Contents lists available at ScienceDirect

Applied Mathematical Modelling

journal homepage: www.elsevier.com/locate/apm

Subset simulation for probabilistic computer models

P.O. Hristov[a,*], F.A. DiazDelaO[b]

[a] Institute for Risk and Uncertainty, School of Engineering, University of Liverpool, Liverpool L69 7ZF, UK
[b] Clinical Operational Research Unit, Department of Mathematics, University College London, London WC1H 0BT, UK

A R T I C L E I N F O

Article history:
Received 26 July 2022
Revised 28 March 2023
Accepted 30 March 2023
Available online 2 April 2023

Keywords:
Reliability analysis
Subset simulation
Probabilistic numerics
Partially-converged simulations

A B S T R A C T

Reliability analysis can be performed efficiently through subset simulation. Through Markov chain Monte Carlo, subset simulation progressively samples from the input domain of a performance function (typically a computer model) to find the failure domain, that is, the set of input configurations that result in an output higher than a prescribed threshold. Recently, a probabilistic framework for numerical analysis was proposed, whereby computation is treated as a statistical inference problem. The framework, called probabilistic numerics, treats the output of a computer code as a random variable. This paper presents a generalisation of subset simulation, which enables reliability analysis for probabilistic numerical models. The advantages and challenges of the method are discussed, and an example with industrial application is presented.

1. Introduction

Probabilistic reliability analysis (PRA) [1] aims to assess the reliability of physical systems whilst accounting for the underlying uncertainties. The system's behaviour is modelled as a *performance function*, i.e., a mapping between uncertain inputs and outputs. A central problem in PRA is the identification of the *failure domain* $\mathcal{F}$, i.e., the set of input configurations that result in a performance function output that exceeds a critical threshold.

Identifying $\mathcal{F}$ for complex, realistic physical systems requires extensive, potentially expensive experimentation. Computer models, also known as simulators, are commonly employed to reduce this cost. However, this brings further unknowns into the analyses. For example, a number of assumptions about the form of the model may be required to make its use feasible, which turn any simulator into an idealised version of the physical system. When calibrating such models, this type of uncertainty, termed *model discrepancy* or *model inadequacy* [2,3] is considered internal to the model and often remains neglected in the analysis of computer code results.

This paper focuses on another type of error, which stems purely from the numerical nature of the model. For instance, errors due to discretisation of a continuous geometry, or errors that propagate when using iterative solvers. Such errors are present even in well-posed, validated numerical simulations [4]. Traditionally, these errors are studied in the field of numerical analysis, where theoretical convergence rates and bounds are derived [5]. These bounds are usually conservative and rarely lend themselves to further propagation. This can be especially problematic in computational pipelines, where the output of one model and its associated uncertainty are fed as an input to another, possibly several times over. These

* Corresponding author.
E-mail address: p.hristov@liv.ac.uk (P.O. Hristov).

https://doi.org/10.1016/j.apm.2023.03.041

computational pipelines arise in, for example, the analysis of complex systems such as aircraft. The complexity can be due to the requirement to analyse intricate phenomena using, for example, coupled aeroelastic models of the fuselage, wings and tail plane. Since computer models run on finite computational resources, they are bound to make numerical errors.

Instead of traditional numerical analysis, an alternative approach to the quantification of computational uncertainty is to analyse it *probabilistically*. Probabilistic numerics (PN) [6] aims to provide explicit estimates of numerical error that propagate through computational pipelines. The governing principle in PN is to think of computation as a statistical inference problem. This means estimating a quantity that is unknown due to numerical uncertainty. This way, the framework of statistics is used to assign a measure of uncertainty to the output of computer models. This provides quantifiable information that can be used to identify computational bottlenecks and inform analysts about sources of numerical error, which could potentially be controlled. Based on this approach, it is now possible to formulate probabilistic versions of computational tasks that underpin numerical analysis. Integration [7,8], stochastic optimisation [9], and the solution of differential equations [10] and linear systems [11,12] are relevant examples.

In order to estimate a system's probability of failure, knowledge of the failure domain $\mathcal{F}$ is required. However, if the system is carefully designed against failure, the volume of $\mathcal{F}$ can be several orders of magnitude smaller than that of the original input domain. Moreover, it may exhibit a complex geometry or be disconnected. This makes identifying $\mathcal{F}$ very challenging. Subset simulation (SuS) [13] is an advanced Monte Carlo method, specifically conceived for engineering PRA. The method offers several advantages over other techniques, among which are its efficiency and scalability [14]. By construction, SuS works with deterministic computer code outputs and has no provision of including probabilistic description to the data generating process. This fact presents two limitations. Firstly, the potentially very small size of $\mathcal{F}$ means that even small deviations in the values of the model output might change the estimated geometry of $\mathcal{F}$, resulting in a biased estimate of the probability of failure. Recognizing and accounting for uncertainty in the output is therefore very important for PRA. Secondly, it is often prohibitively time-consuming to run the underlying model to convergence a sufficient number of times to reliably estimate the probability of failure. Probabilistic numerics allows the use of partially converged simulations, whilst quantifying the output uncertainty resulting from having an incomplete solution. It is currently not possible to use SuS directly with such information.

The main contribution of this paper is to tackle the above limitations of SuS by introducing *probabilistic subset simulation* (P-SuS), an extension of SuS capable of working with probabilistic numerical codes. The proposed algorithm takes into account the probabilistic nature of the output of the simulator, accounting for the computational uncertainty in the identification of $\mathcal{F}$ and the estimation of the probability of failure. The advantages of P-SuS are motivated using PN methods for the solution of sparse linear systems, but the proposed algorithm is generally applicable to any computational model possessing a probabilistic output.

In the case of partially-converged simulations, P-SuS can be used as an efficient pre-processor to a converged SuS-based PRA, in which input combinations far away from $\mathcal{F}$ are discarded at a much lower cost than when using converged simulations and SuS. The paper discusses the proposed method in detail, and demonstrates how P-SuS is a generalisation of SuS. That is, the estimate for the probability of failure converges to that of SuS in the absence of computational uncertainty.

The paper is organised as follows. Section 2 describes PN methods and SuS to a level of detail required for understanding P-SuS. Section 3 introduces the proposed approach in detail and discusses some of its properties. In Section 4, the performance of the algorithm is demonstrated through a low-dimensional example and a realistic finite element model (FEM) of an aircraft wing box. Finally, Section 5 draws some conclusions and outlines directions for future work.

2. Methodology overview

2.1. Subset simulation

Let $h : \mathcal{X} \subseteq \mathbb{R}^d \to \mathbb{R}$ be a *performance function*, namely, a function that models the behaviour of a physical system by mapping the inputs that determine such behaviour onto the system's response. When the output $y = h(\boldsymbol{x})$ at a particular input combination $\boldsymbol{x} \in \mathcal{X}$ exceeds a prescribed safe operational threshold t^*, failure is said to occur. Hence, all combinations of input parameters leading to failure define the *failure domain*, $\mathcal{F} \subset \mathcal{X}$. In this setting, a *failure event* is $F = \{Y > t^*\}$, where Y is a random variable associated with the system output y. This random variable captures the presence of uncertainties in the system geometry, material properties, incomplete physical understanding, and inherent variability of the system itself. Consequently, the system inputs $\boldsymbol{x}$, are modelled with the random variable X, with a joint probability density function (PDF) $g_X(\cdot)$ and cumulative distribution function (CDF) $G_X(\cdot)$. Whenever clear the subscripts denoting the random variable will be omitted for notational simplicity. PRA defines the *probability of failure* as

$$p_F \equiv \mathbb{P}(F) = \int_{\mathcal{F}} dG(\boldsymbol{x}, \cdot) = \mathbb{E}[\mathbb{I}_{\mathcal{F}}(\boldsymbol{x})]. \tag{1}$$

where the indicator function $\mathbb{I}_{\mathcal{F}}(\boldsymbol{x})$ equals 1 if $\boldsymbol{x}$ belongs to $\mathcal{F}$ and 0 otherwise.

When the system modelled by the performance function h is reliable, a failure event will be rare. Thus, the volume of $\mathcal{F}$ can be orders of magnitude smaller than that of $\mathcal{X}$ and the use of direct Monte Carlo is usually impractical. To overcome this, SuS models the failure domain $\mathcal{F}$ as contained in a sequence of m nested intermediate failure domains $\mathcal{F} = \mathcal{F}_m \subset \mathcal{F}_{m-1} \subset \ldots \subset \mathcal{F}_1 \subset \mathcal{F}_0 = \mathcal{X}$. Each intermediate failure domain has an associated intermediate failure event, also known as a *level*, such that $F_i = \{Y > t_i\}$ corresponds to the output exceeding an intermediate operational threshold, $t_{i-1} \leq t_i \leq t_{i+1}$ for $1 < i < m$. Given the nestedness of these events, the probability of failure can be computed as

$$p_F = \mathbb{P}\left(\bigcap_{i=1}^{m} F_i\right) = \mathbb{P}(F_1) \times \mathbb{P}(F_2|F_1) \times \ldots \times \mathbb{P}(F_m|F_{m-1}). \tag{2}$$

In order to sample from $\mathcal{F}$ and estimate the probability of failure, SuS samples from each intermediate failure event, progressively approximating $\mathcal{F}$. This is done by setting two initial parameters: the sample size at each level, denoted by N, and the level probability, denoted by p_0. In practice, the main driver in choosing N is the available computational budget. The value of p_0 directly affects the convergence properties of SuS, and is usually chosen as $p_0 = [0.1, 0.3]$ to minimize the coefficient of variation of the failure probability estimator, $\hat{p}_F^{SuS}$ [15]. The estimator of the failure probability is derived from Eq. (2), and is given by

$$\hat{p}_F^{SuS} = p_0^{m-1} \frac{1}{N} \sum_{k=1}^{N} \mathbb{I}(\boldsymbol{x}_k \in \mathcal{F}). \tag{3}$$

The algorithm requires that the values of p_0 and N are such that p_0N and $1/p_0$ are integers so that they can be used for sample indexing. SuS samples initially from the whole input space and adaptively constructs each intermediate failure event, such that the conditional probabilities in Eq. (2) remain equal to p_0. To achieve this, the performance function responses in level i are sorted in descending order to give the list $y_i^{(k)}$ for $k = 1, \ldots, N$. To ensure that $\mathbb{P}(F_i|F_{i-1}) = p_0$, the next intermediate threshold is given by the $(1 - p_0)^{\text{th}}$ quantile of the responses

$$t_{i+1} = \frac{y_i^{(p_0N)} + y_i^{(p_0N+1)}}{2}. \tag{4}$$

The above is the definition in the original SuS paper [13] and is commonly used in the literature. This does not mean that the intermediate threshold cannot be defined differently, for example by taking the current ${p_0}^{N-\text{th}}$ highest response, or $t_{i+1} = y_i^{(p_0N)}$. This flexibility will become useful when defining a probabilistic version of SuS. By construction, the top p_0N samples in $y_i^{(k)}$ have responses greater or equal to t_{i+1}. This guarantees that these samples already belong to the intermediate failure domain $\mathcal{F}_{i+1}$ and enables the generation of new samples from $\mathcal{F}_{i+1}$. The p_0N samples in $\mathcal{F}_{i+1}$ are used as seeds to generate independent Markov chains from the target PDF $g(\boldsymbol{x}, \cdot|F_{i+1}) \propto g(\boldsymbol{x}, \cdot)\mathbb{I}(\boldsymbol{x} \in \mathcal{F}_{i+1})$. The sample from $\mathcal{F}_{i+1}$ consists of $N_c = p_0N$ Markov chains, each with $N_s = N/N_c = 1/p_0$ samples. Since the seeds are already distributed according to the target distribution, $g(\boldsymbol{x}, \cdot|F_{i+1})$, there is no burn-in period, which is typically required in MCMC simulations to generate a single Markov chain. The process of generating nested intermediate failure events is repeated until at least p_0N samples are obtained from $g(\boldsymbol{x}, \cdot|F)$.

