Available online at www.eccomasproceedia.org
Eccomas Proceedia UNCECOMP (2017) 198-211

ECCOMAS
Proceedia

UNCECOMP 2017
2nd ECCOMAS Thematic Conference on
Uncertainty Quantification in Computational Sciences and Engineering
M. Papadrakakis, V. Papadopoulos, G. Stefanou (eds.)
Rhodes Island, Greece, 15–17 June 2017

# ADAPTIVE EMULATION-BASED RELIABILITY ANALYSIS

**P. O. Hristov[1], F. A. DiazDelaO[1], K. J. Kubiak[2] and U. Farooq[3]**

[1]Institute for Risk and Uncertainty, University of Liverpool
Liverpool L69 7ZF, United Kingdom
e-mail: {sgphrist, fado}@liv.ac.uk

[2] School of Computing and Engineering, University of Huddersfield
Queensgate, Huddersfield, HD1 3DH
e-mail: k.kubiak@hud.ac.uk
[3] Parker Hannifin Manufacturing (UK) Ltd.
e-mail: umer.farooq@parker.com

**Keywords:** Efficient reliability analysis, Gaussian process emulation, Subset simulation.

**Abstract.** *This paper presents an approximation method for performing reliability analysis with high fidelity computer codes at a reasonable computational cost. Complex models are common in science and engineering due to their ability to substitute costly and some times infeasible practical experiments. These models, however, suffer from high computational cost. This causes problems when performing sampling - based reliability analysis, since the failure modes of the system typically occupy a small region of the input space and thus relatively large sample sizes are required for the accurate estimation of 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 to efficiently populate those regions of the initial approximation which are likely to lead to the performance function exceeding a predefined critical threshold. Among all samples, the ones that are likely to contribute most to increasing the quality of the surrogate in the vicinity of the failure regions are selected, using Bayesian optimisation methods. The iterative nature of the method ensures that an arbitrarily accurate approximation of the failure region in performance space is developed at a reasonable computational cost. The presented method is applied to a number of benchmark problems.*


Peer-review under responsibility of the organizing committee of UNCECOMP 2017.
doi: 10.7712/120217.5363.16925

## 1 INTRODUCTION

Reliability analysis, in the most general sense, is concerned with the calculation of a probability of failure, $p_F$ and the identification of the failure domain, $F$ of a function $y = \eta(\mathbf{x})$.

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

The most straightforward method to evaluate the multidimensional integral in (1) is direct Monte Carlo (DMC) sampling. The accuracy of DMC estimates increases with the number of samples. For a typical system the failure region, $F$ is small with respect to its input domain and is considered a rare event. Thus, many thousands of evaluations of the code have to be performed to ensure that $F$ is populated with sufficient number of samples to allow a reasonably accurate estimation of its properties. However, engineering models are in general computationally expensive to evaluate and thus it becomes infeasible to use DMC methods with them. A method widely used in engineering for performing reliability analysis is *Subset Simulation* (SuS) [1]. Despite the significant improvement SuS brings over DMC, it could still be quite expensive to use it directly with the code. In order to address the computational cost of the simulator, one can build a surrogate model for the output of the code. There exists a large number of methodologies for approximating the output of expensive codes, see *e.g.* [2]. 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 can be used as an inexpensive substitute for the simulator, on which reliability analysis can be performed. The issue with such use of the surrogate is that it is typically *trained* via some space-filling sampling plan which aims at exploring the input domain of the simulator with as few points as possible. Such 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 rare events* is built on *frequent* events. The topic of using surrogate models to perform reliability analysis is not a new one and there are a number of frameworks proposed to reduce the cost of the computations. It should be stressed that the reliability analysis literature is as vast as the number of methods that feature in it. Therefore a review of the techniques which are related to the procedure presented in this article is provided. Most, if not all reviewed work relies on sequential sampling for the gradual improvement of the surrogate in the vicinity of the failure domain. It is worth mentioning that some authors such as [3, 4] have explored those strategies for the accurate estimation of target regions in general. They consider that the emulator should only be refined in the regions of interest and that in order to obtain accurate reliability estimates a good quality surrogate is all that is needed. An opposing view is given by [5] where the authors work directly with the reliability estimates as a measure of analysis quality. The present article agrees with and extends the former idea.

