Adaptive Gaussian Process Emulators for Efficient Reliability Analysis

P.O. Hristov[a,*], F.A. DiazDelaO[a], U. Farooq[b], K.J. Kubiak[c]

[a]*Institute for Risk and Uncertainty, School of Engineering, University of Liverpool, Liverpool L69 7ZF, United Kingdom*
[b]*Parker Hannifin Manufacturing (UK) Ltd*
[c]*School of Computing and Engineering, University of Huddersfield*

Abstract

This paper presents an approximation method for performing efficient reliability analysis with complex computer models. The computational cost of industrial-scale models causes problems when performing sampling-based reliability analysis. This is due to the fact that the failure modes of the system typically occupy a small region of the performance space and thus require relatively large sample sizes to accurately estimate their characteristics. The sequential sampling method proposed in this article, combines Gaussian process-based optimisation and subset simulation. Gaussian process emulators construct a statistical approximation to the output of the original code, which is both affordable to use and has its own measure of predictive uncertainty. Subset simulation is used as an integral part of the algorithm to efficiently populate those regions of the surrogate which are likely to lead to the performance function exceeding a predefined critical threshold. The emulator itself is used to inform decisions about efficiently using the original code to augment its predictions. The iterative nature of the method ensures that an arbitrarily accurate approximation of the failure region is developed at a reasonable computational cost. The presented method is applied to an industrial model of a biodiesel filter.

Keywords: Reliability analysis, Gaussian process emulation, Subset simulation, Bayesian optimisation, Filter model.

1. Introduction

1.1. Problem statement

Reliability analysis, in the most general sense, is concerned with the calculation of a probability of failure, p_F via the identification of the failure domain, $F = \{x : \eta(x) > y^*\}$ for a *performance function* $\eta(\cdot)$. This function contains all the available information for a complex

*Corresponding author. Tel. +44(0)747 461 42 87
Email address: p.hristov@liv.ac.uk (P.O. Hristov)

engineering system (*e.g.* stresses, loads). The probability of failure can thus be computed, in principle, by solving

$$p_F = \int_F \pi(\mathbf{x}) d\mathbf{x} \tag{1}$$

where $\pi(x)$ is the joint probability distribution for x. The modeling of x as a random variable is justified given the complexity of modern, realistic engineering systems. Without loss of generality, the performance function $\eta(\cdot)$ is considered to be a mapping from its input space $\mathcal{X} \subset \mathbb{R}^d$, to a scalar on the real line, $y \in \mathbb{R}$. In typical engineering applications the explicit form of this mapping is not known and it is considered to be a *black-box*. Black-box models are ubiquitous in engineering in the form of *e.g.* finite element solvers, computational fluid dynamics tools, climate models among others. A common feature is their computational intensity. For a typical system, the failure region F is significantly smaller than the input domain and is considered a rare event. The most straightforward method to evaluate the multidimensional integral in Eq. (1) is direct Monte Carlo (DMC) sampling. The DMC estimator, $\hat{p}_F$ is unbiased and independent of the number of input dimensions, d, but requires a large number of samples to ensure that F is well populated to allow a reasonably accurate estimation of its properties. This drawback, coupled with a computationally expensive model renders a DMC-based reliability analysis infeasible. A method, which improves on DMC and is widely used in engineering reliability analysis is *subset simulation* (SuS) [1]. Despite the significant savings SuS brings over DMC, it can still be quite expensive when evaluating the simulator directly.

In order to address the computational cost of the simulator, one can build a less expensive approximation to the code output. Such an approximation is widely known as a *metamodel* or *emulator*. There exists a large number of metamodelling paradigms as outlined in *e.g.* Ch. 2 of [2] and Ch. 5 of [3]. A well-established approach is Gaussian process emulation (GPE), which builds a statistical approximation to the output of the code. Once the GPE is built, it provides a predictive distribution for the output of the code which can be used as an inexpensive substitute for the simulator, for the purposes of reliability analysis. One problem with using the surrogate directly is that it is typically *trained* on some space-filling sampling plan which aims at exploring the input domain of the simulator with as few points as possible. These sampling plans are based on Monte Carlo methods and usually select points in high probability regions. This is tantamount to saying that the surrogate that needs to be used to *predict and analyse rare events* is built on *frequent* events.

1.2. Literature review

A number of different frameworks have been proposed to reduce the cost of computations for reliability analysis. The literature survey presented here focuses on approaches which aim to improve the efficiency of reliability analysis via the use of a surrogate model. Methods using Gaussian process emulators are examined in more detail. Many methods rely on sequential sampling for the gradual improvement of the surrogate in the vicinity of the failure domain. This idea is heavily influenced by the field of surrogate-based optimisation (SBO), where an initial surrogate for a function is built from an experimental design and is used to search for optima. The interested reader is referred to [4–7] for more information.

A link between optimisation and reliability analysis is provided by Li and Au, [8], where parallels between the two are exploited. Some authors, such as Ranjan et *al.* [9] and Picheny et *al.* [10] suggest that the emulator should only be refined in regions of interest. An opposing view is given by Zhu and Du, [11] where the authors focus on the global accuracy of the surrogate. The approach proposed in the current article agrees with and extends the former idea.

Most methods for efficient reliability analysis and optimisation follow a framework composed of an initial sampling strategy, a rule for selecting new points and a stopping criterion. To that effect, different initial sampling strategies are presented in [12–16]. The approach proposed in this paper uses Latin hypercube sampling (LHS) by McKay et *al.* [17], which is widely used in surrogate modelling. The details of the initial sampling plan are discussed in Section 3.1. Many approaches to selecting improvement points based on some kind of a utility function, pre-generate a given number of candidate samples from the entire input space (e.g. [18]) and add the point that maximizes that function to the training plan for the next iteration of the algorithm. This could prove to be suboptimal because at the time of generation nothing about the limit state was known and thus there is no guarantee the information these samples provide about the failure domain will be adequate. Furthermore, the majority of methods select one point at a time, which, given the dynamical nature of sequential sampling could either miss important regions of the domain, or slow down convergence. Different strategies to selecting improvement points are presented in [10, 18–29].

The last major part of all adaptive algorithms is the stopping condition. This ranges from the use of reliability indices [20, 22] through error in the estimation of the failure probability [11, 18, 27, 28, 30] and forms of measure of the discrepancy between the emulator predictions and code observations [10, 19, 23, 29] to thresholds on the learning function [9, 26, 31]. Problems with some of these stopping conditions include the fact that they are based on relative error between iterations of the same algorithm, or that they depend on the form of the learning function.

The aim of this work is to provide an improved efficient algorithm for reliability analysis with computationally expensive computer codes. The method contributes:

- A new way to select candidate points, accounting for the presence of disjoint failure domains, while minimizing the number of samples selected from regions far away from the failure domain.

- A stopping condition which relies implicitly on the similarity between the surrogate and the model in the failure domain.

The paper is organised as follows: Section 2 provides an overview of subset simulation and Gaussian process emulation. Section 3 introduces the proposed method and provides a brief comparison between it and existing algorithms; Section 4 demonstrates the performance of the algorithm. Relevant conclusions are drawn in Section 5.

2. Methodology overview

This section provides an overview of the theory behind subset simulation (SuS) and Gaussian process emulation (GPE).

2.1. Subset simulation

A very important problem in engineering is the estimation of the probability of failure of a system, p_F, given in Eq. (1). In the context of numerical simulations failure can be defined as the scenario where a response variable (output) of the model, exceeds some threshold of acceptable system behaviour. The output, y is related to the input variables, $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^d$, via some mapping provided by the simulator $y = \eta(\mathbf{x})$. Thus, the failure domain, as defined in the beginning contains the values of $\mathbf{x}$ which cause the system response, y to exceed a critical value y^*

$$F = \{\mathbf{x} : \eta(\mathbf{x}) > y^*\} \tag{2}$$

Estimating p_F is associated with sampling from F. Usually, for a well-designed system the true value of p_F is very small, that is, F is a *rare event*. Also, a typical model has a high dimensional input space and often the failure domain of that space is disjoint, so sampling from it poses a significant challenge. SuS [1] expresses the failure event as contained in a nested sequence of less-rare events.

$$F \subset F_m \subset F_{m-1} \subset \ldots \subset F_1 \tag{3}$$

where F_1 is a relatively frequent event. Given that sequence, it can be shown that the probability of the rare event F can be expressed as a product of larger conditional probabilities:

$$p_F \equiv \mathbb{P}(F) = \mathbb{P}(F_1) \cdot \mathbb{P}(F_2|F_1) \cdot \ldots \cdot \mathbb{P}(F|F_m) \tag{4}$$

Beginning from the unconditional level F_0, the algorithm "probes" the input space $\mathcal{X}$ via direct Monte Carlo sampling. Then, based on the values of y by the simulator, it constructs the first intermediate failure threshold, $y_1^* < y^*$, defining an intermediate or relaxed failure domain, F_1. SuS then populates F_1 with N_s samples, using an MCMC algorithm. The generation and population of intermediate, equiprobable levels ($\mathbb{P}(F_i|F_{i-1}) = p_0$) continues until a predefined number of samples lie in the true failure domain F. At the end of the algorithm an estimate of the complementary CDF (CCDF) of the response function is generated as shown in Ch. 5 of [32].

2.2. Gaussian process emulation

Using complex scientific models to perform analyses relying on a large number of code evaluations may prove to be infeasible. When such models are used for reliability analysis, subset simulation (SuS) offers a way to significantly reduce the number of calls to the simulator, as compared to direct Monte Carlo (DMC). Despite this fact, SuS remains a time-consuming method to use directly with the code, since it utilises MCMC sampling.