2.2. Bayesian conjugate gradient

For some applications, the performance function h can be described by a simple analytical expression. However, in realistic (e.g., industrial) settings, h is typically implemented as a computer model, or a series of coupled computer models. In engineering, for example, these computational pipelines consist of finite element (FE) or computational fluid dynamics (CFD) models. In those scenarios, the physical model usually takes the form of a system of coupled differential equations. These are discretised, and ultimately expressed as a linear system of the form $\boldsymbol{A}\boldsymbol{y}^* = \boldsymbol{b}$, where $\boldsymbol{A} \in \mathbb{R}^{d\times d}$ is a global system matrix, $\boldsymbol{b} \in \mathbb{R}^d$ is a forcing vector, and $\boldsymbol{y}^* \in \mathbb{R}^d$ is the solution vector. The matrix $\boldsymbol{A}$ is typically sparse and, for realistic systems, can be very large. For solving this kind of systems, iterative techniques such as the *conjugate gradient* (CG) method are suitable [16]. At the i^{th} iteration of CG, information about the solution vector $\boldsymbol{y}^*$ is provided by the *search directions* $\boldsymbol{s}_i$ (for $i = 1, \ldots, d$), which determine how the space is explored. In exact arithmetic, the method is guaranteed to find the solution after at most d iterations. However, in most practical applications d is very large and the solution gets polluted by round-off errors. This justifies the selection of another iteration threshold $n \ll d$, at which an approximate solution is obtained.

The Bayesian conjugate gradient (BCG) [12] is a probabilistic numerical method (PNM) that computes the posterior distribution of the solution vector $\boldsymbol{y}^*$, given partial information provided by search directions in the form

$$z_i \equiv \boldsymbol{s}_i^{\mathrm{T}}\boldsymbol{A}\boldsymbol{y}^* = \boldsymbol{s}_i^{\mathrm{T}}\boldsymbol{b} \tag{5}$$

After n iterations of the algorithm, the information about the solutions $\boldsymbol{y}^*$ is encoded in the vector $\boldsymbol{z}_n = [z_1, z_2, \ldots, z_n]^{\mathrm{T}}$. In BCG it is assumed that the products in Eq. (5) are computed in exact arithmetic, implying a likelihood model, $\boldsymbol{z}_n|\boldsymbol{y}$ which follows a Dirac distribution $\delta(\boldsymbol{z}_n - \boldsymbol{S}_n^{\mathrm{T}}\boldsymbol{A}\boldsymbol{y})$, where $\boldsymbol{S}_n$ is a matrix whose columns are given by the first n search directions.

The vector $\boldsymbol{y}$ is treated as a random variable, expressing the uncertainty about the solution vector $\boldsymbol{y}^*$. A prior Gaussian distribution for $\boldsymbol{y}$ is assumed, namely

$$\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{y}; \boldsymbol{y}_0, \boldsymbol{\Sigma}_0) \tag{6}$$

where y_0 and $\boldsymbol{\Sigma}_0$ are the prior mean and covariance respectively.

By defining the matrix $\mathbf{\Lambda}_n = \mathbf{S}_n^{\mathrm{T}} \mathbf{A} \mathbf{\Sigma}_0 \mathbf{A}^{\mathrm{T}} \mathbf{S}_n$ and the residual $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{y}_0$, it can be shown that the posterior distribution of $\mathbf{y}$ given $\mathbf{z}_n$ is

$$\mathbf{y}|\mathbf{z}_n \sim \mathcal{N}(\mathbf{y}; \mathbf{y}_n, \mathbf{\Sigma}_n) \tag{7}$$

where the posterior mean and variance of the multivariate normal are:

$$\mathbf{y}_n = \mathbf{y}_0 + \mathbf{\Sigma}_0 \mathbf{A}^{\mathrm{T}} \mathbf{S}_n \mathbf{\Lambda}_n^{-1} \mathbf{S}_n^{\mathrm{T}} \mathbf{r}_0 \tag{8}$$

$$\mathbf{\Sigma}_n = \mathbf{\Sigma}_0 - \mathbf{\Sigma}_0 \mathbf{A}^{\mathrm{T}} \mathbf{S}_n \mathbf{\Lambda}_n^{\mathrm{T}} \mathbf{S}_n^{\mathrm{T}} \mathbf{A} \mathbf{\Sigma}_0 \tag{9}$$

For more details on the design and implementation of BCG, the reader should refer to Cockayne et al. [12]. For this work, it is important to observe that when a PNM is used as part of a more general computational model, each response from the model will have its own probability distribution, determined by the probabilistic numerical method itself. That is, the PNM determines $G_Y(\cdot)$ for the model output, whereas evaluating the model at some input combination x_i, gives y_i, the collection of parameters describing the joint distribution of the discretised model output, such that $Y_i \sim G_Y(y_i)$. This observation will become important in the definition of some of the features of P-SuS, described in the next section. In the case of BCG, $G_Y(\cdot)$ will be a normal or a Student-t distribution as described in Cockayne et al. [12]. However, since the linear system under consideration is a discretisation of a continuous mathematical model, using a PNM effectively casts its solution in the form of a stochastic process. This stochastic process is defined by the choice of PNM (in the case of BCG, a Gaussian process), which in turn defines the family of the marginal distributions.

3. Probabilistic subset simulation

This section introduces probabilistic subset simulation (P-SuS), an extension of SuS, designed to accommodate computational models using probabilistic numerical methods. The building blocks of P-SuS are discussed in turn, highlighting the modifications introduced and how each step relates to its SuS counterpart.

3.1. Response ranking

P-SuS starts the same way as SuS, namely, with an unconditional level (level 0), where Monte Carlo sampling populates the input domain with N samples $\mathbf{x}_1, \ldots, \mathbf{x}_N$ from the input distribution $G_X(\cdot)$. These samples are then evaluated using a probabilistic numerical computer model that encodes the performance function h, resulting in the data set $\mathcal{D} = \{(\mathbf{x}_j, y_j)\}_{j=1}^N$. The response y_j is not a realisation of the output random variable $Y(\mathbf{x}_j)$, as is the case for traditional stochastic models, but instead contains information about the probability distribution of the true value of the model output, $h(\mathbf{x}_j)$ at the input combination $\mathbf{x}_j$. Thus, it provides a parameter tuple such that $Y(\mathbf{x}_j) \sim G_Y(\cdot, y_j)$. For instance, if each response is distributed as a Gaussian random variable, $Y(\mathbf{x}_j) \sim \mathcal{N}(\mu_j, \sigma_j)$, then y_j is the tuple containing the mean, μ_j and standard deviation, σ_j of the Gaussian distribution. The interested reader is referred to the source material for detailed description of the methods [10,12,17]. In this paper, the output of h is assumed to be one dimensional or, if this is not true, that different outputs are independent. For ease of notation, $Y(\mathbf{x}_j)$ will be shortened to Y_j where this does not cause confusion. Lastly, $G_Y(\cdot)$ will be regarded as a member of the location-scale family of distributions, a general assumption made for computational reasons that will be described next.

In order to determine an intermediate failure threshold for level 1, the random variables Y_j must be ranked according to the information in y_j. In P-SuS, this is done using stochastic ordering. The topic is central to many important problems and has therefore been studied extensively, giving rise to various methods for ranking univariate random quantities [18]. One option is to compute a distribution for the *k-order statistic* of the outputs [19]. This can be challenging because the random variables Y_j are dependent and non-identically distributed [20]. Moreover, the process requires a large computational effort due to the combinatorial growth of the number of possible rankings.

Another option to rank the random variables is to use the concept of first- and second-order stochastic dominance [21]. This approach provides a partial order of the random variables, based on their CDFs. Let A and B be two random variables with corresponding CDFs, $G_A(z, \cdot)$ and $G_B(z, \cdot)$. By definition, A (first-order) stochastically dominates B if $G_A(z, \cdot) \leq G_B(z, \cdot)$ for all values of z. In situations where the CDFs of A and B intersect, first-order stochastic dominance is unable to provide a clear ordering. It is then useful to employ second-order stochastic dominance, where A dominates B if $\int_{-\infty}^{z} G_B(t, \cdot) - G_A(t, \cdot) dt \geq 0, \forall z$. Second-order stochastic dominance implies first-order stochastic dominance. In the context of P-SuS, the partial ranking, provided by stochastic dominance needs to be translated to a total order, to allow the establishment of the first intermediate failure threshold. One way to achieve this is by using Copeland counting [22]. This method counts how many times each of the N random variables has been identified as dominant in the pairwise comparisons, awarding it one point. At the same time, it deducts one point when the random variable has been flagged as being dominated. This results in a final score for each variable, with ties occurring only when two variables are identical. The ranking procedure requires $\mathcal{O}(n^2)$ operations and is summarised in Algorithm 1.

Computationally, second-order stochastic dominance is more expensive than its first-order counterpart, due to the integral operation. Additionally, both methods require the repeated evaluation of $G_Y(z, y_j)$ at each input location. These two

Algorithm 1 Total stochastic ordering.

```
Require: A set of N distribution parameter tuples $\{y_j\}$.
Ensure: A set of dominance scores, $\boldsymbol{s}$ for N random variables.
 1: Assign values to the ranking parameters $c$ and $N_q$.
 2: Initialise $\boldsymbol{s} \equiv \{s_1, \dots, s_N\} \leftarrow \{0, \dots, 0\}$
 3: for $j = 1$ to $N$ do                                  ▷ Copeland counting
 4:     for $k = j + 1$ to $N$ do
 5:         $r = \mathrm{mFOSD}(y_j, y_k, c)$                  ▷ Use Algorithm 2
 6:         if $r = 0$ then
 7:             $r = \mathrm{SOSD}(y_j, y_k, N_q)$             ▷ Use Algorithm 3
 8:         end if
 9:         $s_j = s_j + r$
10:         $s_k = s_k - r$
11:     end for
12: end for
```

factors can have an appreciable effect on running time when several hundred to several thousand random variables must be compared. Thus, reducing the computational cost on a per-comparison basis is essential to achieving ranking within reasonable time.

