, 1995.

[40] R.E. Caflisch, W. Morokoff, and A. Owen. Valuation of mortgage backed securities using Brownian bridges to reduce effective dimension. *J. Computational Finance*, 1(1):27–46, 1997.

[41] M.D. McKay, R.J. Beckman, and W.J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 42(1):55–61, 1979.

[42] M.H. Kalos and P.A. Whitlock. *Monte Carlo Methods. Vol. 1: basics*. Wiley-Interscience New York, NY, USA, 1986.