To reduce the cost of the model, a Gaussian process emulator (GPE) is constructed for its output. Gaussian processes (GP) are a generalization of the Gaussian distribution from the

space of vectors to the space of functions. Gaussian processes are completely defined by their mean and covariance functions and possess many desirable properties, including tractability of computation. Gaussian process emulators are a class of non-parametric surrogate models, which places a Gaussian process prior over functions, expressing the belief that any finite collection of simulator outputs is jointly-distributed as a multivariate Gaussian see Sec 2.2 of [33].

In GP emulation, the following form for the model structure is widely used [34]

$$\eta(\mathbf{x}) = h(\mathbf{x})^T\boldsymbol{\beta} + Z(\mathbf{x}) \tag{5}$$

In Eq. (5) a regression term, $h(\mathbf{x})^T\boldsymbol{\beta}$ which encapsulates any prior knowledge about the simulator is added to the GP. The function $h(\cdot) \in \mathbb{R}^q$ can be any known expression of its inputs [35]. The GP, $Z(\mathbf{x})$ has zero mean and covariance $\sigma^2 c(\mathbf{x}, \mathbf{x}'; \boldsymbol{\psi})$, where σ^2 is a scale parameter and $\boldsymbol{\psi}$ is a vector of parameters specifying the behaviour of the correlation function $c(\cdot, \cdot; \boldsymbol{\psi})$. In contrast to $h(\cdot)$, $c(\cdot, \cdot; \boldsymbol{\psi})$ must be a positive semi-definite function (see Sec 2.2 of [33]). There is a wide choice of correlation functions, but in this paper the exponentiated square function is used

$$c(\mathbf{x}, \mathbf{x}'; \boldsymbol{\psi}) = \prod_{j=1}^{d} \exp\left(-\frac{|x_j - x'_j|^2}{2\psi_j}\right) \tag{6}$$

Let $\mathcal{D} = \{\mathbf{x}_i, y_i = \eta(\mathbf{x}_i)\}_{i=1}^n$ be a collection of inputs and corresponding simulator outputs. Then, conditioning Eq. (5) on $\mathcal{D}$ and using the properties of Gaussian processes results in a posterior distribution for the output of the computer code

$$\eta(\cdot)|\mathcal{D}, \boldsymbol{\beta}, \sigma^2, \boldsymbol{\psi} \sim \mathcal{GP}(M(\cdot), V(\cdot, \cdot)) \tag{7}$$

where $M(\cdot)$ and $V(\cdot, \cdot)$ are the mean and covariance functions of the GP, respectively. A Bayesian treatment of Eq. (7) will result in placing a prior on each of the parameters and deriving their posterior, conditioned on $\mathcal{D}$. Typically, the maximum a posteriori estimate is taken as an estimate for $\boldsymbol{\beta}$ and σ^2 as

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{H}^T\mathbf{C}^{-1}\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{C}^{-1}\mathbf{y} \tag{8}$$

$$\hat{\sigma^2} = \frac{\mathbf{y}^T\left(\mathbf{C}^{-1} - \mathbf{C}^{-1}\mathbf{H}\left(\mathbf{H}^T\mathbf{C}^{-1}\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{C}^{-1}\right)\mathbf{y}}{n - q - 2} \tag{9}$$

where $\mathbf{H} = [h(\mathbf{x}_1), \ldots, h(\mathbf{x}_n)]^T$ and

$$\mathbf{C} = \begin{bmatrix} c(\mathbf{x}_1, \mathbf{x}_1; \boldsymbol{\psi}) & \cdots & c(\mathbf{x}_1, \mathbf{x}_n; \boldsymbol{\psi}) \\ \vdots & \ddots & \\ c(\mathbf{x}_n, \mathbf{x}_1; \boldsymbol{\psi}) & \cdots & c(\mathbf{x}_n, \mathbf{x}_n; \boldsymbol{\psi}) \end{bmatrix} \tag{10}$$

Replacing the original parameters in Eq. (7) with their estimates in Eq. (8) and Eq. (9), yields the posterior predictive distribution of $\eta(\mathbf{x}^*)$ at any unobserved point $\mathbf{x}^*$, as

$$\eta(\mathbf{x}^*)|\mathcal{D}, \boldsymbol{\psi} \sim m(\mathbf{x}^*) + s(\mathbf{x}^*, \mathbf{x}^*)t_{n-q} \tag{11}$$

where t_{n-q} denotes the standard t-distribution with $n-q$ degrees of freedom, $m(\mathbf{x}^*)$ is the posterior predictive mean of the model

$$m(\mathbf{x}^*) = h(\mathbf{x}^*)^T\hat{\boldsymbol{\beta}} + \mathbf{t}(\mathbf{x}^*)^T\boldsymbol{C}^{-1}(\mathbf{y} - \mathbf{H}\hat{\boldsymbol{\beta}}) \tag{12}$$

and $s(\mathbf{x}^*, \mathbf{x}^*)$ is the square root of the posterior predictive variance

$$\begin{aligned} s^2(\mathbf{x}^*, \mathbf{x}^*) =& \hat{\sigma}^2 \Big[c(\mathbf{x}^*, \mathbf{x}^*; \hat{\boldsymbol{\psi}}) - \mathbf{t}(\mathbf{x}^*)^T\mathbf{C}^{-1}\mathbf{t}(\mathbf{x}^*) \\ &+ \left(h(\mathbf{x}^*)^T - \mathbf{t}(\mathbf{x}^*)^T\mathbf{C}^{-1}\mathbf{H}\right)\left(\mathbf{H}^T\mathbf{C}^{-1}\mathbf{H}\right)^{-1} \\ &\times \left(h(\mathbf{x}^*)^T - \mathbf{t}(\mathbf{x}^*)^T\mathbf{C}^{-1}\mathbf{H}\right)^T\Big] \end{aligned} \tag{13}$$

In Eq. (12) and Eq. (13), $\mathbf{t}(\mathbf{x}^*) \in \mathbb{R}^n$ is such that $\mathbf{t}(\mathbf{x}^*) = (c(\mathbf{x}^*, \mathbf{x}_1; \boldsymbol{\psi}), \ldots, c(\mathbf{x}^*, \mathbf{x}_n; \boldsymbol{\psi}))^T$.Finally, estimating $\boldsymbol{\psi}$ is usually done via maximizing the likelihood of the data with respect to these parameters as suggested in Sec. 2.4 of [2].

The process of going from Eq. (7) to Eq. (11) is referred to as training of the emulator [36]. It entails the repeated inversion of $\mathbf{C}$ to optimize over $\boldsymbol{\psi}$. It is a well-known fact that matrix inversion scales with the number of training points as $\mathcal{O}(n^3)$. However, once constructed, the emulator can be evaluated efficiently as neither Eq. (12), nor Eq. (13) involves sampling the simulator. Furthermore, every call to the GPE provides a prediction and an associated uncertainty measure, which is central to the method proposed in the next section.

3. Proposed approach

This section introduces the proposed method for efficient reliability analysis. The method is a novel combination between Gaussian process emulation and subset simulation, and as such is termed *Gaussian process subset simulation* (GPSS). Substituting the simulator for the GPE significantly decreases the computational cost of reliability analysis. However directly replacing the simulator with the GPE may lead to important parts of the failure region being missed or to failing to find such a region at all. This section outlines the novel ways in which GPSS addresses these issues. A pseudocode is provided as Algorithm 1.

3.1. Initialization

In order to be reliably used, the emulator needs to be of sufficient quality around the estimated failure regions. This is not usually the case with GPE approximations built using data from high probability regions (according to the input PDF). For the physical model it is assumed that the critical failure threshold, y^* is known (given a priori) and sensible (i.e. there is a set of values of $\mathbf{x}$ for which $\eta(\mathbf{x}) > y^*$). The proposed algorithm starts by building a GPE based on a set of data points, selected according to a space-filling strategy, LHS in

this case. The initial number of samples in the present work is chosen to be $10d$, where d is the number of input dimensions, after [37]. The GPE can be validated to check if there are any large discrepancies between the emulator and simulator as in [35]. This is done to ensure that the initial approximation is of a reasonable overall quality.
It is likely that the original threshold is not attained on the first iteration since the GPE learns about the simulator output from samples that do not constitute a rare event. This is to say

$$F_1 = \{\mathbf{x} : m_1(\mathbf{x}) > y^*\} = \varnothing \tag{14}$$

where F_ℓ is the failure domain according to the ℓ^{th} emulator ($\ell = 1$ above), with predictive mean $m_\ell(\cdot)$ given by Eq. (12). Since the emulator is an efficient approximation of the code, in theory one can search the input space exhaustively. However, the size of a Monte Carlo sample required to get a reliable estimate of the features of the failure domain is usually large, leading to an inefficient use of the GPE. To ensure high efficiency, it is proposed to use SuS to sample from the posterior predictive distribution of the emulator, which leads to a quick and reliable convergence, especially in high dimensions. As outlined in Section 2.1, SuS converges when a predefined number of data points lie in the failure domain. If it is indeed the case that $F_1 = \varnothing$, the algorithm will be unable to converge due to all candidate samples being rejected. To resolve this, GPSS provides an alternative "failure level", $\tilde{y}_1^*$ such that $\mathbb{P}\left(m_1(\mathbf{x}) > \tilde{y}_1^*\right) > 0$. This approach gives rise to an intermediate emulator "failure" domain with respect to $\tilde{y}_1^*$, denoted as $\tilde{F}_1$. The number of samples, N_s for the algorithm can be set high enough to ensure that even truly rare subregions of $\tilde{F}_1$ are populated. Note, that this is now associated with a minimal increase in computation time, since the GPE is evaluated instead of the true function.