A metric is required to quickly identify pairs of responses with clear separation and dominance. That is, if two distributions are far enough apart, there is no need to compare them using an expensive test which allows for crossovers. For the location-scale family of distributions, one can devise a test which does not rely on evaluating $G_Y(z, y_j)$. In this case, quantiles that lie at an equal distance from the respective distribution mean will have the same cumulative probability under that distribution. For example, let $A \sim \mathcal{U}(a_A, b_A)$ and $B \sim \mathcal{U}(a_B, b_B)$, where $a_A < b_A$ and $a_B < b_B$, then

$$\mathbb{P}(A \le m_A + cs_A) = \mathbb{P}(B \le m_B + cs_B)$$
$$m_X = \frac{a+b}{2}$$
$$s_X = \sqrt{\frac{(b-a)^2}{12}}$$

where $X = \{A, B\}$ and $c \in \mathbb{R}$. If two values of c are chosen, such that the corresponding quantiles are far apart, then subtracting those quantiles and observing the signs of their differences provides an efficient way of determining the dominance of one distribution over another. Let $a_A = 0.2$, $b_A = 5$, $a_B = -5.7$, $b_B = 1$, and $c = \pm 1.65$. The corresponding quantiles, $q_A = \{0.314, 4.886\}$ and $q_B = \{-5.541, 0.841\}$ enclose approximately 95% of the mass of their respective distributions. The signs of both elements of the quantile difference, $q_A - q_B = \{5.855, 4.045\}$ are positive, indicating that A dominates B in the first-order sense.

Note that no evaluation of CDFs is involved in the comparison. This offers speed-up over first-order stochastic dominance. This modified first-order stochastic dominance (mFOSD) is summarised in Algorithm 2. In the case where the two signs

Algorithm 2 Modified first-order stochastic dominance (mFOSD).

```
 1: function MFOSD($y_j$, $y_k$, $c$)
 2:     Compute $\mu_j$, $\mu_k$, $\sigma_j$ and $\sigma_k$ from $y_j$ and $y_k$.
 3:     $\mu_d = \mu_k - \mu_j$
 4:     $\sigma_d = c(\sigma_k - \sigma_j)$
 5:     if $\mu_d < 0$ and $(\mu_d - \sigma_d) < 0$ then
 6:         return $d_1 = 1$                        ▷ $Y_j$ dominates $Y_k$
 7:     else if $\mu_d > 0$ and $(\mu_d - \sigma_d) > 0$ then
 8:         return $d_1 = -1$                       ▷ $Y_k$ dominates $Y_j$
 9:     else
10:         return $d_1 = 0$     ▷ Dominance cannot be established via mFOSD
11:     end if
12: end function
```

above differ, the comparison is carried out using the second-order dominance test, detailed in Algorithm 3. Two points about mFOSD must be made at this time. Firstly, the larger the value of c, the higher the confidence that the comparison is correct. However, c also controls the sensitivity of the test. Setting $c = \pm 10$ will provide a very high degree of confidence about the dominance result, but it means that mFOSD will be unable to decide between distributions that cross over at a very low probability level, which is unlikely to produce reversal of the ranking in practice. Secondly, as mentioned before,

Algorithm 3 Second-order stochastic dominance (SOSD).

```
1:  function SOSD($y_j$, $y_k$, $N_q$)                          ▷ Requires access to CDF for $Y_j$ and $Y_k$
2:      Discretise support of $Y_j$ and $Y_k$ into $N_q$ levels $\{z_r\}_{r=1}^{N_q}$
3:      for $r = 1$ to $N_q$ do
4:          Compute $\bar{G}_{j,r} \approx \int_{\infty}^{z_r} G_Y(z, y_j)dz$
5:          Compute $\bar{G}_{k,r} \approx \int_{\infty}^{z_r} G_Y(z, y_k)dz$      ▷ Via a quadrature scheme
6:      end for
7:      $\Delta\bar{G} = \sum_r \bar{G}_{j,r} - \bar{G}_{k,r}$
8:      if $\Delta\bar{G} < 0$ then
9:          return $d_2 = 1$                                    ▷ $Y_j$ dominates $Y_k$
10:     else
11:         return $d_2 = -1$                                   ▷ $Y_k$ dominates $Y_j$
12:     end if
13: end function
```

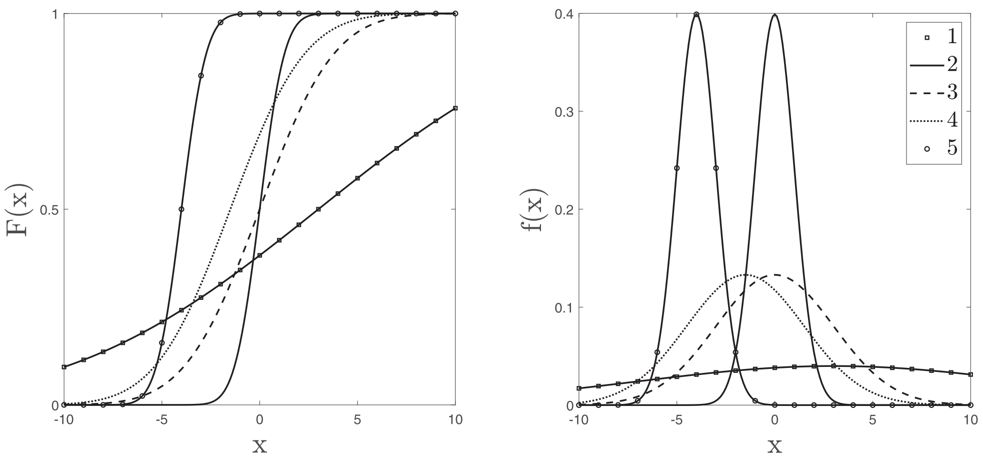


Fig. 1. Ranking of distributions using the augmented second-order stochastic dominance test with $c = \pm 3$. Cumulative distribution (left) and probability density (right) function perspective. Final ranking is given in the legend on the right.

this approach only works when ranking random variables which come from the location-scale family, with c determined according to the distributions being ranked. However, P-SuS is a general PRA method, so it can be used with any form of uncertain responses if one accepts the increased computational cost coming from using the full first-order stochastic dominance test.

It should also be noted that, as the fidelity of simulation results increases and the uncertainty from the PNM shrinks around its mean estimate, the separation between different Y_j becomes increasingly clear. The stochastic ordering used in P-SuS reflects this and is constructed in a way which provides results identical to those in deterministic SuS ranking, in the limit of vanishing uncertainty.

Figure 1 illustrates the ranking of 5 distributions with the augmented second-order procedure in Algorithm 1, for $c = \pm 3$. The comparisons between the distributions ranked first and fourth and between those ranked second and third were performed with the modified first-order dominance test. The rest of the comparisons required second-order testing.

3.2. Intermediate failure levels

The main purpose of P-SuS is to allow for the uncertainty in the output of the PN model to have an effect on the estimator of the failure probability, p_F. One possible way to do this is by adopting a bottom-up approach whereby the PNM uncertainty is reflected in the estimation of the conditional probabilities, p_0 , which in turn are used to construct the final, probabilistic estimator for p_F. Since each threshold, t_i is based on y_j, it would normally inherit the characteristics of these responses. In P-SuS, this means that t_i should have its own probability distribution. However, using the freedom in the choice of intermediate thresholds, t_i are given scalar values equal to the mean of the $(p_0 N)$ – th highest response, denoted $\mu^{(p_0 N)}$. Since $Y(\cdot)$ is a stochastic process, having a scalar t_i means that p_0 will be a random variable. For this reason, in P-SuS, the probability of F_i is renamed from p_0 to p_{F_i} to emphasise the fact that it will no longer be a constant as in deterministic SuS. Consequently, the distribution of p_{F_i} will be denoted $G_{F_i}(\cdot)$, and the parameter p_0 will be used in P-SuS to denote the nominal value of p_{F_i}

3.3. Probabilistic description of p_{F_i}

The possibility of treating the level probability as a random variable to enhance the results of SuS has been investigated before [15], albeit in a manner unrelated to the one proposed in P-SuS. There, the authors investigate the uncertainty present in the estimate of p_{F_i} due to the use of a finite sample to compute t_i, which may lead to a value of p_{F_i} different from p_0.

In the case of a probabilistic computer code, the uncertainty in p_{F_i} is of a different nature. For a fixed t_i, each input combination $\boldsymbol{x}_j$ belongs to $\mathcal{F}_i$ with some probability p_{ij}, given by

$$p_{ij} = \mathbb{P}\Big(h\big(\boldsymbol{x}_j\big) \geq t_i\Big) = 1 - G_Y\big(t_i, y_j\big) \text{ for } j = 1 \dots N_i \tag{10}$$

where N_i is the number of responses at the ith level. The probability of F_i is given by

$$\begin{aligned} p_{F_i} &= \int_{\mathcal{X}} \mathbb{I}_{F_i}(\boldsymbol{x}) g(\boldsymbol{x}, \cdot | F_{i-1}) d\boldsymbol{x} \\ &\approx \frac{1}{N_i} \sum_{j=1}^{N_i} \mathbb{I}_{F_i}(\boldsymbol{x}_j) \end{aligned}$$

with $\boldsymbol{x}_j \sim g(\boldsymbol{x}, \cdot | F_{i-1})$. In P-SuS, the indicator function is a Bernoulli random variable, such that

$$\mathbb{I}_{F_i}(\boldsymbol{x_j}) \sim \text{Bernoulli}(p_{ij}) \tag{11}$$

The summation of N_i independent Bernoulli trials with varying probabilities of success is distributed as a Poisson-binomial random variable, with mean $\mu_i = \sum_j p_{ij}$ and variance $\sigma_i^2 = \sum_j p_{ij}(1 - p_{ij})$.

However, since $\mathbb{I}_{F_i}(\cdot)$ come from the same underlying model, they will not, in general, be independent. In such cases, the distribution of the sum can be approximated by a Poisson distribution, with an approximation error bound determined by the strength of the correlations [23]. Moreover, the sum can also be bounded by a normal approximation [24]

$$\sum_j \mathbb{I}_{F_i}(\boldsymbol{x}_j) \mathrel{\dot\sim} \mathcal{N}(\mu_i, C\sigma_i^2) \tag{12}$$

where C is a positive constant. Defining the joint probability mass function of N_i dependent Bernoulli trials, requires the specification of probability values for each of the 2^{N_i} possible outcomes. Even for moderate sample sizes, such a task becomes computationally infeasible. At the same time, momentarily neglecting the dependence among $\mathbb{I}_{F_i}(\boldsymbol{x}_j)$ and $\mathbb{I}_{F_i}(\boldsymbol{x}_k)$ $\forall j \neq k$, allows the construction of a bounding distribution through the use of Lyapunov's central limit theorem, which states that the sum of the indicator functions is distributed $\mathcal{N}(\mu_i, \sigma_i^2)$. Comparing the two results, it becomes apparent that regardless of dependence, the distribution of the sum of non-identically distributed Bernoulli random variables can be approximated by a normal distribution, the dependence affecting only the approximation variance through the scaling constant, C.