A short literature review focused on reliability analysis is provided below. In general, most approaches follow a framework which is composed of a sampling rule, utility function, stopping criterion and any other specific details. The authors of [6] use constraint boundary sampling to select improvement points. A combination between a MCMC sampling and k-means clustering is used to select new data points in [7]. An approach based on the first order reliability method is used in [8] and a probabilistic classification function is presented in [9]. In [10] the authors derive a stepwise uncertainty reduction (SUR) methodology based on expected improvement (EI) [11], but formulated from a Bayesian risk perspective. They use SUR to select new design points to improve the surrogate. Other methods that use EI or EI-based strategies are [12, 13, 14]. Other previously used utility functions include, the U-function [13, 15], and the

improved U-function [16], least improvement function [17] and an unnamed expression in [18]. All approaches based on a utility function, except [14] search the entire input space for a candidate point that maximizes that function and add it to the training plan for the next iteration of the algorithm. Furthermore, many of the aforementioned strategies rely on a pre-generated population of samples (e.g. [13]), which could prove to be a suboptimal approach. Last from a sampling point of view, the majority of the methods sample 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. These issue are addressed in the present article.The other major part of all adaptive algorithms is the stopping condition. This ranges from the use of reliability indices [7, 8] through error in the estimation of the failure probability [5, 13, 15, 16, 17] and forms of measure of the discrepancy between the GPE predictions and code observations [4,6,9,18] to thresholds on the learning function [3, 12, 14]. Most frameworks use some form of statistic related to the surrogate, which, depending on the use and complexity of the problem, could prove insufficiently robust. In this paper a stopping condition which relies implicitly on the similarity between the surrogate and the model is proposed to terminate the learning process.

Finally, Gaussian process emulation is not the only surrogate used in reliability analysis. Among others, general response surfaces [19, 20, 21, 22], neural networks [23] and support vector machines [24] have been used.

The aim of this paper is to propose an improved efficient algorithm for performing reliability analysis with complex computer codes.

The remainder of the article is structured as follows: Section 2 briefly introduces Subset Simulation. Section 3 summarizes the theory behind Gaussian process emulation. Section 4 introduces the proposed method and Section 5 demonstrates the performance of the algorithm before any conclusions are drawn in Section 6.

## 2 SUBSET SIMULATION

One very important problem in engineering is the estimation of the probability of failure, $p_F$ of a system 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 model,

$$y = \eta(\mathbf{x}) \tag{2}$$

thus the failure domain is defined as the values of $\mathbf{x}$ which cause the system response, $y$ to exceed some critical value $y_c$

$$F = \{\mathbf{x} : \eta(\mathbf{x}) > y_c\} \tag{3}$$

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] aims to divide the rare event into a series of nested less-rare events.

$$F \subset F_m \subset F_{m-1} \subset \ldots \subset F_1 \tag{4}$$

In Eq. (4) $F_1$ is a relatively frequent event. Given that sequence, it can be shown that the probability of the rare event $F$ could be expressed as a product of larger conditional probabilities:

$$\mathbb{P}(F) = \mathbb{P}(F_1) \cdot \mathbb{P}(F_1|F_2) \cdot \ldots \cdot \mathbb{P}(F|F_m) = p_F \tag{5}$$

Beginning from the unconditional level $F_1$, the algorithm "probes" the input space $\mathcal{X}$ via direct Monte Carlo sampling. Then, based on the values of $y$ in Eq. (2) it constructs the first intermediate failure threshold, $y_1^* < y_c$, defining a "relaxed failure domain", $F_1$. SuS then populates $F_1$ using a MCMC algorithm. The generation of intermediate levels 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 [25].

## 3 GAUSSIAN PROCESS EMULATION

Simulators used to model complex scientific phenomena are usually very computationally expensive. This is to say that a single evaluation of the code's output at a given set of input values takes sufficiently long time, as to prohibit any type of analysis which requires a large number of model runs. These include finite element modelling, computational fluid dynamics and other ubiquitously used simulation tools. Even though SuS requires fewer samples than DMC to estimate the failure probability, it still relies on such amounts of simulation runs which result in very large computational times. Clearly, the analysis cannot be carried out using the code directly. In such cases it is common to use a less expensive approximation of the code output. These approximations are widely known as *metamodels* or *emulators*. There is a number of existing metamodelling techniques, but for reasons outlined below, Gaussian process emulation (GPE) is used. Formally, the model structure is expressed as:

$$\eta(\mathbf{x}) = h(\mathbf{x})^T \boldsymbol{\beta} + Z(\mathbf{x}) \tag{6}$$

where $\eta(\mathbf{x})$ is the simulator output as a function of its inputs, $h(\mathbf{x})^T$ is a known function of the inputs, $\boldsymbol{\beta}$ is a vector of unknown coefficients and $Z(\mathbf{x})$ is a Gaussian process with zero mean and covariance $\sigma^2 c(\mathbf{x}, \mathbf{x}'; \boldsymbol{\psi})$. The function $h(\mathbf{x})$ should express any expert opinion about the form of the simulator output and together with the parameter $\boldsymbol{\beta}$ reflects its overall trend. In practice, however, the trend is often taken to be constant as $h(\mathbf{x}) = 1$ and $\beta \in \mathbb{R}$, charging the Gaussian process in Eq. (6) with the responsibility of capturing the behaviour of the underlying function [26]. In the formulation above, $\sigma^2$ is a scale parameter and $\boldsymbol{\psi}$ is a parameter specifying the behaviour of the correlation function, $c(\cdot, \cdot; \boldsymbol{\psi})$.