3.2. Level generation

The above discussion raises the question of how to calculate $\tilde{y}_\ell^*$ such that SuS converges, whilst simultaneously exploring potentially interesting regions of the input space. This section provides a comparison between three distinct rules giving GPSS different behaviour. The rules are presented in Table 1 along with relevant observations. A plot of each level used for the Goldstein-Price function (presented in Section 4.1) is shown in Figure 1. Each level-generating rule is only a function of simulator observations which carry no uncertainty. In Table 1, y_i are the n training responses and $\mathbb{I}(y_i)$ is an indicator function returning 1 if $y_i > \tilde{y}_{(\ell-1)}^*$ and 0 otherwise. This is not to be confused with $\mathbb{I}_F$ in Algorithm 1, which is the failure indicator function. It is noted that in the absence of prior knowledge about the function, Rule 3 provides the most balanced performance to GPSS. This is because it ensures that the current threshold will lie above the previous one, whilst safeguarding against setting it too high and discarding important regions. Rule 3 is used for the remainder of this article. However, the analyst could give preference to either quicker convergence or longer exploration by selecting other rules.

3.3. Sample selection

Let $\mathbf{X}_{\tilde{F}_\ell} = \{\mathbf{X} \in \tilde{F}_\ell\}$ be the set of points, produced by SuS, whose predicted response lies above the current failure level $\tilde{y}_\ell^*$. Among all of these samples some additional points at

Table 1: Failure level generation rules.

Rule	Expression	Remarks
1	$\tilde{y}^*_\ell = \left.\frac{\max(y_i)+\min(y_i)}{2}\right\vert_\ell$	Longer exploration for highly non-linear outputs.
2	$\tilde{y}^*_1 = \frac{\sum_{i=1}^N y_i}{N}$ $\tilde{y}^*_\ell = \frac{1}{N+1}\left[\sum_{i=1}^N y_i \mathbb{I}(y_i) + \tilde{y}^*_{(\ell-1)}\right]$	Information from new samples is given more weight for faster convergence.
3	$\tilde{y}^*_1 = \frac{\sum_{i=1}^N y_i}{N}$ $\tilde{y}^*_\ell = \frac{1}{2}\left[\frac{\sum_{i=1}^N y_i \mathbb{I}(y_i)}{N} + \tilde{y}^*_{(\ell-1)}\right]$	Previous level is given more weight to control progress speed.

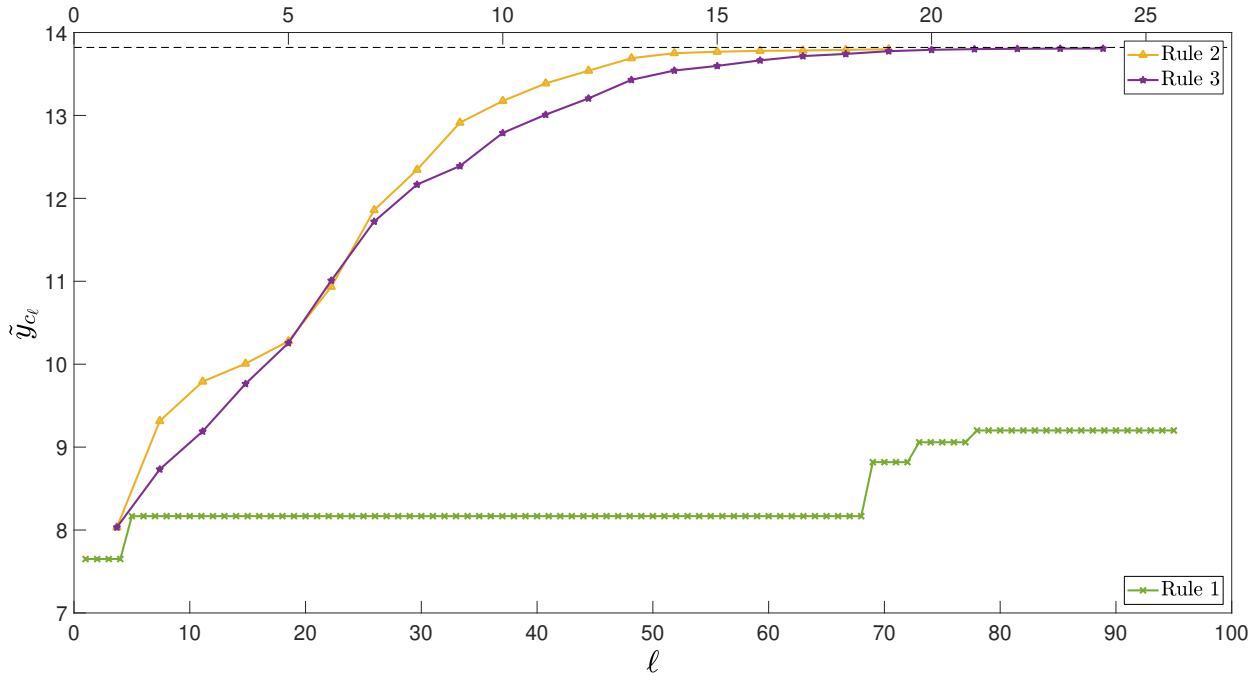

Figure 1: Comparison between alternative failure levels for the Goldstein-Price function.

which to run the model have to be selected before fitting the next emulator prediction. The posterior predictive variance of the emulator is proposed as an indicator of the local quality of the approximation, such that points whose predictive distribution indicates they are close enough to an existing training sample are not considered for repeated simulator evaluation. That is

$$\mathbf{X}_{cand} = \left\{ \mathbf{X}_{\tilde{F}_\ell} : s\left(\mathbf{X}_{\tilde{F}_\ell}, \mathbf{X}_{\tilde{F}_\ell}\right) \geq \varepsilon \right\} \tag{15}$$

where $\mathbf{X}_{cand}$ is the candidate population to be considered for code evaluation.

The constant ε in Eq. (15), is a small threshold on the standard deviation below which predictions from the GPE are considered accurate. This criterion is based on the use of a stationary correlation function, such as the one in Eq. (6), where the predictive variance depends only on the distance between test and training points. The value of ε is selected as follows. Let the value of y^* be specified as a number with k decimal places. Then, subtracting a number, $\mathcal{O}\left(10^{-k-1}\right)$ from y^* will not change it to the desired precision. Thus, predictions which lie in F according to $m_\ell(\cdot)$ will remain there according to their predictive distribution as long as $s(\cdot,\cdot) \leq \varepsilon$[1].

Let $\mathbf{X}_{add}$ denote the points to be added to the training sample for the next GPE, such that

$$\mathbf{X}_{add} = \underset{\mathbf{X}_{cand}}{\operatorname{argmax}}\, \mathbb{E}\big[I\left(\mathbf{X}_{cand}\right)\big]_\ell \tag{16}$$

where $\mathbb{E}\big[I(\cdot)\big]_\ell$ is the ℓ^{th} *expected improvement* function (EI) [38].

Expected improvement is a strategy that balances exploitation of the emulator mean and exploration of the design space and is given in functional form by

$$\mathbb{E}[I(\mathbf{x})]_\ell = \big| \left(y^* - m_\ell(\mathbf{x})\right) \big| \Phi\left(\frac{y^* - m_\ell(\mathbf{x})}{s_\ell(\mathbf{x},\mathbf{x})}\right) + s_\ell(\mathbf{x},\mathbf{x})\phi\left(\frac{y^* - m_\ell(\mathbf{x})}{s_\ell(\mathbf{x},\mathbf{x})}\right) \tag{17}$$

where the subscript ℓ denotes information regarding the ℓ^{th} emulator and the symbols $\Phi(\cdot)$ and $\phi(\cdot)$ denote the cumulative, and probability distribution functions of a standard Gaussian random variable, respectively.

Expected improvement can be run on each point in $\mathbf{X}_{cand}$ and the one that maximizes it will be added to the training plan for the next level GPE. Adding a single point or a fixed number of points to the training sample is widespread in the reliability literature (*e.g.* [18, 28, 30]). However, applying expected improvement directly to $\mathbf{X}_{cand}$ poses a risk of neglecting subregions in the intermediate emulator failure domain. Unless $\tilde{F}_\ell$ is not disjoint, the presence of separate modes needs to be accounted for. GPSS achieves this by first identifying the structure of $\tilde{F}_\ell$, detecting any modes and calculating EI on the samples in each mode. A clustering algorithm is used to discover the separate failure sub-domains. GPSS utilizes the density-based algorithm DBSCAN [39]. This algorithm does not need the number of clusters to be specified a priori. It is noted that clustering has been used for parallelisation [26] and as a sample strategy [20, 40] in related work, but not as a means of discovering multiple modes in the topology of $\tilde{F}_\ell$. Ultimately, the design plan for the

[1] See Section 4 for examples

emulator at the next level is composed of the current design plus the new points, formally $\mathcal{D}_{\ell+1} = \mathcal{D}_\ell \cup \{\mathbf{X}_{add}, \eta(\mathbf{X}_{add})\}$, where $\mathbf{X}_{add}$ is given in Eq. (16).

3.4. Stopping condition

In order to stop the iterative refinement of the emulator, the following rule is proposed. Consider running SuS with the true function. By design SuS will stop generating new levels once at least $N_s p_0$ samples from the last level lie in the failure domain, F. It follows that a necessary and sufficient criterion for the accuracy of the emulator in F is the ability of SuS to generate the same number of samples, denoted $\mathbf{X}_{F_\ell}$ whose predictive distribution indicates that they indeed belong to F_ℓ, that is

$$\begin{aligned} N_{F_\ell} &\geq N_s p_0 \\ N_{F_\ell} &= \sum \mathbb{I}_F \left(s\left(\mathbf{X}_{F_\ell}, \mathbf{X}_{F_\ell}\right) < \varepsilon \right) \end{aligned} \tag{18}$$

If this condition is satisfied, there will be no distinction between emulated and simulated samples.