It can be seen then that $G_{F_i}(\cdot)$ is Gaussian, such that

$$p_{F_i} \mathrel{\dot\sim} \mathcal{N}\left(\frac{\mu_i}{N_i}, \frac{C\sigma_i^2}{N_i^2}\right) \tag{13}$$

where C can be specified if information on the dependence of $\mathbb{I}_{F_i}(\boldsymbol{x}_j)$ is available [25]. This issue will be the subject of future work. In this paper, $C = 1$. It should be noted, that in the limit of vanishing uncertainty about Y_j, the variance of the bounding distribution, G_{F_i} approaches 0, while its mean approaches the constant $p_0 N$, coinciding with the deterministic case.

3.4. Seed selection and conditional sampling

As discussed in Section 2.1, to populate the ith intermediate failure domain $\mathcal{F}_i$, SuS uses MCMC sampling. At each level, N_C samples are selected as seeds to initialise the Markov chains. Since the number of seeds is based on the total number of samples per level and the probability of that level, in P-SuS N_C will be a random variable, here denoted as N_{C_i}.

In order to determine which input combinations will be selected as seeds, the algorithm must acknowledge the computational uncertainty present in the responses of h. Given the definitions established in Section 3.3, this can be done in two steps. In the first step, to respect the probabilistic nature of the PNM output, N'_{C_i} out of N_i input combinations are selected with probability p_{ij}, given in Eq. (10). It can be seen that the distribution of N_{C_i} is given in Eq. (12). To respect this counting distribution, in the second step, the first $\mu_i = \sum_j p_{ij}$ out of the N'_{C_i} samples are chosen. This two-step procedure results in the selection of $N_{C_i} = \min(N'_{C_i}, \mu_i)$ seeds. Once the sample selection is complete, each point is taken as a seed of a Markov chain with

$$N_{S_i} = \left\lceil \frac{N - N_{C_i}}{N_{C_i}} \right\rceil \tag{14}$$

states, where $\lceil \cdot \rceil$ is the ceiling function. This ensures that there will be at least N samples per level, as required by the analyst. Inspecting Eq. (14) more closely, it can be seen that the number of states reflects the fact that the seeds are kept as part of the sample from $\mathcal{F}_i$, since there is no burn-in, similar to SuS [26]. To generate N_{S_i} states, each Markov chain is progressed in two stages:

1. A candidate input combination, $\tilde{\boldsymbol{x}}_j$, is proposed using the independent-component Metropolis algorithm (ICMA) [14].
2. The corresponding response is accepted with probability p_{ij} in Eq. (10).

Step 1 is the same as in deterministic SuS. This is because probabilistic numerical models attach uncertainty to the output only, leaving information about the input unaffected. Step 2 is carried out as follows. For each candidate sample the probability p_{ij} in Eq. (10) is computed and compared to a random realization of a standard uniform distribution, denoted u_j. If $p_{ij} > u_j$ the corresponding candidate input combination, $\tilde{\boldsymbol{x}}_j$ is accepted. This procedure results in $\tilde{\boldsymbol{x}}_j$ being accepted $p_{ij} \times 100\%$ of the time, as the number of repetitions goes to infinity, thus allowing for samples that would otherwise be considered far away from $\mathcal{F}$ to be analysed. In line with this probabilistic acceptance policy, some input combinations with higher p_{ij}[1] may be rejected, reflecting the fact that under the uncertainty provided by the PNM, these samples may, actually, be outside of $\mathcal{F}$. The final probability of acceptance thus becomes

$$p_{acc} = p_{icma} \times p_{ij} \tag{15}$$

where p_{icma} denotes the acceptance probability in the independent-component Metropolis algorithm. The value of p_{icma} can be monitored to ensure it is between 30% and 70%, to allow the stable evolution of the Markov chains [15].

As more information about h becomes available, the uncertainty of the PN model will decrease, indicating the increase in confidence about the mean estimate. This reduction in posterior variance is manifested in the sampling step of P-SuS in two ways. Firstly, the number of samples N_i generated in each level will stabilise and begin approaching N. The same thing holds for, N_{C_i} and N_{S_i}, which will also converge to their deterministic counterparts in SuS. Secondly, the acceptance probability in Eq. (15) will become dominated by p_{icma}, as p_{ij} transforms from a continuous variable on [0,1] to a dichotomous one - $p_{ij} \in \{0, 1\}$ as in SuS. Therefore, in the limit of vanishing uncertainty about h, the sampling step in P-SuS will be identical to that in SuS. The sampling procedure is summarized in Algorithm 4.

Algorithm 4 Seed selection and conditional sampling for P-SuS.

```
Require: A vector of probabilities of success, $p_{ij}$ for $N_i$ Bernoulli random variables, $X_1, \dots, X_{N_i}$ and a threshold $t_i$.
Ensure: A sample from $g(\boldsymbol{x}, \cdot|F_i)$ and moments of the $i^{\text{th}}$ level counting distribution.
 1: Compute $\mu_i = \sum_j p_{ij}$                                    ▷ Mean of counting distribution
 2: Compute $\sigma_i^2 = \sum_j p_{ij}(1 - p_{ij})$                   ▷ Variance of counting distribution
 3: Draw $N_i$ random numbers $u \sim \mathcal{U}(0, 1)$
 4: Select $N'_{C_i}$ points with $p_{ij} > u$
 5: Take the first $\mu_i$ out of $N'_{C_i}$ samples and denote the set $N_{C_i}$
 6: Compute $N_{S_i}$ in Equation (14)
 7: for $j = 1$ to $N_{S_i}$ do                                        ▷ Conditional sampling
 8:     Propose $N_{C_i}$ candidate input combinations $\tilde{\boldsymbol{x}}_j$ via ICMA [14]
 9:     Evaluate $h(\tilde{\boldsymbol{x}}_j)$ to get $\tilde{y}_j$
10:     Compute $\tilde{p}_{ij} = 1 - G_Y(t_i, \tilde{y}_j)$
11:     Draw $N_{C_i}$ random numbers $u \sim \mathcal{U}(0, 1)$
12:     Accept those $\tilde{\boldsymbol{x}}_j$ for which $\tilde{p}_{ij} > u$
13: end for
```

3.5. Stopping condition

At any given level of P-SuS, each sample $\boldsymbol{x}_j$ belongs to $\mathcal{F}$ with probability given by

$$p_j^* = \mathbb{P}\big(h(\mathbf{x}_j) \geq t^*\big) = 1 - G_{Y_j}\big(t^*, y_j\big) \text{ for } j = 1 \dots N_i \tag{16}$$

Based on Eq. (16), a number of failure points, N_F, can be identified in a manner similar to the one used in the selection of N'_{C_i}, described in Section 3.4. In this way, the selection of failure points takes into account the uncertainty present in the output of h.[2]Once $N_F \geq p_0 N$, P-SuS is deemed to have populated $\mathcal{F}$ well enough to stop the generation of new conditional

[1] For example, in the case of a symmetric uncertainty model samples with $p_{ij} > 0.5$ belong to $\mathcal{F}$ according to their mean estimate. However, they can still be rejected as the spread of the associated uncertainty may indicate a non-negligible probability for the converged response to lie outside of $\mathcal{F}$.

[2] To completely acknowledge the uncertainty in the algorithm, it can be recognised that N_F is also a random variable with its own probability distribution, which will determine the number of intermediate levels in P-SuS. This point is mentioned as a potential direction for future work in Section 5.

levels. The point value of the probability of failure after m conditional levels is then estimated as

$$\mathbb{P}(F) \approx \hat{p}_F^{P-SuS} = \frac{N_F}{N_m} \prod_{i=1}^{m} \frac{N_{C_i}}{N_{i-1}} \tag{17}$$

where $N_0 = N$ is the number of direct Monte Carlo samples at the unconditional level. The quotient in the product operator of Eq. (17) is a point estimator of p_{F_i}. Based on the discussion in Section 3.3, there is uncertainty associated with p_{F_i} which can be propagated to $\hat{p}_F^{P-SuS}$. In order to get from the individual $G_{F_i}(\cdot)$ to a distribution $G_F(\cdot)$ for $\hat{p}_F^{P-SuS}$, one needs to compute the distribution of the product of m normal random variables. The product distribution problem is an active area of research and has been so for decades. There are a number of results for the product of two correlated normal random variables [27], as well as for the product of m independent, zero-mean normal random variables [28].

By using the product operator to compute probabilities, Eq. (17) implicitly assumes independence among the m levels. This assumption is also at the heart of deterministic SuS, allowing it to express the final estimator for p_F in the form of Eq. (3), and to obtain some of its other properties [15,29]. Under the independence assumption, the expected value of the product distribution $G_F(\cdot)$ for $\hat{p}_F^{P-SuS}$ can be computed as

$$\mathbb{E}\left[\hat{p}_F^{P-SuS}\right] = \prod_{i=1}^{m} \mathbb{E}[p_{F_i}] \tag{18}$$

where $\mathbb{E}$ is the expected value operator. Due to the seed selection procedure, outlined in Section 3.4, $\mathbb{E}\left[\hat{p}_F^{P-SuS}\right]$ will be slightly higher than the point estimator in Eq. (17). This artefact depends on the amount of uncertainty in the output of h and disappears for high fidelity responses. The variance of $G_F(\cdot)$ is given by the recurrence relation

$$\begin{aligned}\mathbb{V}\left[\hat{p}_F^{P-SuS}\right] &= \mathbb{V}\left[\prod_{i=1}^{m} p_{F_i}\right] \\ &= \mathbb{V}[p_{F_m}]\mathbb{V}\left[\prod_{i=1}^{m-1} p_{F_i}\right] + \mathbb{V}[p_{F_m}]\mathbb{E}\left[\prod_{i=1}^{m-1} p_{F_i}\right]^2 \\ &+ \mathbb{E}[p_{F_m}]^2\mathbb{V}\left[\prod_{i=1}^{m-1} p_{F_i}\right]\end{aligned} \tag{19}$$

where $\mathbb{V}$ is the variance operator. Due to the independence assumption, $G_F(\cdot)$ will be a good descriptor of the uncertainty in $\hat{p}_F^{P-SuS}$ whenever the correlation between levels, described in Au and Patelli [29] is not too large.

In the limit of vanishing uncertainty, it is straightforward to show, by inspection of constituent terms, that Eq. (17) becomes identical to Eq. (3) and $G_F(\cdot)$ degenerates around $\hat{p}_F^{P-SuS}$. The steps of P-SuS are summarised in Algorithm 5 and an implementation in MATLAB is available at https://github.com/PeterHristov/psus.

4. Numerical experiments

This section presents two examples: a smooth two-dimensional performance function, and a finite element model (FEM) of a wing box. The first example is a well-known deterministic function. For these experiments, uncertainty is added in a way in which sampling from one or more sub-domains of $\mathcal{F}$ is more challenging than with deterministic SuS. The performance function in the second example uses the Bayesian conjugate gradient method outlined in Section 2.2 to estimate wing tip displacement under static aerodynamic loading.