Using Gaussian process emulation, a *posterior* probability distribution for the mean of the computer code's output can be constructed, conditional on a relatively small number of simulator runs, $\mathbf{y}$ and the parameter estimators, $\hat{\boldsymbol{\theta}} = \{\hat{\beta}, \hat{\sigma^2}, \hat{\boldsymbol{\psi}}\}$. It can be shown that at any unobserved point, $\mathbf{x}^*$ this distribution has the form:

$$\eta(\mathbf{x}^*)|\mathbf{y}, \hat{\boldsymbol{\theta}}, \sim \mathcal{N}(m(\cdot), C(\cdot, \cdot)) \tag{7}$$

with posterior predictive mean (also called a *surrogate*):

$$m(\mathbf{x}^*) = \hat{\beta} + \mathbf{t}(\mathbf{x}^*)^T \boldsymbol{C}^{-1}(\mathbf{y} - \mathbf{1}\hat{\beta}) \tag{8}$$

and posterior predictive variance:

$$C(\mathbf{x}^*, \mathbf{x}'^*) = \hat{\sigma}^2(c(\mathbf{x}^*, \mathbf{x}^*; \hat{\boldsymbol{\psi}}) - \boldsymbol{t}(\mathbf{x}^*)^T \boldsymbol{C}^{-1} \boldsymbol{t}(\mathbf{x}'^*)) \tag{9}$$

In Eq. (8) and Eq. (9), $\mathbf{C} \in \mathbb{R}^{n \times n}$ is such that $C_{ij} = c(\mathbf{x}_i, \mathbf{x}_j; \hat{\boldsymbol{\psi}})$; $\mathbf{t}(\mathbf{x}^*) \in \mathbb{R}^n$ such that $\mathbf{t}(\mathbf{x}^*) = (c(\mathbf{x}^*, \mathbf{x}_1; \hat{\boldsymbol{\psi}}), \ldots, c(\mathbf{x}^*, \mathbf{x}_n; \hat{\boldsymbol{\psi}}))^T$; and $\mathbf{1} \in \mathbb{R}^n$ such that $\mathbf{1} = (1, \ldots, 1)^T$.The process of learning $\hat{\boldsymbol{\theta}}$ from observed data is referred to as *training* and is well described in [2] from a classical prospective or in [27, 28] from Bayesian standpoint. Once the emulator is trained, its posterior distribution can be sampled many times at an affordable cost to provide data for various analyses.

## 4 GAUSSIAN PROCESS SUBSET SIMULATION

### 4.1 Initialization and sampling

In order to reliably use the emulator it needs to be of sufficient quality around the estimated failure regions. This is not usually the case with GP approximations built using data from the high probability regions. For the physical model it is assumed that the critical failure threshold, $y_c$ is known (given *a priori*) and sensible (i.e. there is a set of values of $\mathbf{x}$ for which $\eta(\mathbf{x}) > y_c$). The algorithm starts by building a GP emulator based on a parsimonious set of data points, selected according to a space-filling strategy (e.g. Latin hypercube sampling (LHS) [29]). The GPE can be validated to check if there are any large discrepancies between the emulator and simulator as in [30]. This is done to ensure that the initial approximation is of a reasonable overall quality. It is very likely that the original threshold is not reachable from the first GPE which learned about the model from frequent data. That is to say,

$$F_1 = \{\mathbf{x} : \mathbb{E}_1[\eta(\mathbf{x})|y] > y_c\} = \varnothing \tag{10}$$