It is important to note that the main aim of GPSS is to provide a way for the emulator to reliably discover the failure domain, with the use of minimal simulation resources. This is in contrast to increasing the efficiency of the GPE itself, which is not addressed here. Due to the importance of reliability analysis a number of methods exploiting Gaussian process emulators to reduce its cost have been developed in recent years. A short comparison is provided in Table A.1. Refer to Section 1 for an explanation of potential problems.

4. Numerical experiments

In this section, the performance of GPSS is demonstrated with three numerical problems. The function in Section 4.1 features a rare event for an efficiency test. The problem in Section 4.2 tests the ability of GPSS to deal with disjoint failure domains and Section 4.3 presents an example of GPSS being applied to an industrial-scale, expensive computer code.

4.1. Goldstein-Price function

The Goldstein-Price function is given by Eq. (19). The natural logarithm of the function is taken to reduce the range of the output values. The critical level, $y^* = 13.821$ was chosen to yield a small failure probability of $p_F = 5.04 \times 10^{-5}$. The corresponding $\varepsilon = 0.0005$, as $13.821 - 0.0005 = 13.8205$, which rounded to the original precision of y^*, still equals 13.821.

$$\begin{aligned} \eta(\mathbf{x}) = &[1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 + 14x_2 + 6x_1x_2 + 3x_2^2)] \\ &\times [30 + (2x_1 - 3x_2)^2 (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2)] \\ x_{1,2} &\in [0, 1] \end{aligned} \tag{19}$$

The initial GPE was trained with $n = 20$ LHS points with x_2 in $[0, 0.7]$. Figure 2 shows a selection of GPSS steps from $\ell = 1$ to completion. Each sub-figure shows the current

Algorithm 1 GPSS

Generate a space-filling plan, $\mathbf{X} \in \mathbb{R}^{n \times d}$ and form $\mathcal{D}_1 = \{\mathbf{X}_i, y_i = \eta(\mathbf{X}_i)\}$ for $i = 1 \dots n$
Train a GPE $\eta_1(\mathbf{x})|\mathcal{D}_1, \hat{\boldsymbol{\psi}} \sim m_1(\mathbf{x}) + s_1(\mathbf{x}, \mathbf{x})t_{n-q}$ and validate it.
Assign values to all parameters for GPSS - (p_0, N_s, y^*).
Calculate ε.
$\ell \leftarrow 1$
$N_{F_\ell} \leftarrow 0$
while $N_{F_\ell} < p_0 N_s$ **do**
    if $\ell = 1$ **then**
        $\tilde{y}^*_\ell = \frac{1}{n} \sum_{i=1}^{n} y_i$
    else
        $\tilde{y}^*_\ell = \frac{1}{2}\left[\frac{1}{n} \sum_{i=1}^{n} y_i \mathbb{I}(y_i) + \tilde{y}^*_{(\ell-1)}\right]$
    end if
    if $\tilde{y}^*_\ell > y^*$ **then**
        $\tilde{y}^*_\ell = y^*$
    end if
    $\tilde{F}_\ell = \{\mathbf{X}_i : m_\ell(\mathbf{X}_i) > \tilde{y}^*_\ell\}, \forall i$
    Populate $\tilde{F}_\ell$ using SuS $\rightarrow \mathbf{X}_{\tilde{F}_\ell}$
    Form a candidate sample $\mathbf{X}_{cand} = \left\{\mathbf{X}_{\tilde{F}_\ell} : s\left(\mathbf{X}_{\tilde{F}_\ell}, \mathbf{X}_{\tilde{F}_\ell}\right) \geq \varepsilon\right\}$
    Calculate $N_{F_\ell} = \sum \mathbb{I}_F\left(s\left(\mathbf{X}_{F_\ell}, \mathbf{X}_{F_\ell}\right) < \varepsilon\right)$
    Group $\mathbf{X}_{cand}$ in k clusters $\{\mathbf{X}^{(1)}_{cand}, \dots, \mathbf{X}^{(k)}_{cand}\}$
    $\mathbf{X}_{add} \leftarrow \varnothing$
    for j in $1 \dots k$ **do**
        $\mathbf{X}_{add} \leftarrow \mathbf{X}_{add} \cup \operatorname{argmax}_{\mathbf{X}^{(j)}_{cand}} \mathbb{E}\left[I(\mathbf{X}^{(j)}_{cand})\right]_\ell$
    end for
    $\mathcal{D}_{\ell+1} \leftarrow \mathcal{D}_\ell \cup \{\mathbf{X}_{add}, \eta(\mathbf{X}_{add})\}$
    $\ell = \ell + 1$
    Train a GPE $\eta_{\ell+1}(\mathbf{x})|\mathcal{D}_{\ell+1}, \hat{\boldsymbol{\psi}} \sim m_\ell(\mathbf{x}) + s_\ell(\mathbf{x}, \mathbf{x})t_{n-q}$
end while

intermediate threshold, $\tilde{y}^*_\ell$ in red and the boundary of the failure domain, $\tilde{F}_\ell$ in black. The region that the algorithm samples from at each iteration is shaded in grey. Notice that in all stages the sampling domain chosen by SuS covers the general region of the failure domain. In sub-figures $\ell = 6$ through $\ell = 9$ the algorithm samples in more than one location. This is due to the fact that in these particular cases the mean of the emulator, β was estimated such that predictions away from the training points had relatively high values. Once enough information was obtained in these regions they were discarded as seen in $\ell = 11$ onwards. Finally, the last sub-figure contains an inset showing a magnification of the sampling and failure regions. The estimated mean probability of failure, based on 1000 runs of SuS on the improved GPE, was $\bar{p}_F = 5.09 \times 10^{-5}$. The coefficient of variation and relative error based on the same sample were $\delta_{p_F} = 26.1\%$ and $\Delta p_F = 0.91\%$, respectively. A more complete idea about the quality of the approximation can be obtained from Figure 3, showing a comparison between the CCDF curves for SuS and GPSS. The results point to two important remarks. Firstly, the importance of supplying simulator information to the GPE, instead of relying on a one-iteration emulator is clearly demonstrated. Secondly, the existence of different level-generation rules provides means of benchmarking results obtained with a particular set-up.

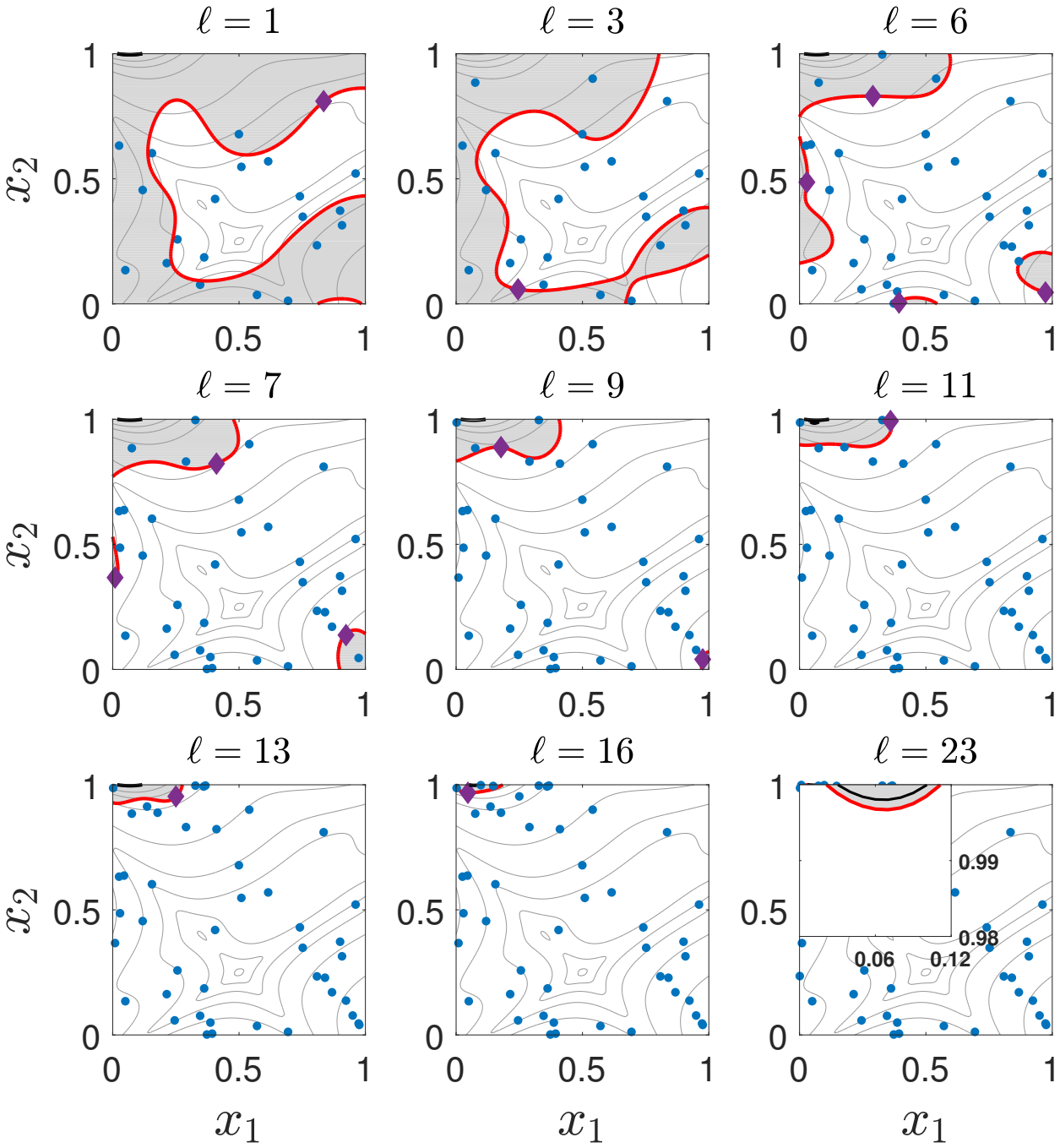

Figure 2: Progression of GPSS using Rule 3 for the modified Goldstein-Price function. The titles in each sub-figure denote the algorithm iteration. Red contours correspond to $\tilde{y}_\ell^*$ and black contours show $\tilde{F}_\ell$. New data points from the shaded regions are plotted as diamonds. *Inset in last tile:* Failure domain (black) and last GPSS level (red). The number of training samples in each tile is $n = \{20, 23, 30, 34, 39, 42, 44, 47, 54\}$.