4.1. Two-dimensional performance function

Consider the modified Branin function [30] $h : [-5, 10] \times [0, 15] \to \mathbb{R}$:

$$h(x_1, x_2) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10(1-t)\cos(x_1) + 10 + 5x_1 \tag{20}$$

The use of the modified Branin function as a benchmark for algorithms in uncertainty quantification and Bayesian optimisation (e.g., to test the robustness of algorithms to getting stuck in local optima) is widespread [31,32]. In addition, the geometry of the Branin function makes it especially suitable as a benchmark for reliability analyses, due to the fact that complex failure domains can be generated easily. Thus, for instance, letting $t^* = 230$, the failure domain $\mathcal{F} = \{\boldsymbol{x} : h(\mathbf{x}) > t^*\}$, develops two disjoint sub-domains, such that $\mathcal{F} = \mathcal{F}^{(1)} \cup \mathcal{F}^{(2)}$, where $\mathcal{F}^{(1)} = \{x_1 \in [0, 0.037] \times x_2 \in [0, 0.108] : h(x_1, x_2) \geq t^*\}$ and $\mathcal{F}^{(2)} = \{x_1 \in [0.678, 0.866] \times x_2 \in [0.964, 1] : h(x_1, x_2) \geq t^*\}$. This feature is instrumental in showing that the proposed algorithm can uncover different regions of the failure domain, regardless of it being disconnected. The level sets of the Branin function are shown in Fig. 2, with $\mathcal{F}$ highlighted in red. By construction, the Branin function is deterministic. In order to mimic a probabilistic numerical model (PNM), the output is prescribed a location-scale distribution, $G_Y(\cdot)$, with

Algorithm 5 P-SuS.

```
Require: An input distribution, $G_X(\cdot)$, a probabilistic numerical computer model for $h()$, a critical system threshold, $t^*$, a
    nominal computational budget, $N$ and a nominal level probability, $p_0$.
Ensure: An uncertainty-aware estimate of the probability of failure of the system, $\hat{p}_F^{P-SuS}$, under $G_X(\cdot)$.
 1: Obtain $N$ data points $\boldsymbol{X} \sim G_X(\cdot)$ and responses $\boldsymbol{y} = h(\boldsymbol{X})$.
 2: Compute dominance scores, $\boldsymbol{s}$, given $\boldsymbol{y}$.                          ▷ Use Algorithm 1
 3: Sort $\boldsymbol{s}$ in descending order.
 4: Renumber $\boldsymbol{y}$ and $\boldsymbol{X}$ to match $\boldsymbol{s}$.
 5: Calculate $p_j^*$ for all samples in $\boldsymbol{X}$.                            ▷ In Eq. (16)
 6: Draw $N$ points $u \sim \mathcal{U}(0,1)$.
 7: Compute $N_F = \sum_{j=1}^{N} 1(p^* \geq u)$.
 8: Identify $\mathcal{F}$ through $\boldsymbol{X}_F = \{\boldsymbol{X} : p^* \geq u\}$.
 9: Set $i = 1$.
10: while $N_F \leq p_0 N$ do
11:     Identify $\mathcal{F}_i$ by computing $t_i = \mu^{(p_0 N)}$ from $\boldsymbol{y}$.
12:     Compute $p_{ij}$ for all samples in $\boldsymbol{X}$.                        ▷ In Eq. (10)
13:     Obtain $\boldsymbol{X} \sim g(\boldsymbol{x}, \cdot|F_i)$, $\boldsymbol{y} = h(\boldsymbol{X})$, $\mu_i$ and $\sigma_i$.        ▷ Use Algorithm 4
14:     Repeat steps $2 - 8$.
15:     if $N_F \geq p_0 N$ then                    ▷ Recompute moments of counting distribution
16:         Compute $\mu_i = \sum_j p_j^*$
17:         Compute $\sigma_i^2 = \sum_j p_j^*(1 - p_j^*)$
18:     end if
19: end while
20: Compute $\hat{p}_F^{P-SuS}$, $\mathbb{E}\left[\hat{p}_F^{P-SuS}\right]$ and $\mathbb{V}\left[\hat{p}_F^{P-SuS}\right]$.          ▷ In Eqs. (17)–(19)
```

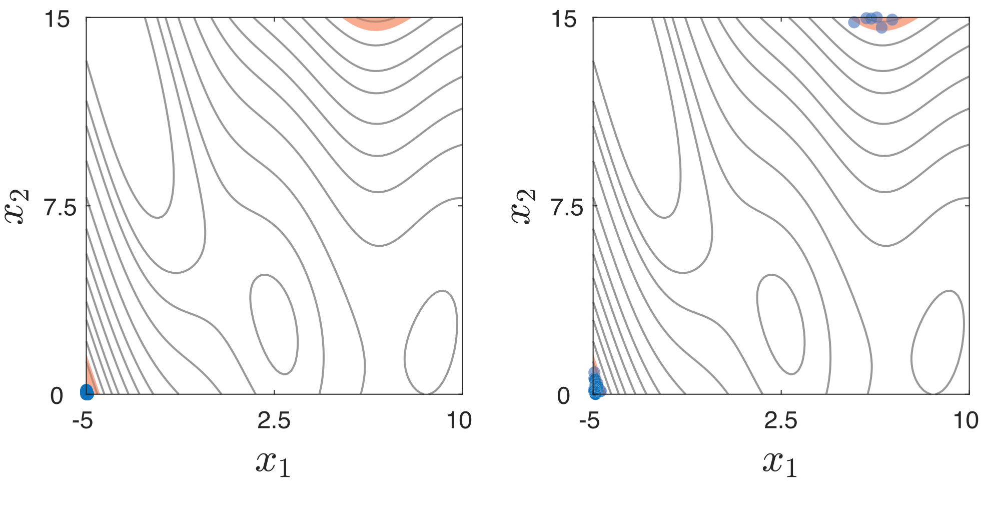


Fig. 2. Level contours and reliability analysis results for the modified Branin function. Red shaded regions highlight the failure domain, $\mathcal{F}$. a) deterministic SuS, which relies only on the mean estimate of the output to identify $\mathcal{F}$, samples (blue dots) only from $\mathcal{F}^{(1)}$ (red shading in bottom left); b) using the full probability distribution of the output $h(\boldsymbol{x})$, P-SuS is able to sample from both sub-domains of $\mathcal{F}$ (for interpretation of the references to colour in this figure, the reader is referred to the web version of the article).

mean, μ_j and variance, σ_j^2. Each evaluation of the model supplies information about the value of its output through the tuple $y_j = (\mu_j, \sigma_j^2)$. In this case study, the variance (scale parameter) of $G_Y(\cdot)$ can be constructed in an arbitrary manner, but is here given the following functional form

$$\sigma_j^2 = \sigma^2\left(\boldsymbol{x}_j\right) = e^{-\ell}\left|h\left(\boldsymbol{x}_j\right)\right| \tag{21}$$

where $\ell > 0$ is a parameter that controls the scale of the uncertainty in the model response. Since the uncertainty decreases as the value of ℓ increases, it can be seen as a proxy for the bound on the computational budget available to the analyst. Varying ℓ allows the performance of P-SuS at different levels of fidelity and its behaviour for $\sigma_j^2 \to 0$ to be evaluated.

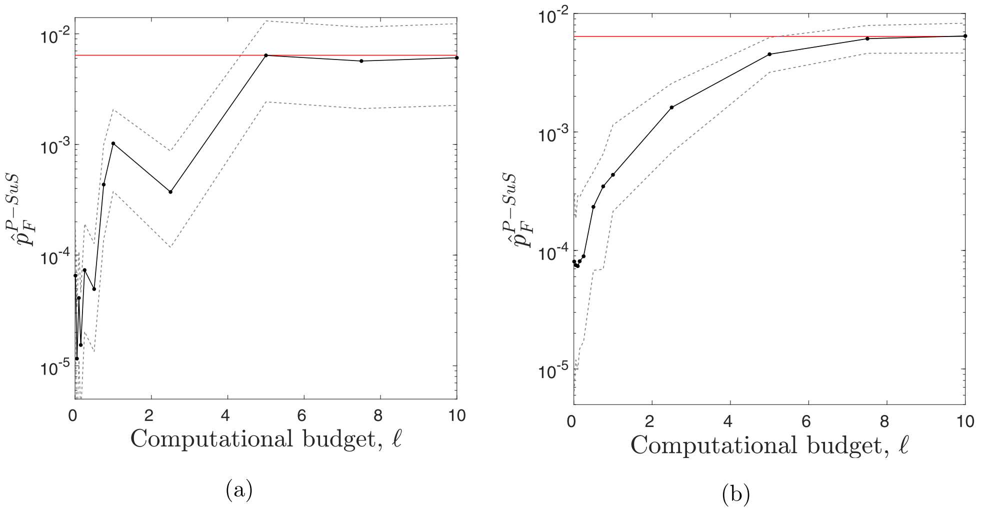


Fig. 3. Estimation of failure probability for varying values of the computational budget parameter, ℓ. (a) convergence history for a single run of P-SuS with median and ±3 standard deviations of $G_F(\cdot)$ for maximal values of C and (b) median (dot-line), 10th and 90th percentiles (dashed lines) of p_F from 100 runs of P-SuS.

It can be shown that for general PN methods, the mean of $G_Y(\cdot)$ converges to the solution of the associated (deterministic) numerical method [6]. To model the information-dependent deviation of the mean estimate from the truth, that estimate is here defined as

$$\mu_j = \mu(\boldsymbol{x}_j) = h(\mathbf{x_j}) - c\sigma_j \tag{22}$$

where c is a constant scaling factor and σ_j is determined via Eq. (21). Formulated in this way, the PNM proxy is effectively a deterministic model, i.e., despite the probabilistic information it provides, y_j will be precisely the same for repeated values of $\boldsymbol{x}_j$ and ℓ.

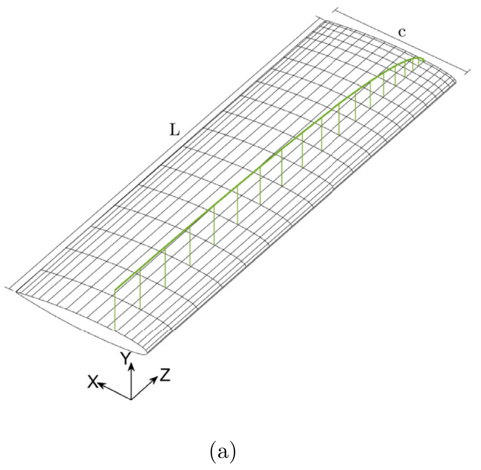
Running PRA with deterministic SuS limits the available information to that carried by $\mu(\boldsymbol{x})$. As such, any conclusions from the analysis will be based on partial observations and will likely be misguided. To demonstrate this, let $\ell = 0.01$ in order to emulate a situation in which the available computational budget is such that $\mu(\boldsymbol{x})$ is only a relatively low-fidelity approximation to $h(\mathbf{x})$. The shape of $\mathcal{F}$, according to $\mu(\boldsymbol{x})$ changes, compared to that based on $h(\mathbf{x})$ with $\mathcal{F}^{(1)}$ shrinking considerably and $\mathcal{F}^{(2)}$ disappearing altogether. Despite the fact that $\mathcal{F}^{(2)}$ still exists, deterministic SuS is unable to find it, as indicated by the lack of samples (blue dots) in the top right of Fig. 2(a).