where $F_j$ is the failure domain according to the $j^{th}$ emulator and $\mathbb{E}_j[\cdot]$ is its predictive mean, whose functional form is given in Eq. (8). In general, for a high-dimensional model it will be feasible, but slow to search the input domain exhaustively, since a lot of samples are needed to uncover the location of the failure modes if such are present. Computationally this translates to the calculation of large number of distances between points. Therefore, it is proposed to use SuS to sample from the posterior predictive mean of the emulator. As outlined in Section 2, 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 naturally converge due to all candidate samples being rejected. Then, an alternative "failure level", $y_{c_1}^{GP}$ could be set for which $\mathbb{P}(\mathbb{E}_1[\eta(\mathbf{x})|y] > y_{c_1}^{GP}) > 0$. This approach gives rise to an intermediate emulator failure domain with respect to $y_{c_1}^{GP}$, denoted as $F_1^{GP}$. This domain and its associated probability may still be a rare event and thus SuS is used to populate it efficiently. The purpose here is to sample from $F_1^{GP}$ rather than to estimate the probabilities of failure (conditional or otherwise). The emulator is still a very efficient approximation of the code and thus in theory one can search it exhaustively. However, a preference is given SuS as it leads to a quick and reliable convergence, especially in high dimensions. The algorithm can be ran with a fairly large number of samples (a lot less than if exhaustive search is performed) at each level in order to ensure that even truly rare subregions of $F_1^{GP}$ are populated. This raises the question of how to calculate $y_{c_j}^{GP}$ such that SuS converges, while at the same time it explores potentially interesting regions of the input space. The candidates vary from a single significant point (e.g. minimum or maximum) through moment estimators (e.g. sample mean) to more sophisticated equiprobable measures (e.g. sample percentiles). The criterion that was found to give satisfactory results is presented in Eq. (11). Essentially, the idea is to select the current threshold as the average of all training responses that are lying above the previous critical level and then take the average between the resulting quantity and the previous level value. This method ensures that the current threshold will lie above the previous one, while also safeguarding against setting it too high and discarding regions that may lead to system failure.

$$y_{c_j}^{GP} = \begin{cases} \frac{\sum_{i=1}^{N} y_i}{N} & \text{if } j = 1 \\ \frac{1}{2}\left[\frac{\sum_{i=1}^{N} y_i \mathbb{I}\left(y_i > y_{c_{(j-1)}}^{GP}\right)}{N} + y_{c_{(j-1)}}^{GP}\right] & \text{if } j > 1 \end{cases} \tag{11}$$

In Eq. (11), $y_i$ are all $N$ training responses and $\mathbb{I}(\cdot)$ is an indicator function.

### 4.2 Sample selection

When SuS converges, it populates the space with points whose predicted response lies above the current failure level $y_{c_j}^{GP}$. Among all samples $\mathbf{X}^{F_j^{GP}}$ that lie in $F_j^{GP}$ some additional point/s, $\mathbf{x}_{add}$ at which to sample the model have to be selected before fitting the next emulator prediction. Since the model is expensive, these points have to be prudently selected such that they maximize the amount of information gain from sampling at their locations. A popular choice for selecting the coordinates is the *expected improvement* (EI) function [11].

$$\begin{aligned}\mathbb{E}[I(\mathbf{x})]_j =&(y_c - \mathbb{E}_j[\eta(\mathbf{x})|y])\Phi\left(\frac{y_c - \mathbb{E}_j[\eta(\mathbf{x})|y]}{\mathbb{V}_j[\eta(\mathbf{x})|y]}\right)+ \\ &+\mathbb{V}_j[\eta(\mathbf{x})|y]\phi\left(\frac{y_c - \mathbb{E}_j[\eta(\mathbf{x})|y]}{\mathbb{V}_j[\eta(\mathbf{x})|y]}\right)\end{aligned} \tag{12}$$

The EI expression as used in this framework is given in Eq. (12). Following the notation above, the subscript $j$ denotes information regarding the $j^{th}$ emulator. This includes $\mathbb{V}_j[\cdot]$ which is the posterior variance of the $j_{th}$ emulator given in its functional form in Eq. (9). The symbols $\Phi(\cdot)$ and $\phi(\cdot)$ denote the cumulative and probability distribution functions of a standard normal random variable, respectively. Expected improvement is a strategy that balances exploitation of the emulator mean and exploration of the design space based on the its variance. Gaussian process emulators are particularly suited for use with EI, since a predictive variance is generated as part of sampling the posterior distribution.

$$\mathbf{x}_{add} = \underset{\mathbf{X}^{F_j}}{\operatorname{argmax}}\, \mathbb{E}\big[I\big(\mathbf{X}^{F_j}\big)\big]_j \tag{13}$$

Expected improvement can be run on each point in $F_j^{GP}$ and the one that maximizes it will be selected as the one that is expected to bring the greatest improvement in the quality of the next level GPE. However, applying expected improvement directly to the data poses the risk of neglecting subregions in the intermediate emulator failure domain, $F_j^{GP}$. Unless $F_j^{GP}$ is definitely not disjoint, the presence of separate modes has to be accounted for. This could be achieved by, first identifying the structure of $F_j^{GP}$, detecting any modes and calculating EI on the samples in each mode. A clustering algorithm (here DBSCAN [31]) is used to discover the separate failure sub-domains. Ultimately, the design plan for the emulator at the next level is composed of the current design plus the new points, formally $\mathcal{D}_{j+1} = \mathcal{D}_j \cup \{\mathbf{x}_{add}, \eta(\mathbf{x}_{add})\}$, where $\mathbf{x}_{add}$ is given in Eq. (13).