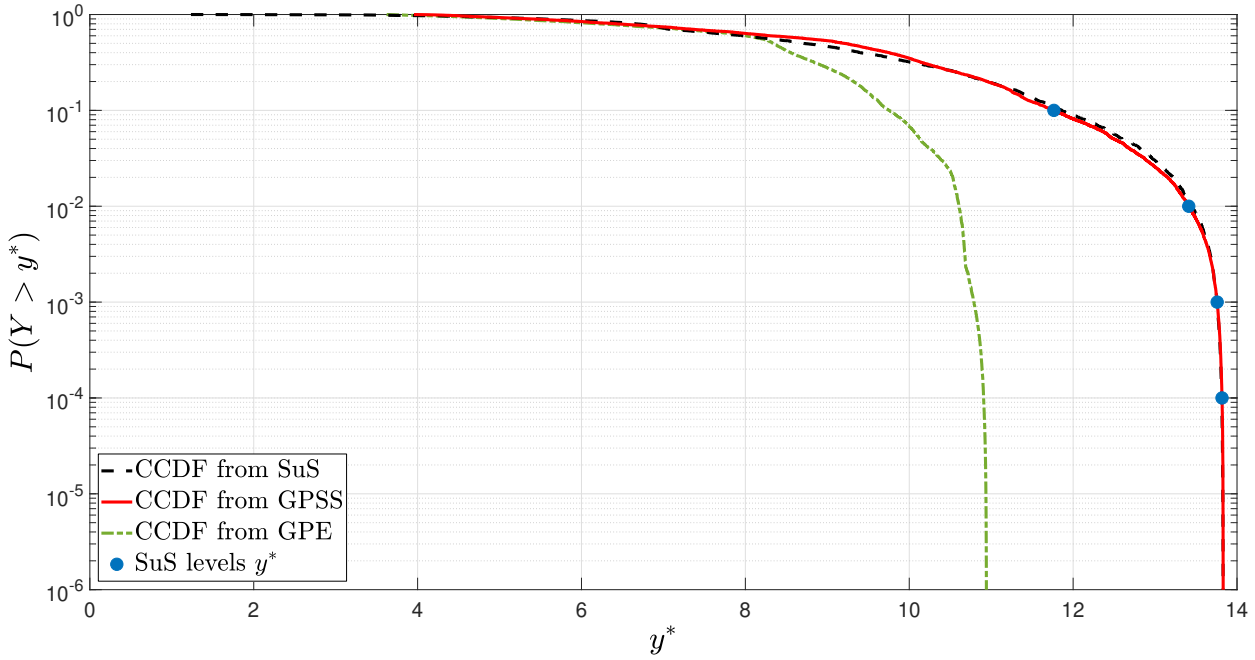

Figure 3: Complementary CDF of the modified Goldstein-Price function according to the GPE (green), GPSS (red) and SuS (black). The blue circles show the intermediate levels of SuS. Each level was populated with 3000 samples.

These rules can be used to tune the algorithm to the studied problem. The performance of the algorithm was tested for smaller values of p_F, by setting $y^* = 13.8308$. The small increase in critical level results in the probability of failure being reduced to $p_F = 2.29 \times 10^{-7}$. GPSS converged in 30 iterations, sampling the simulator a total of 55 times, with $\delta_{p_F} = 36.9\%$ and $\Delta p_F = 1.75\%$. To demonstrate the overall efficiency of GPSS, a series of experiments with y^* of varying orders of magnitude were conducted. These compared the number of evaluations of the real function, Δp_F and δp_F between GPSS and SuS. The results are summarized in Table 2. The study also indicates the number of Monte Carlo samples required to achieve the coefficient of variation at each p_F. It can be seen that GPSS requires significantly fewer samples than both SuS and MCS. A result expected from any surrogate model-based algorithm. Importantly however, GPSS manages to match SuS in both relative error and coefficient of variation, combining accuracy and efficiency.

Table 2: Efficiency analysis for GPSS on the Goldstein-Price function.

	GPSS			SuS			MCS
p_F	N	$\Delta p_F\%$	$\delta p_F\%$	N	$\Delta p_f\%$	$\delta p_F\%$	N
$\mathcal{O}(10^{-2})$	64	0.29	5.2	6×10^3	0.01	5.2	7.03×10^3
$\mathcal{O}(10^{-3})$	67	0.55	12.6	9×10^3	0.66	13.2	1.24×10^4
$\mathcal{O}(10^{-4})$	54	1.39	20.6	12×10^3	0.67	20.2	4.80×10^4
$\mathcal{O}(10^{-5})$	54	0.91	26.1	15×10^3	3.69	27.2	2.91×10^5
$\mathcal{O}(10^{-6})$	61	0.36	30.7	18×10^3	2.54	30.6	2.24×10^6
$\mathcal{O}(10^{-7})$	55	1.75	36.9	21×10^3	2.62	37.3	3.21×10^7

4.2. Gaussian mixture function

This function was created to test the robustness of GPSS to outputs with multimodal failure domains. The function has the form:

$$\begin{aligned} \eta(\mathbf{x}) &= \frac{10^2}{2}\left[a\phi(\mathbf{x}_A) + b\phi(\mathbf{x}_B) + c\phi(\mathbf{x}_C) + d\phi(\mathbf{x}_D)\right] \\ \mathbf{x}_A &= 10\,(\mathbf{x} - 1/4) \\ \mathbf{x}_B &= 10\,(\mathbf{x} - 3/4) \\ \mathbf{x}_C &= [10\,(x_1 - 3/4)\,, 10\,(x_2 - 1/4)] \\ \mathbf{x}_D &= [10\,(x_1 - 1/3)\,, 10\,(x_2 - 5/6)] \end{aligned} \tag{20}$$

In Eq. (20) $\mathbf{x} = [x_1, x_2] \in [0,1]^2$ and $\phi(\cdot)$ is the standard Gaussian PDF. The failure threshold was set as $y^* = 7.9$, with an associated failure probability $p_F = 7.27 \times 10^{-3}$. The corresponding $\varepsilon = 0.05$. The parameters in Eq. (20) are selected in way which results in a failure domain with 3 disjoint sub-domains. The peak on the top left in the sub-plots on Figure 4 is close to, but outside of F.

The GPE was initially trained with $n = 20$ LHS points. Figure 4 shows the process of discovery of the failure domain. Each plot shows the contour values of the intermediate levels $\tilde{y}_\ell^*$ as a red line. The level of y^* is indicated with a black line, but is absent in the first row of Figure 4 as it was not accessible at these stages of the GPE refinement. During most of the process points were added in all three modes. However, from levels $\ell = 15$ and $\ell = 16$ to the end, the algorithm could not improve the approximations in the bottom left and top right modes, and stopped sampling from them. This is reflected in the last four sub-figures of Figure 4 where the new simulation runs, shown as diamonds, are only added to the bottom right peak. This feature of GPSS is useful in the presence of highly disjoint failure domains, where the local quality of the GPE can increase independently and resources will not be wasted where they are not needed. The mean probability of failure, based on 100 runs of

SuS was found to be $\bar{p}_F = 7.31 \times 10^{-3}$. The corresponding c.o.v. and relative error were $\delta_{p_F} = 14.1\%$ and $\Delta p_F = 0.55\%$, respectively. Figure 5 shows the CCDF curve for the failure probability estimated with SuS, the one obtained via GPSS and the one calculated relying on the unimproved GPE. It can be seen, that subset simulation was unable to find failure points using the GPE alone. A study similar to the one performed for the Goldstein-Price function is presented in Table 3. A slight increase in function evaluations with the increase of the failure probability is observed. This behaviour is explained with the fact that larger p_F correspond to bigger portions of the input domain, which require more training points to drive the $s(\cdot, \cdot)$ below ε. Nevertheless, the conclusion arrived at from Table 2 in that GPSS provides a combination between speed and accuracy is confirmed

Table 3: Efficiency analysis for GPSS on the Gaussian mixture function.

	GPSS			SuS			MCS
p_F	N	$\Delta p_F\%$	$\delta p_F\%$	N	$\Delta p_f\%$	$\delta p_F\%$	N
$\mathcal{O}(10^{-2})$	98	0.52	5.8	6×10^3	0.24	6.0	5.58×10^3
$\mathcal{O}(10^{-3})$	64	0.55	14.1	9×10^3	1.59	14.5	6.87×10^3
$\mathcal{O}(10^{-4})$	55	2.49	22.9	12×10^3	2.02	29.9	3.75×10^4
$\mathcal{O}(10^{-5})$	73	1.84	17.7	15×10^3	4.81	32.4	6.28×10^5

4.3. A lattice Boltzmann model of a filter

Cleanliness of fuel is an important aspect of its quality. Fuels can get polluted with a variety of contaminants, including solid particles and water. Water is especially challenging to separate from diesel fuels during daily use. This is due to the chemical nature of these fuels. Designing reliable filters with high percentage of water separation is thus key to the normal operation of engines using diesel products. Non-woven filters are known for their suitability in cleaning fuel from water, dispersed in the medium in the form of microscopic droplets. Successfully capturing droplets below 5μm in diameter, which frequently appear in high-pressure fuel systems, is a challenging task [41].