In contrast, P-SuS detects the presence of $\mathcal{F}^{(2)}$, as seen in Fig. 2(b), by preserving samples $\boldsymbol{X}_{\mathcal{F}}$ which, given the uncertainty in the model output, may belong to $\mathcal{F}$. In this example, SuS was able to find the second failure mode for $\ell > 2.5$.

Another important outcome of the analysis is the estimated probability of failure, given in Eq. (17). The estimates of p_F given by P-SuS, for varying ℓ is shown in Fig. 3. The results in Fig. 3(a) show a single trace of $\hat{p}_F^{P-SuS}$, for a maximal value of C. In this case, the median of $G_F(\cdot)$ is plotted as a solid line, whereas the dashed lines form the 80% credible interval for $\hat{p}_F^{P-SuS}$. The results in Fig. 3(b) are based on 100 independent runs of P-SuS. The solid line shows the median value of the estimates, while the dashed lines on either side depict their 10th and the 90th percentiles. One thing to notice in particular is that despite the ability of P-SuS to identify both sub-domains of $\mathcal{F}$, $\hat{p}_F^{P-SuS}$ consistently underestimates p_F, shown in red in Fig. 3. While the underestimation of p_F in this particular case is due to the way the mean of the example function, in Eq. (22), was constructed, the P-SuS estimator will still produce results away from p_F when the output uncertainty is large. One possible reason for this is that $C = 1$ in Eqs. (12) and (13), which encodes an assumption of independence among the Bernoulli random variables in Eq. (11). Setting C to a value which results in G_{F_i} having the widest possible positive support, however, does not seem to address the issue completely, as seen in Fig. 3(a), where the true value of p_F is still far away from the credible region at small ℓ. Other assumptions which could be revised in an attempt to remedy this behaviour in $\hat{p}_F^{P-SuS}$ include the assumption of independence among levels and the form of the estimator in Eq. (17). These and other pointers for future work are discussed in Section 5.

4.2. Wing box model

The main purpose of this example is to demonstrate the capability of P-SuS to work with models which have features representative of industrial problems. Because the example is illustrative by nature, the failure criteria is defined so as to provide a challenging topology for the failure domain, not an engineering failure condition. In this way, the main advantage of the proposed method is demonstrated, namely its capability to discover regions of the input space of the numerical model that could be extremely difficult or impossible to discover otherwise, due to the numerical uncertainty obscuring them.



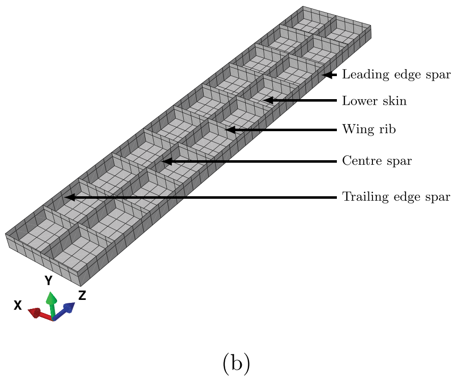


Fig. 4. Wing displacement model. (a) panel representation for aerodynamic force analysis. The lift distribution is shown in green. (b) cross section of the wing box. Finite element mesh of all main components is visible in dark grey. Elements corresponding to input variables in Table 1 are shown with black arrows. Upper skin is removed to expose inner wingbox structure (for interpretation of the references to colour in this figure, the reader is referred to the web version of the article).

Table 1
Inputs to the wing box finite element model. Thicknesses are given in meters (m) and elastic moduli are given in gigapascal (GPa).

Input			Distribution	
Part	Section	Subscript	Thickness, t	Young's modulus, E
Skin	Upper	US	$\mathcal{U}(0.006, 0.02)$	$\mathcal{U}(68, 88.5)$
	Lower	LS	$\mathcal{U}(0.006, 0.02)$	$\mathcal{U}(68, 88.5)$
Spars	Leading edge	SL	$\mathcal{U}(0.01, 0.1)$	$\mathcal{U}(68, 88.5)$
	Center	SC	$\mathcal{U}(0.01, 0.1)$	$\mathcal{U}(68, 88.5)$
	Trailing edge	ST	$\mathcal{U}(0.01, 0.1)$	$\mathcal{U}(68, 88.5)$
Ribs	Root to tip	$R_1, \ldots, R_{11}$	$\mathcal{U}(0.01, 0.05)$	$\mathcal{U}(68, 88.5)$

4.2.1. Problem description

In this section, P-SuS is applied to a wing box model, subject to aerodynamic lift forces. The test model is a uniform cantilever wing and is shown in Fig. 4. The wing has a semi-span (distance from root to tip) $L = 20$ m and a constant chord, $c = 6$ m. For the purposes of load calculation, the wing was given a symmetric, NACA 0012-64 airfoil and was analysed with an open-source potential flow panel solver [33,34]. The dynamic pressure and Reynolds number of the free stream were set to $q = 2.48$ kPa and $Re = 19.7 \times 10^6$, respectively. At an angle of attack of 2.5°, which corresponds to a shallow climb, the total lift force produced by the wing is $F_Y = 59.5$ kN with a lift distribution shown in green in Fig. 4(a). This distribution was used in defining the load on the wing box.

The structural model is inspired by the Goland research wing [35], whose main purpose is to provide a common and simplified test platform for research into aeroelastic problems. This model has also been used in uncertainty quantification for various aeroelastic phenomena [36,37]. The modified wing box used in this paper has length $L = 20$ m, width $w = 4$ m and height $v = 1$ m, instead of the dimensions and imperial units used in other studies. This modification was made to scale the wing to a size more representative of a single-aisle passenger aircraft. The wing box, shown in Fig. 4(b), is made up of lower and upper skins, three spars running along the length of the box and eleven ribs perpendicular to the spars. The components of the wing box are all basic rectangular shapes, allowing them to be accurately discretised into finite elements. The model was discretised in the commercial FEM suite Abaqus, using 1056 S4R shell elements [38]. In this representation, the structural response is parameterised by the thickness and stiffness of each element. The parameters were given a uniform joint distribution over their ranges to simulate the preference to any specific input combination. The variable parameters along with their distributions are summarised in Table 1. In this model, the stringers in the wing box were absorbed into the lower and upper skins [39]. For this reason, the upper bound on the distributions of the two skins is relatively large compared to v. The wing box is modelled as cantilever beam, i.e., having no translation and rotation at the root. In view of the illustrative nature of the problem, the quantity of interest for PRA is the maximum vertical displacement at the wing tip. The failure threshold is set as $t^* = 1.3$ m.

4.2.2. Probabilistic subset simulation setup

The discretisation described in Section 4.2.1 results in a linear system of the form $\boldsymbol{Ku} = \boldsymbol{f}$ where $\boldsymbol{K} \in \mathbb{R}^{d \times d}$ is called the stiffness matrix, $\boldsymbol{f} \in \mathbb{R}^d$ is referred to as the nodal force vector and $\boldsymbol{u} \in \mathbb{R}^d$ is the displacement vector, and $d = 5400$. The performance function can be written as

$$h(\mathbf{x}) = \max \boldsymbol{u}_{\text{y-tip}} = \max \left(\boldsymbol{K}^{-1}\boldsymbol{f}\right)|_{\text{y-tip}} \tag{23}$$

The subscript "y-tip" denotes the subset of vertical displacements at the tip. Thus, even though the solution vector, $\boldsymbol{u}$ contains the displacements of the entire wing box, the performance function $h(\mathbf{x})$ is only concerned with the maximum vertical displacement of the tip. This has the effect of reducing the dimensionality of the output from d to 1. The input vector $\boldsymbol{x}$ appears implicitly in $h(\mathbf{x})$, determining the values of the elements in $\boldsymbol{K}$. Despite the fact that this system can be solved using direct methods, it is used in this paper to test P-SuS on a problem which preserves all aspects of systems, for which an iterative solver would be preferred.

Matrices describing physical systems are often times ill-conditioned. This is of particular importance to BCG, due to the fact that its convergence is governed by $\kappa(\boldsymbol{K}\boldsymbol{\Sigma}_0\boldsymbol{K}^{\mathrm{T}}) \geq \kappa(\boldsymbol{K})$ [12], where $\kappa(\boldsymbol{K})$ is the conditioning number of $\boldsymbol{K}$, which is very large for ill-conditioned systems. In order to reduce the computational burden and increase the solution stability, the prior covariance matrix, $\boldsymbol{\Sigma}_0$, can be chosen such as to form a suitable preconditioner for the system. In this paper, we adopt the formulation $\boldsymbol{\Sigma}_0 = (\boldsymbol{P}^{\mathrm{T}}\boldsymbol{P})^{-1}$, where $\boldsymbol{P} = \boldsymbol{L}\boldsymbol{L}^{\mathrm{T}}$ and $\boldsymbol{L}$ is the lower triangular factor of the incomplete Cholesky (IC) decomposition of $\boldsymbol{K}$, denoted as IC(α), where the level of fill-in is determined by the parameter α. In the present example it was possible to recompute the IC(α) factorisation at every input combination. This was done in an attempt to identify an *optimal* α for each system. Naturally, for much larger problems, a different approach to preconditioning will have to be adopted. For example, it is possible to use a common $\boldsymbol{\Sigma}_0$ for systems arising from similar values of the physical parameters. Another approach may be to use the *posterior* covariance matrix, $\boldsymbol{\Sigma}_{\boldsymbol{m}}$ in Eq. (9), as the prior covariance matrix to a subsequent problem. In general, computing an effective preconditioner efficiently is a non-trivial task that often requires physical insight into the problem at hand. It is largely an area of ongoing research [40].

As discussed in Section 3.1, using PN models, such as BCG results in a set of tuples $\{y_j\}$, which contain parameter values for the output distribution, $G_Y(\cdot, y_j)$. Since the posterior distribution of the BCG solution vector is Gaussian, $y_j = (\mu_j, \sigma_j)$. In BCG, both μ_j and σ_j are functions of the number of BCG iterations n. For a well-conditioned system, the posterior mean will approach the true solution and the posterior variance will contract around the mean. The rate of this mean-variance evolution should be such that $h(\mathbf{x}_j)$ can be seen as a feasible realisation from the high density region of $G_Y(\cdot, y_j)$ at any n. In practice, this means that if μ_j is close to $h(\mathbf{x}_j)$, then this must be reflected by a small σ_j and vice-versa. The process of ensuring this is the case is referred to as *uncertainty calibration* [6]. It is claimed here, that in order to efficiently use the estimator of $h(\mathbf{x}_j)$ in any subsequent analysis (including PRA), the posterior uncertainty about Y_j must be well-calibrated.

In this paper, instead of fixing the number of iterations prior to the run of BCG, bounds on the minimum and maximum iterations are placed and the algorithm is run until the coefficient of variation (CoV), denoted δ_{Y_j}, falls below a predefined threshold, $\bar{\delta}_{Y_j}$. The convergence history of a typical run of BCG is shown in Fig. 5. At very small number of iterations, particularly $n < 10$, the output of BCG is erratic and it is difficult to draw a conclusion about the displacement of the structure. At $n \geq 15$ the output converges steadily to the solution of the system obtained with a direct solver. From $n \approx 150$, it becomes difficult to distinguish any further convergence. Based on this behaviour, BCG was run for $15 \leq n \leq 200$ iterations at each point in P-SuS. Another observation from Fig. 5 is that, despite the fact that μ_j converges relatively quickly to the deterministic solution, σ_j takes much longer to shrink around μ_j, resulting in a conservative estimator for $h(\mathbf{x}_j)$. This problem about BCG is well-known [12] and solutions are being developed [41].

To assess the performance of P-SuS at different levels of uncertainty, the analysis was carried out for maximum CoV $\bar{\delta}_{Y_j} = \{1, 0.75, 0.5, 0.25, 0.1, 0.05\}$. The first four levels were easily attained within the maximum number of iterations. At $\bar{\delta}_{Y_j} = \{0.1, 0.05\}$, for which BCG was not able to reach the desired fidelity for all points, the maximum number of iterations was increased to $n = 500$.

Each P-SuS run was set up with $N = 500$ samples per level and target level probability, $p_0 = 0.1$. The number of conditional levels in P-SuS depends strongly on the bound of the CoV. In the case of $\bar{\delta}_{Y_j} = \{1, 0.75\}$, P-SuS converged at the Monte Carlo level, whereas analyses with $\bar{\delta}_{Y_j} = 0.5, 0.25, 0.1$, converged after one conditional level, and that with $\bar{\delta}_{Y_j} = 0.05$ required two conditional levels. Using estimations from BCG runs that are allowed to terminate early, due to looser posterior variance requirements, P-SuS can count points far away from $\mathcal{F}$ as actually failing. The results from setting $\bar{\delta}_{Y_j} = 0.5$ are shown in the panels of the lower triangle of Fig. 6, where the panels depict two-dimensional pairwise projections of $\mathcal{F}$, as estimated by the scatter of failure samples. Points considered by P-SuS to have resulted in the occurrence of F are shown as gray dots. For comparison, samples from SuS, based on a deterministic solution to the linear system and having the same values for N and p_0 are plotted with red squares. It can be seen that the accuracy of the identification of $\mathcal{F}$ by P-SuS is determined by the fidelity of the supplied data. Only skin and spar thickness parameters are shown in Fig. 6, since their variation had the biggest effect on the occurrence of F. In general, for $\bar{\delta} = 0.5$, the size of $\mathcal{F}$ is overestimated and the projected domain is shifted to higher input values. These are both indicators of the conservative nature of P-SuS, due to the fact that the algorithm takes the effect of the uncertainty present in the available data into account. Despite this, P-SuS is capable of populating the failure domain reasonably well. This is particularly noticeable in the $t_{LS} - t_{US}$ projection,

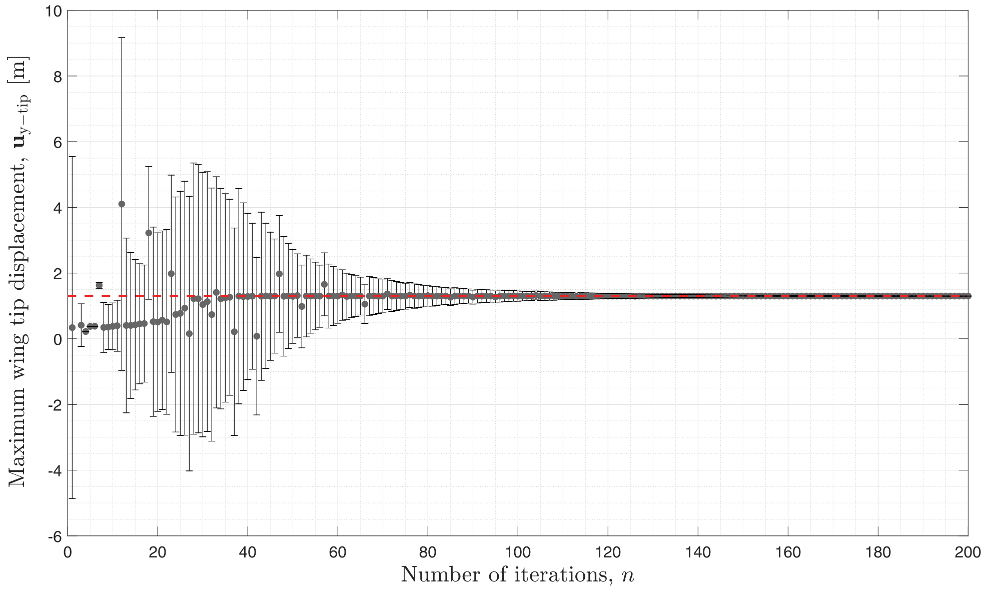


Fig. 5. Convergence of the Bayesian conjugate gradient for a single input combination. Deterministic solution is given as a red dashed line. Error bars show 95% credible interval of Y_j. Estimated responses for $n > 10$ contain the true value in their credible intervals, but exhibit conservative uncertainty (for interpretation of the references to colour in this figure, the reader is referred to the web version of the article).

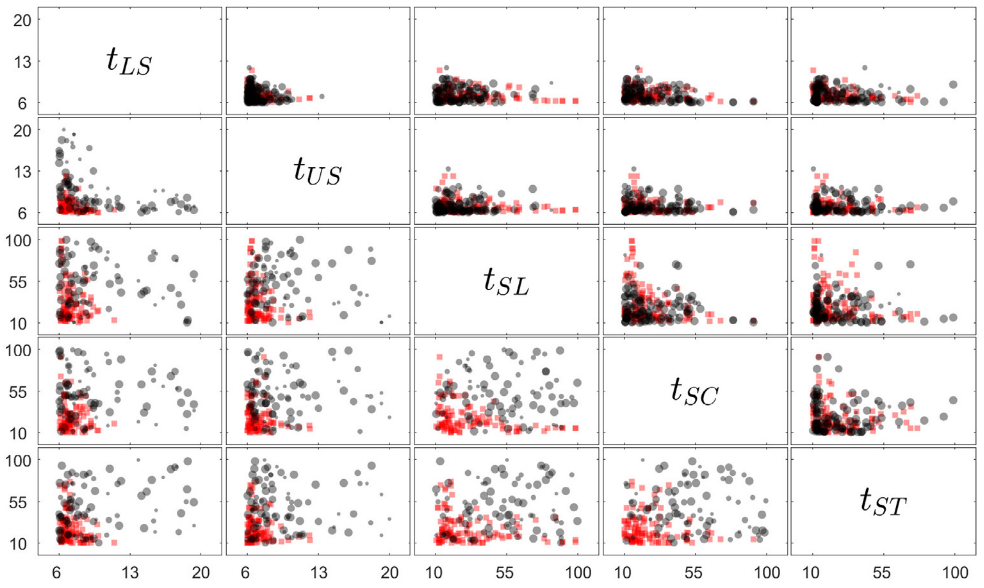


Fig. 6. Two-dimensional projections of samples in $\mathcal{F}$. Results from $\bar{\delta}_{Y_j} = 0.5$ and $\bar{\delta}_{Y_j} = 0.05$ in lower and upper triangle plots, respectively. Input combinations that are deemed by P-SuS to lead to tip displacements larger than t^* are shown as black circles. The size of the circles is proportional to p_j^*. Failure samples from SuS are plotted with red squares (for interpretation of the references to colour in this figure, the reader is referred to the web version of the article).

where a considerable part of $\mathcal{X}$ is correctly identified as safe, without having to invest computational resources to run BCG to convergence.

The upper triangle of Fig. 6 is set up similarly, excepts that the data shown in gray comes from a P-SuS run based on data with $\bar{\delta}_{Y_j} = 0.05$. A significant overlap between the P-SuS and SuS scatters is evident, as was anticipated by the discussion in Section 3, where it was argued that in the limit of vanishing uncertainty about the underlying data, the results from P-SuS become qualitatively identical to those of SuS. In both sets of plots, the relative size of the gray dots indicates the value of p_j^*. For $\bar{\delta}_{Y_j} = 0.5$ (lower triangle in Fig. 6), a considerable number of larger dots can be seen far from the red squares. Thus, this plot also reflects the quality of the uncertainty calibration of BCG. Because both, μ_j and σ_j evolve with each iteration

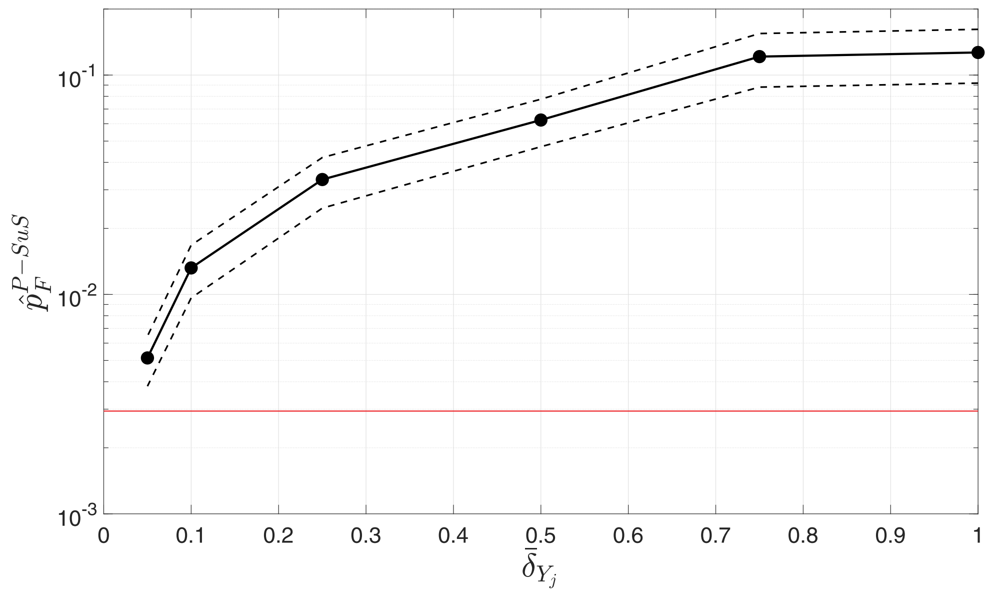


Fig. 7. Estimates of the probability of failure for varying levels of response fidelity for the static wing loading example. The mean (black solid line) and ± 3 standard deviations (grey dashed lines) of G_F steadily approach the $\hat{p}_F^{SuS}$ (red solid line) (for interpretation of the references to colour in this figure, the reader is referred to the web version of the article).

of the algorithm, $\boldsymbol{x}_j$ can be far away from $\mathcal{F}$, but if $\mu_j > t^*$ and σ_j sufficiently small, this may not be indicated as being in fact a false positive.

A depiction of $\hat{p}_F^{P-SuS}$ at each $\bar{\delta}_{Y_j}$ is shown in Fig. 7. The point values of the probability of failure for $\bar{\delta}_{Y_j} = 0.5$ and $\bar{\delta}_{Y_j} = 0.05$, corresponding to the failure samples shown in Fig. 6, are $\hat{p}_F^{P-SuS} = 0.0560$ and $\hat{p}_F^{P-SuS} = 0.0056$, respectively. However, since the output of BCG is uncertain, this uncertainty will be propagated through the conditional probabilities at each level, to the final estimate of the failure probability p_F^{P-SuS}, as outlined in Sections 3.3 and 3.5. In Fig. 7, this uncertainty is plotted as gray dashed lines depicting ± 3 standard deviations of $G_F(\cdot)$. It can be seen that $\hat{p}_F^{P-SuS}$ overestimates p_F at all CoV levels, in line with the conservative sampling in Fig. 6. This conservatism emerges through the probabilistic sample acceptance in the ICMA step and is responsible for the ability of P-SuS to discover hidden failure sub-domains. As the uncertainty about the true value of the output decreases (right to left in Fig. 7), the estimate of the probability of failure approaches the value estimated via SuS (red solid line in Fig. 7).