In order to help in its design, a computer model of the filter is used, which allows the systematic investigation of the filter's performance. Lattice Boltzmann modelling (LBM) is chosen as the simulation paradigm, due to its ability to represent multi-component flows through the complex filter geometry [42].

A full filter model has a multitude of inputs ranging from filter and flow properties, to parameters of the model itself. To simplify this case study, a small LB model of a single-fibre filter is presented in this section. The modelling set-up comprises a 2D simulation of a water droplet (red) submerged in fuel (dark blue) and impacting with a circular fibre (light blue), and is shown in Figure 6. At time $t = 2800$ and $t = 3000$ in Figure 6 some of the water volume

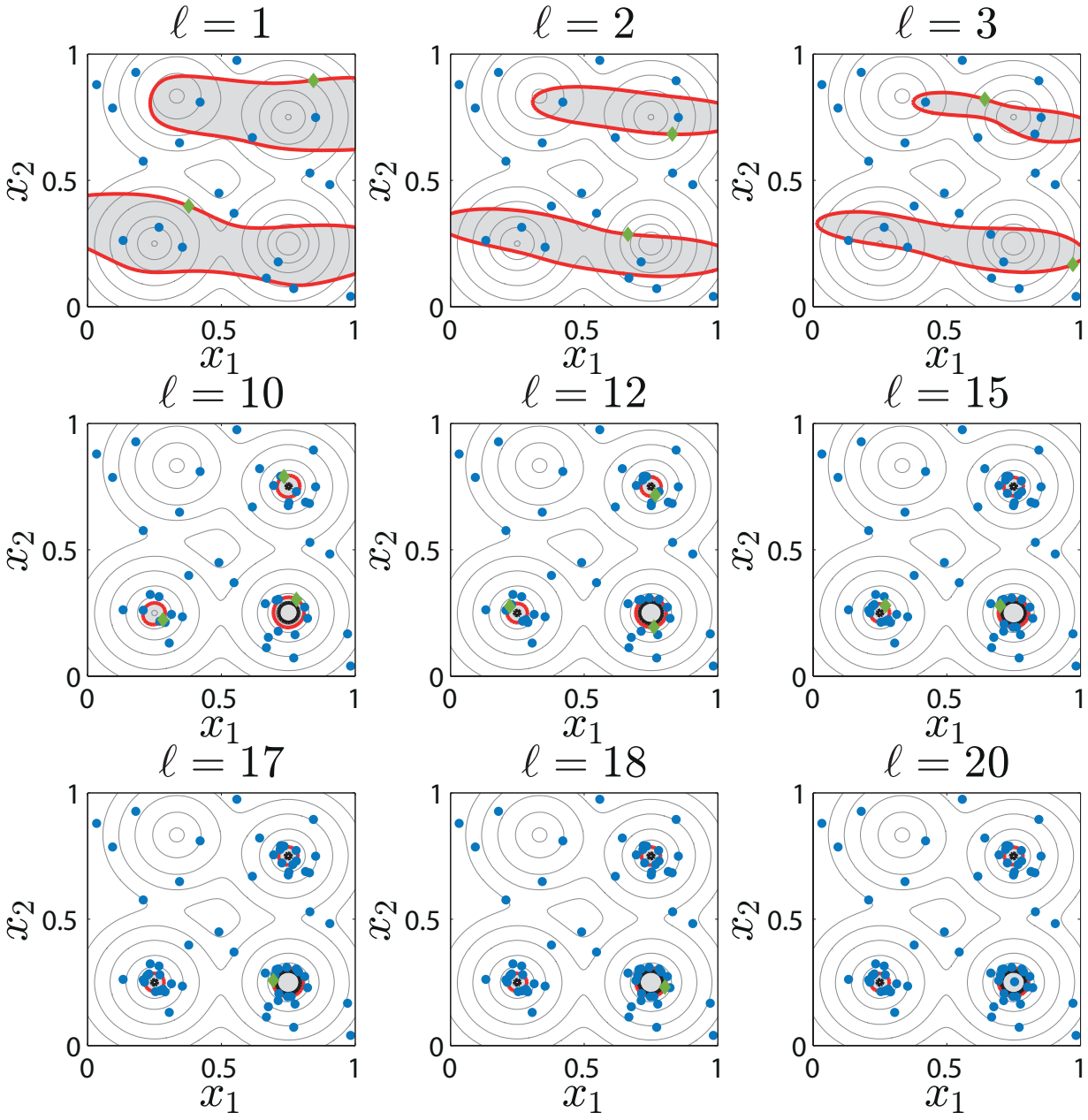

Figure 4: The performance of GPSS on the Gaussian mixture function using Rule 2 in Table 1. The titles in each sub-figure signify the level of the algorithm. Red contours correspond to $\tilde{y}_\ell^*$ and black contours show $\tilde{F}_\ell$. New data points are plotted as diamonds. The number of training samples in each tile is $n = \{20, 22, 24, 44, 50, 59, 62, 63, 64\}$.

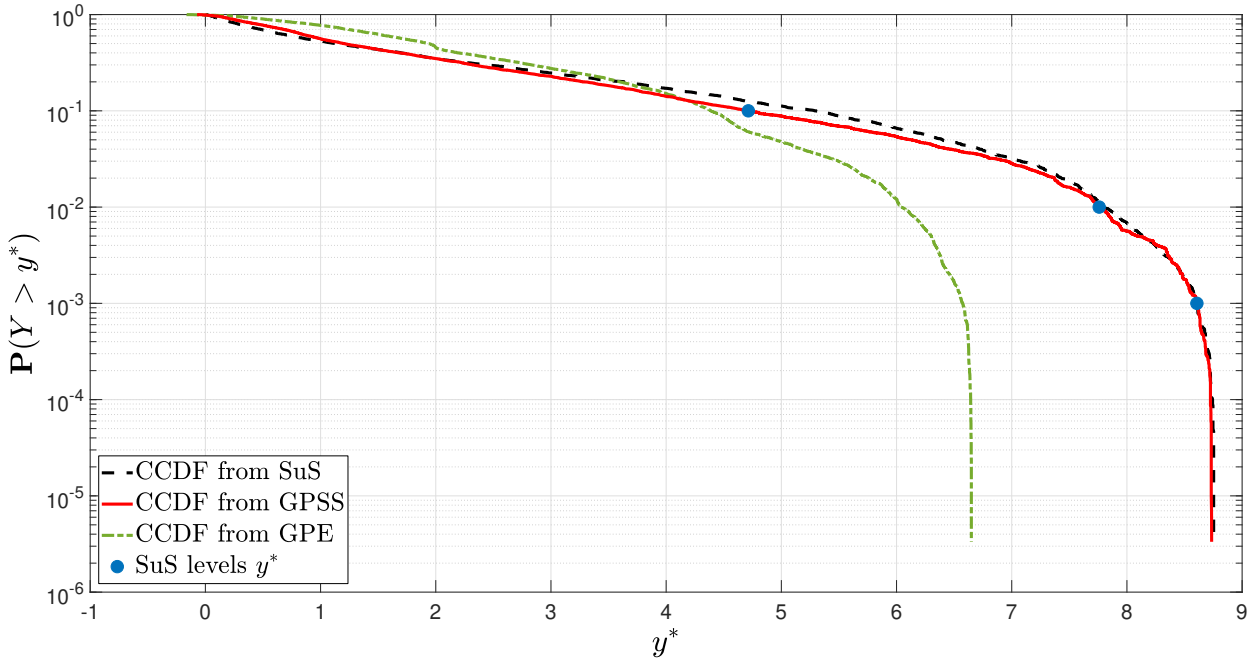

Figure 5: Complementary CDF of the Gaussian mixture function according to the GPE (green), GPSS (red) and SuS (black). The blue circles show the intermediate levels of SuS. Each level was populated with 3000 samples.

can be seen to "escape" from the fibre and go to the outlet of the domain, which represents the downstream portion of the fuel line. The escape volume, denoted with V_e is of interest in this simulation since its presence signifies that the filter has failed to accomplish its task. The model has three input variables, namely droplet diameter, d_d, fibre diameter, d_f and contact angle between fibre and droplet, θ_c. Information about each input along with their ranges is presented in Table 4. The LB model solves a series of linearised partial differential transport equations at each lattice site in the simulation domain. Despite the use of parallel computing techniques and low-level programming, LBM remains a computationally intensive computer code in that, its running time makes it infeasible to perform the number of evaluations required by either MCS or SuS indicated in Table 2 and Table 3.

Table 4: Inputs and their distributions for single-fibre LBM simulations.

Input	**Symbol**	**Distribution**
Droplet diameter (μm)	d_d	$\mathcal{U}(48, 80)$
Fibre diameter (μm)	d_f	$\mathcal{U}(20, 40)$
Contact angle (deg)	θ_c	$\mathcal{U}(130, 180)$

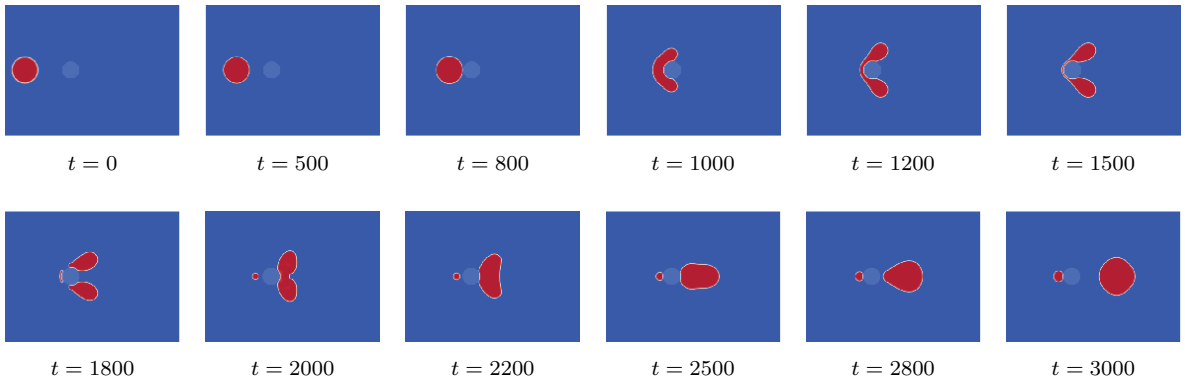

Figure 6: A selection of steps from the single component multiphase model used for preliminary emulation.

A GP emulator for the model was trained with $n = 50$ LHS points from the simulator. Another $m = 50$ LHS validation runs were also obtained. The GPE was validated according to [35] and was determined to be a reasonably accurate representation of the model. After validation, the test set $\mathbf{X}^*$ was added to the training sample and the GPE was refitted, keeping the values of the hyperparameters calculated with the original $n = 50$ samples. The critical level for the analysis was set equal to the value of the maximum training point, $y^* = 7452$ lu. The natural logarithm of the code output was taken such that the transformed threshold was $y^* = 8.916$. The corresponding $\varepsilon = 0.0005$.

GPSS was run with $p_0 = 0.1$ and $N_s = 3000$. Figure 7 shows the CCDF obtained using only the GPE as compared to the one from GPSS. It could be seen that using the GPE alone results in a overconfident estimate of the reliability of the system with $\bar{p}_F = 8.42 \times 10^{-4}$, based on 100 runs of SuS. The same analysis run on GPSS returned $\bar{p}_F = 5.71 \times 10^{-3}$. The coefficient of variation of two estimates was 15.6% and 10.5% for the GPE and GPSS respectively. It should be noted that the small difference in p_F estimated using only the GPE and as opposed to GPSS is only due to the way the physical problem was set up. Defining $y^* = \max(\mathbf{y})$ to be the maximum of the training responses was a way to ensure that GPSS is tasked with providing information about an input region, the emulator has not seen during training. It could be reasoned, that the closeness between the two p_F estimations is due to a smooth response of the code above y^*, which was estimated with some accuracy by the GPE at $\ell = 1$. An interesting observation can be made on the features of the failure domain F. The estimates from the GPE and GPSS differ not only quantitatively, but also qualitatively. Figure 8 shows a comparison between the samples lying in F according to the GPE and GPSS. The former completely misses the two regions in top and bottom left. The region around $d_f = 0.7$, $\theta_c = 0.4$ which SuS populates with failure points when using the initial GPE can be ascribed to a local extrapolation issue, resolved by GPSS.

5. Conclusion

A surrogate-model-based method for reliability analysis, called GPSS was presented. The algorithm combines standard practice tools, namely subset simulation and Gaussian process

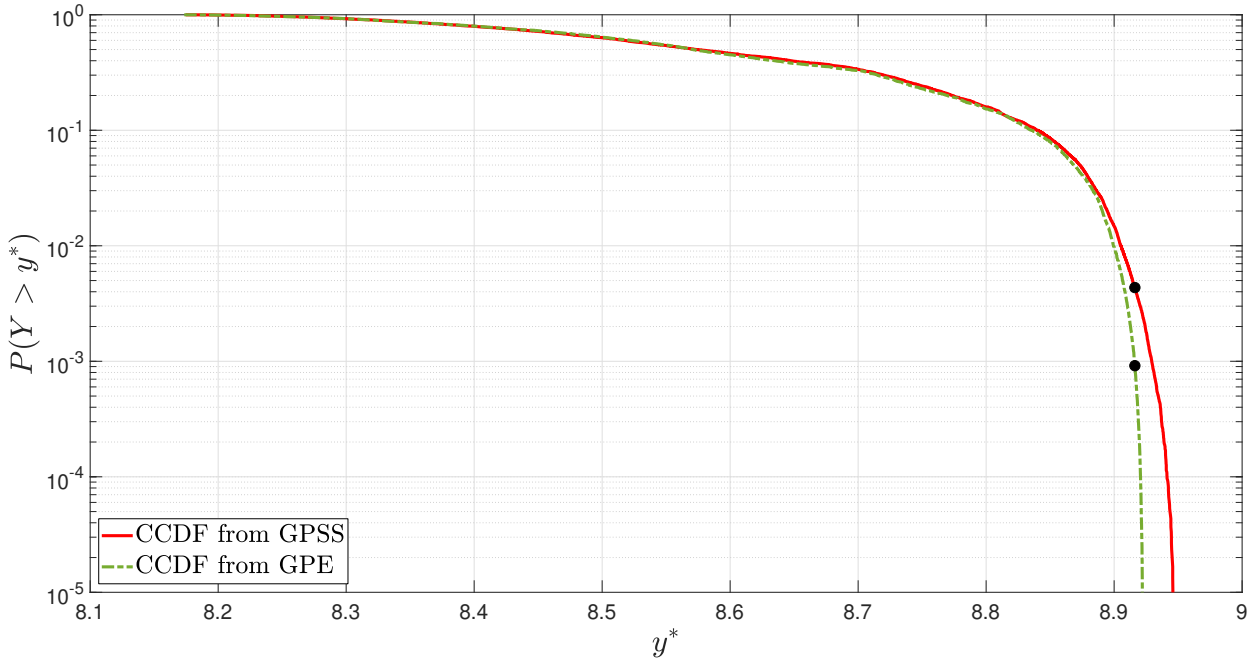

Figure 7: Complementary CDF of the LBM code using the GPE only (dash-dot) and GPSS (solid). The black circles show the probability of exceeding y^* estimated from the two surrogates.

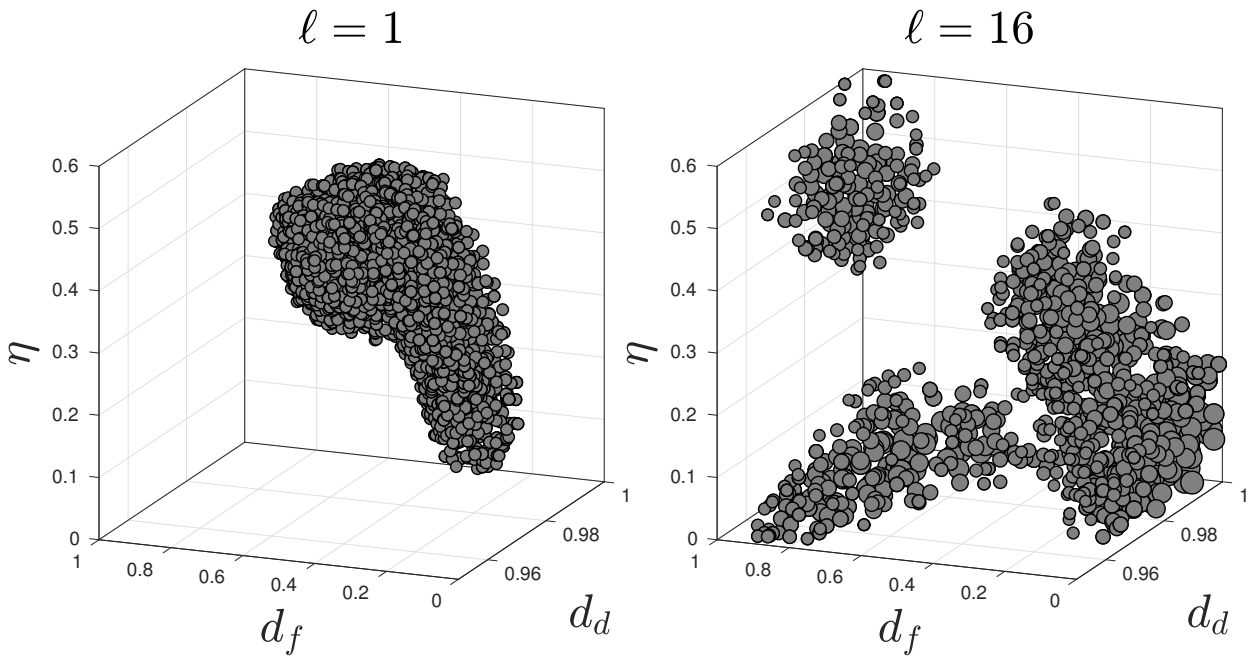

Figure 8: Samples in $\tilde{F}_\ell$ from the GPE ($\ell = 1$) and after GPSS ($\ell = 16$). GPSS not only estimates the probability of failure, but also provides an identification of the failure domain. The multimodality of F is clearly seen after $\ell = 16$ iterations. The size of the markers indicates the value of the response at that data point.

emulation into an efficient algorithm for reducing the cost of reliability analysis of complex computer codes.

GPSS has two main improvements over existing algorithms. Firstly, a complete methodology for selecting points to augment the emulator was presented which leads allows uni- and multi-modal failure domains to be analysed, whilst minimizing the need to obtain computationally expensive simulator runs. GPSS automatically stops sampling subregions of the failure domain for which the emulator is accurate enough.