4.3. Extensions and further applications

In light of the above examples, it is worth noting that P-SuS can be used when the computational cost of the performance function is very large. This is not uncommon in industrial settings, where a statistical emulator is used to approximate the model output. In that case, PRA and statistical emulators can be combined [42] and P-SuS could be readily applied. Since P-SuS can work with any probabilistic output it could be used for different applications where SuS is already being used, such as optimisation (e.g., Li and Au [43]) and model calibration (e.g., Gong et al. [44]).

5. Conclusions

The analysis of models that incorporate a probabilistic description of computational uncertainty, known as probabilistic numerical models, has emerged in recent years as a research field called probabilistic numerics. This paper proposes an approach for conducting probabilistic reliability analysis (PRA) with such models. The method, called P-SuS, can be used for conducting probabilistic reliability analysis with any probabilistic numerical method in two principal directions:

1. P-SuS can be used for conducting reliability analyses in the setting in which the available computational budget and the evaluation time of the computer model are such that only results from partially-converged simulations are available.
2. Even if the computational budget is not strictly limited P-SuS can be used as a way of including in PRA the computational uncertainty present at any level of model fidelity.

The proposed method is based on the widely-used subset simulation algorithm for PRA. Each component of P-SuS was described in detail, outlining underlying assumptions and showing how P-SuS generalises SuS in the presence of computational uncertainty. The performance of the proposed approach was demonstrated with two numerical examples, including an industrially-representative finite element problem using a probabilistic numerical linear solver. In particular, the examples demonstrated the capability of P-SuS to reliably discover parts of the input space failure domain which may otherwise remain unexplored, and to inform the prioritisation of computational experiments at potentially important input locations.

Both probabilistic numerical methods and reliability analyses are extensive topics. As such, any method attempting to integrate them needs careful consideration and testing. The main aspect of P-SuS which requires further investigation is the estimator for the probability of failure, so as to enable its distribution to more accurately reflect the amount of uncertainty present in the model responses. Work is ongoing to develop improved uncertainty bounds by allowing computational uncertainty to have an effect on the number of conditional levels and by identifying more robust ways of incorporating (lack of knowledge about) dependence at different points in the algorithm. Another feature which is under development focuses on characterising the complete complementary cumulative distribution function (CCDF) of the model in the presence of computational uncertainty. Such a capability will be of particular importance to a fuller understanding of the model response.

Data availability

No data was used for the research described in the article.

Acknowledgements

This work was funded under the EPSRC grant EP/S001476/1. P. O. Hristov acknowledges the support of Innovate UK, under Research and Innovation Grant 46357 for the project “Development of advanced wing solutions” (DAWS). F. A. DiazDelaO acknowledges the support of the Data-centric engineering programme at The Alan Turing Institute, where he was a visiting fellow under the EPSRC grant EP/S001476/1.

References

[1] T. Bedford, R. Cooke, Probabilistic Risk Analysis: Foundations and Methods, Cambridge University Press, 2001.
[2] M.T. Pratola, O.A. Chkrebtii, Bayesian calibration of multistate stochastic simulators, Stat. Sin. 28 (2) (2018) 693–719.
[3] M.C. Kennedy, A. O’Hagan, Bayesian calibration of computer models, J. R. Stat. Soc. 63 (3) (2001) 425–464.
[4] W.L. Oberkampf, C.J. Roy, Verification and Validation in Scientific Computing, Cambridge University Press, 2010.
[5] B. Chartres, R. Stepleman, A general theory of convergence for numerical methods, SIAM J. Numer. Anal. 9 (3) (1972) 476–492.
[6] P. Hennig, M.A. Osborne, M.A. Girolami, Probabilistic numerics and uncertainty in computations, Proc. R. Soc. A 471 (2015) 20150142.
[7] Z. Ghahramani, C. Rasmussen, Bayesian Monte Carlo, Advances in Neural Information Processing Systems, vol. 15, 2002.
[8] F.-X. Briol, C.J. Oates, M.A. Girolami, M.A. Osborne, D. Sejdinovic, Probabilistic integration: a role in statistical computation? Stat. Sci. 34 (1) (2019) 1–22.
[9] M. Mahsereci, P. Hennig, Probabilistic line searches for stochastic optimization, in: Advances in Neural Information Processing Systems, vol. 28, 2015, pp. 181–189.
[10] O.A. Chkrebtii, D.A. Campbell, B. Calderhead, M.A. Girolami, Bayesian solution uncertainty quantification for differential equations, Bayesian Anal. 11 (4) (2016) 1239–1267.
[11] P. Hennig, Probabilistic interpretation of linear solvers, SIAM J. Optim. 25 (2015) 234–260.
[12] J. Cockayne, C.J. Oates, I.C.F. Ipsen, M.A. Girolami, A Bayesian conjugate gradient method (with discussion), Bayesian Anal. 14 (3) (2019) 937–1012.
[13] S.-K. Au, J.L. Beck, Estimation of small failure probabilities in high dimensions by subset simulation, Probab. Eng. Mech. 16 (4) (2001) 263–277.
[14] K.M. Zuev, Subset simulation method for rare event estimation: an introduction, 2015. arXiv:1505.03506.
[15] K.M. Zuev, J.L. Beck, S.-K. Au, L. Katafygiotis, Bayesian post-processor and other enhancements of subset simulation for estimating failure probabilities in high dimensions, Comput. Struct. 92–93 (2012) 283–296.
[16] J.R. Shewchuk, An Introduction to the Conjugate Gradient Method Without the Agonizing Pain, Technical Report, Carnegie Mellon University, USA, 1994.
[17] A. O’Hagan, Bayes–Hermite quadrature, J. Stat. Plan. Inference 29 (3) (1991) 245–260.
[18] F. Belzunce, C. Riquelme, J. Mulero, An Introduction to Stochastic Orders, Elsevier – Academic Press, 2015.
[19] H. David, H. Nagaraja, Order Statistics, John Wiley & Sons, 2004.
[20] R.B. Bapat, M.I. Beg, Order statistics for nonidentically distributed variables and permanents, Sankhyã: Indian J. Stat., Ser.A 51 (1) (1989) 79–93.
[21] A. Müller, D. Stoyan, Comparison Methods for Stochastic Models and Risks, John Wiley & Sons, 2002.
[22] N.B. Shah, M.J. Wainwright, Simple, robust and optimal ranking from pairwise comparisons, J. Mach. Learn. Res. 18 (199) (2018) 1–38.
[23] L.H.Y. Chen, Poisson approximation for dependent trials, Ann. Probab. 3 (3) (1975) 534–545.
[24] C. Stein, A bound for the error in the normal approximation to the distribution of a sum of dependent random variables, in: Proceedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability, Vol. II: Probability theory, Univ. California Press, Berkeley, Calif., 1972, pp. 583–602.
[25] W.L. Oberkampf, W.T. Tucker, J. Zhang, L. Ginzburg, D.J. Berleant, S. Ferson, J. Hajagos, R.B. Nelsen, Dependence in Probabilistic Modeling, Dempster–Shafer Theory, and Probability Bounds Analysis, Technical Report, Sandia National Laboratories, 2004.
[26] S.-K. Au, Y. Wang, Engineering Risk Assessment with Subset Simulation, John Wiley & Sons, 2014.
[27] S. Nadarajah, T.K. Pogny, On the distribution of the product of correlated normal random variables, C. R. Math. 354 (2) (2016) 201–204.
[28] M.D. Springer, W.E. Thompson, The distribution of products of beta, gamma and Gaussian random variables, SIAM J. Appl. Math. 18 (4) (1970) 721–737.
[29] S.-K. Au, E. Patelli, Rare event simulation in finite-infinite dimensional space, Reliab. Eng. Syst. Saf. 148 (2016) 67–77.
[30] A.I.J. Forrester, A. Sobester, A.J. Keane, Engineering Design via Surrogate Modelling : A Practical Guide, John Wiley & Sons, 2008.
[31] A. Garbuno-Inigo, F.A. DiazDelaO, K.M. Zuev, Gaussian process hyper-parameter estimation using parallel asymptotically independent Markov sampling, Comput. Stat. Data Anal. 103 (2016) 367–383.
[32] X. Yang, D. Barajas-Solano, G. Tartakovsky, A.M. Tartakovsky, Physics-informed CoKriging: A Gaussian-process-regression-based multifidelity method for data-model convergence, J. Comput. Phys. 395 (2019) 410–431.
[33] J. Anderson, Fundamentals of Aerodynamics, McGraw-Hill Education, 2016.
[34] A. Deperrois, xflr5, 2021. http://www.xflr5.tech/.
[35] P.S. Beran, N.S. Khot, F.E. Eastep, R.D. Snyder, J.V. Zweber, Numerical analysis of store-induced limit-cycle oscillation, J. Aircr. 41 (6) (2004) 1315–1326.
[36] S. Marques, K. Badcock, H. Khodaparast, J. Mottershead, CFD based aeroelastic stability predictions under the influence of structural variability, in: 50th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, 2009, pp. 2699–2721.

[37] M. Kurdi, N. Lindsley, P.S. Beran, Uncertainty quantification of the Goland+ Wing's flutter boundary, in: AIAA Atmospheric Flight Mechanics Conference and Exhibit, 2007, pp. 104–123.
[38] Anonymous, Shell elements, in: ABAQUS 6.14 Analysis User's Guide Volume IV: Elements, Dassault Systèmes, 2014, pp. 29.6.1–29.6.10.
[39] R.M. Ajaj, M.I. Friswell, D. Smith, A.T. Isikveren, A conceptual wing-box weight estimation model for transport aircraft, Aeronaut. J. 117 (1191) (2013) 533–551.
[40] M. Benzi, Preconditioning techniques for large linear systems: a survey, J. Comput. Phys. 182 (2) (2002) 418–477.
[41] T.W. Reid, I.C.F. Ipsen, J. Cockayne, C.J. Oates, A probabilistic numerical extension of the conjugate gradient method, 2020. arXiv:2008.03225v1.
[42] P.O. Hristov, F.A. DiazDelaO, U. Farooq, K.J. Kubiak, Adaptive Gaussian process emulators for efficient reliability analysis, Appl. Math. Model. 71 (2019) 138–151.
[43] H.-S. Li, S.-K. Au, Design optimization using subset simulation algorithm, Struct. Saf. 32 (6) (2010) 384–392.
[44] Z.T. Gong, F.A. DiazDelaO, P.O. Hristov, M. Beer, History matching with subset simulation, Int. J. Uncertain. Quantif. 11 (5) (2021) 19–38.