Secondly, a stopping condition utilizing the full predictive capabilities of the Gaussian process emulator was presented. It is based on the direct similarity between the model and the emulator in the failure domain. The quality of the performance of the algorithm was demonstrated by the means of two illustrative benchmark problems and a computational fluid study of a filtration model. Results showed that GPSS provides estimators of the probability of failure with errors comparable to subset simulation and direct Monte Carlo, but at a fraction of the computational cost. Future research envisions the implementation of different MCMC samplers that can potentially increase the efficiency of subset simulation.

Acknowledgments

P.O. Hristov acknowledges the funding provided by Parker-Hannifin Manufacturing (UK) Ltd.

F.A. DiazDelaO would like to thank the Isaac Newton Institute for Mathematical Sciences for support and hospitality during the programme "Uncertainty quantification for complex systems: theory and methodologies" when work on this paper was undertaken. This work was supported by EPSRC through grants EP/K032208/1 and EP/R014604/1.

References

[1] S. K. Au, J. L. Beck, Estimation of small failure probabilities in high dimensions by subset simulation, Probabilistic Engineering Mechanics 16 (4) (2001) 263–277.

[2] A. Forrester, A. Sobester, A. Keane, Engineering Design via Surrogate Modelling: A Practical Guide, 1st Edition, John Wiley & Sons, 2008.

[3] A. Keane, P. Nair, Computational Approaches for Aerospace Design, John Wiley & Sons, Derby, 2005.

[4] T. W. Simpson, Comparison of Response Surface and Kriging Models in the Multidisciplinary Design of an Aerospike Nozzle, Tech. Rep. February, NASA Langley (1998).

[5] C. Turner, Generic sequential sampling for metamodel approximations, in: ASME 2003 Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Chicago, 2003, pp. 1–10.

[6] N. V. Queipo, R. T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, P. Kevin Tucker, Surrogate-based analysis and optimization, Progress in Aerospace Sciences 41 (1) (2005) 1–28.

[7] G. G. Wang, S. Shan, Review of Metamodeling Techniques in Support of Engineering Design Optimization, Journal of Mechanical Design 129 (4) (2007) 370.

[8] H.-S. Li, S.-K. Au, Design optimization using Subset Simulation algorithm, Structural Safety 32 (6) (2010) 384–392.

[9] P. Ranjan, D. Bingham, G. Michailidis, Sequential Experiment Design for Contour Estimation From Complex Computer Codes, Technometrics 50 (4) (2008) 527–541.

[10] V. Picheny, D. Ginsbourger, O. Roustant, R. T. Haftka, N.-H. Kim, Adaptive Designs of Experiments for Accurate Approximation of a Target Region, Journal of Mechanical Design 132 (7) (2010) 071008.

[11] Z. Zhu, X. Du, Reliability Analysis With Monte Carlo Simulation and Dependent Kriging Predictions, Journal of Mechanical Design 138 (12) (2016) 121403.
[12] I. G. Osio, C. H. Amon, An engineering design methodology with multistage Bayesian surrogates and optimal sampling, Research in Engineering Design 8 (4) (1996) 189–206.
[13] I. K. Bang, D. S. Han, G. J. Han, K. H. Lee, Structural optimization for a jaw using iterative Kriging metamodels, Journal of Mechanical Science and Technology 22 (9) (2008) 1651–1659.
[14] R. C. Kuczera, Z. P. Mourelatos, On Estimating the Reliability of Multiple Failure Region Problems Using Approximate Metamodels, Journal of Mechanical Design 131 (12) (2009) 121003.
[15] V. Dubourg, Adaptive surrogate models for reliability analysis and reliability-based design optimization, Ph.D. thesis, Blaise Pascal University (2011).
[16] B. Jones, R. T. Johnson, Design and Analysis for the Gaussian Process Model, Quality and Reliability Engineering International 23 (2007) 517–543.
[17] M. D. McKay, R. J. Beckman, W. J. Conover, Comparison of three methods for selecting values of input variables in the analysis of output from a computer code, Technometrics 21 (2) (1979) 239–245.
[18] B. Echard, N. Gayton, M. Lemaire, AK-MCS: An active learning reliability method combining Kriging and Monte Carlo Simulation, Structural Safety 33 (2) (2011) 145–154.
[19] T. H. Lee, J. J. Jung, A sampling technique enhancing accuracy and efficiency of metamodel-based RBDO: Constraint boundary sampling, Computers and Structures 86 (13-14) (2008) 1463–1476.
[20] V. Dubourg, B. Sudret, J. M. Bourinet, Reliability-based design optimization using kriging surrogates and subset simulation, Structural and Multidisciplinary Optimization 44 (5) (2011) 673–690.
[21] R. Jin, W. Chen, A. Sudjianto, On sequential sampling for global metamodelling in engineering design, in: DETC ASME 2002 Design Engineering Technical Conferences And Computers and Information in Engineering Conference, Montreal, 2002, pp. 1–10.
[22] G. Su, B. Yu, Y. Xiao, L. Yan, Gaussian Process Machine-Learning Method for Structural Reliability Analysis, Advances in Structural Engineering 17 (9) (2014) 1257–1271.
[23] L. Zhang, Z. Lu, P. Wang, Efficient structural reliability analysis method based on advanced Kriging model, Applied Mathematical Modelling 39 (2) (2015) 781–793.
[24] J. Bect, D. Ginsbourger, L. Li, V. Picheny, E. Vazquez, Sequential design of computer experiments for the estimation of a probability of failure, Statistics and Computing 22 (3) (2012) 773–793.
[25] B. J. Bichon, M. S. Eldred, L. P. Swiler, S. Mahadevan, J. M. McFarland, Efficient Global Reliability Analysis for Nonlinear Implicit Performance Functions, AIAA Journal 46 (10) (2008) 2459–2468.
[26] Z. Wen, H. Pei, H. Liu, Z. Yue, A Sequential Kriging reliability analysis method with characteristics of adaptive sampling regions and parallelizability, Reliability Engineering & System Safety 153 (2016) 170–179.
[27] C. Tong, Z. Sun, Q. Zhao, Q. Wang, S. Wang, A hybrid algorithm for reliability analysis combining Kriging and subset simulation importance sampling, Journal of Mechanical Science and Technology 29 (8) (2015) 3183–3193.
[28] Z. Sun, J. Wang, R. Li, C. Tong, LIF: A new Kriging based learning function and its application to structural reliability analysis, Reliability Engineering & System Safety 157 (2017) 152–165.
[29] X. Yang, Y. Liu, L. Gao, Unified reliability analysis by active learning Kriging model combining with random-set based Monte Carlo simulation method, Int. J. Numer. Meth. Engng 108 (2016) 1343–1361.
[30] X. Huang, J. Chen, H. Zhu, Assessing small failure probabilities by AK-SS: An active learning method combining Kriging and Subset Simulation, Structural Safety 59 (2016) 86–95.
[31] B. Bichon, S. Mahadevan, M. Eldred, Reliability-Based Design Optimization Using Efficient Global Reliability Analysis, in: 50th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, Palm Springs, 2009.
[32] S. K. Au, Y. Wang, Engineering Risk Assessment with Subset Simulation, Wiley, 2014.
[33] C. E. Rasmussen, C. K. I. Williams, Gaussian processes for machine learning, MIT Press, 2006.
[34] A. O'Hagan, Bayesian analysis of computer code outputs: A tutorial, Reliability Engineering and System Safety 91 (10-11) (2006) 1290–1300.
[35] L. S. Bastos, A. O'Hagan, Diagnostics for Gaussian Process Emulators, Technometrics 51 (4) (2009)

425–438.
[36] J. Oakley, Bayesian uncertainty analysis for complex computer codes, Ph.D. thesis, University of Sheffield (1999).
[37] J. L. Loeppky, J. Sacks, W. Welch, Choosing the Sample Size of A Computer Experiment: A Practical Guide, Technometrics 51 (4) (2008) 366–376.
[38] D. Jones, M. Schonlau, W. Welch, Efficient global optimization of expensive black-box functions, Journal of Global optimization 13 (4) (1998) 455–492.
[39] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise, in: Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, 1996, pp. 226–231.
[40] V. Dubourg, B. Sudret, F. Deheeger, Metamodel-based importance sampling for structural reliability analysis, Probabilistic Engineering Mechanics 33 (2013) 47–57.
[41] P. O. Hristov, Numerical modelling and uncertainty quantification of biodiesel filters, Ph.D. thesis, University of Liverpool (2018).
[42] M. C. Sukop, D. T. Thorne, Lattice Boltzmann Modeling: An Introduction for Geoscientists and Engineers, Springer, 2006.

Appendix A. Some existing methods

Table A.1: Comparison of main features between GPSS and existing methods. Refer to Section 1 for an explanation of potential problems.

Method	DoE	Learning	Termination	Remarks
GPSS	LHS.	Multipoint-subset-simulation maximum expected improvement.	Discrepancy between GPE and simulator in F.	Algorithm provided in this paper.
AK-MCS	Random sample.	Maximum expected feasibility; U-function.	Coefficient of variation of p_F.	One point added per iteration from a pre-generated random sample. See [18].
AK-SS	LHS.	U-function as in [18].	Coefficient of variation of p_F.	p_F calculated via SuS. See [30].
AK-SSIS	LHS.	Importance-sampling-enhanced-subset-simulation U-function.	Coefficient of variation of p_F.	Reliability indices feature in the algorithm. See [27].
LIF	LHS.	Least improvement function.	Threshold on the uncertainty function.	Fixed population of augmentation points. See [28].
Dubourg	Empty.	Slice sampling with k-means for multipoint selection.	Overall improvement in the reliability indices.	SuS used to perform reliability studies. See [20]