Adaptively improving the approximation from the GP will represent the failure regions with increasing accuracy. However, due to the generic nature of both Subset Simulation and expected improvement, new points can be added in regions where the GPE is already good enough for the purposes of estimation of the failure characteristics. Thus a condition is needed which dictates how to choose improvement points. The posterior predictive variance of the emulator is a good indication of the local quality of the approximation. Since the GPE is an interpolator, at any given training point $\mathbb{V}_j[\eta(\mathbf{x})|y] = 0$. Therefore, if at any level of the GPE improvement the value of the posterior predictive variance for a sample point lies below a certain level $\varepsilon$, that point should not be added to the design for the next level. This strategy prevents the algorithm from suggesting new samples in regions where the emulator is already doing well and thus saving potentially appreciable time for code evaluations. The correct value of $\varepsilon$ is problem dependent since it scales with the output of the simulator. Of course it could be set arbitrarily small, but attention needs to be paid to the trade-off between final emulator accuracy and computational cost.

### 4.3 Stopping condition

As outlined above, the GPE could be made to reflect the failure regions of the true function with an arbitrary precision. However to keep the procedure efficient it needs to be refined just enough for the purposes of the underlying analysis. In order to stop the iterative generation of predictions a rule inspired by SuS is proposed. Consider running subset simulation with the true function. By design SuS will stop generating new levels once a sufficient number of 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 is the ability of SuS to generate the same number of samples that belong to the failure domain and have $\mathbb{V}_j[\eta(\mathbf{x})|y] < \varepsilon$. If this condition is satisfied, SuS will not be able to differentiate the model from the emulator.

## 5 NUMERICAL EXPERIMENTS

In this section, the performance of the algorithm is demonstrated with three benchmark problems. These were chosen to test different aspects of GPSS. The function in Section 5.1 presents a challenging limit state contour for the GPE. The problem in Section 5.2 tests the ability of GPSS to deal with disjoint failure domains and Section 5.3 explores the performance of the algorithm in high dimensions.

### 5.1 Four branch series system

The four branch series system is popular in the reliability literature [17]. The performance function has the following form:

$$\eta(\mathbf{x}) = \min \left\{ \begin{array}{l} 3 + 0.1(x_1 - x_2)^2 - (x_1 + x_2)/\sqrt{2} \\ 3 + 0.1(x_1 - x_2)^2 + (x_1 + x_2)/\sqrt{2} \\ (x_1 - x_2) + 6/\sqrt{2} \\ (x_2 - x_1) + 6/\sqrt{2} \end{array} \right\} \tag{14}$$

where $\mathbf{x} = [x_1, x_2] \in [-5, 5]^2$. The two inputs are considered independent and historically have been subject to standard normal distribution. However here a uniformly distributed sample is assumed to reduce the concentration of initial training points in the failure domain. The GPE was trained with $n = 20$ LHS samples. GPSS with $\varepsilon = 10^{-4}$ was used with this function on a failure threshold, $y_c = 2$. At this level the associated probability of failure, $p_F \approx 7.44 \times 10^{-2}$. The challenge associated with this function is the complex limit state contour which has some nearly discontinuous points. The progression of the estimation of the surface via GPSS is shown in Figure 1. The coefficient of variation and relative error were estimated from 100 runs of SuS on the improved surface as $\delta_{p_F} = 4.2\%$ and $\Delta p_F = 1.5\%$, respectively. The estimated mean failure probability was $\bar{p}_F = 7.62 \times 10^{-2}$ based on the same 100 runs. A comparison between the CCDF from SuS and CCDF from GPSS is shown in Figure 2. The nature of the local quality of the surrogate is evident from the variations in the first potion of the curve. Indeed this is as intended, since there is no attempt to improve the surrogate globally.

### 5.2 Mixture of Gaussians

This function was created to test the robustness of the GPSS algorithm. The choice of a Gaussian function as constructive element is justified by the fact that this work is not trying to challenge the emulator itself, but to improve it such that it enables the accurate prediction of the characteristics of the failure domain.

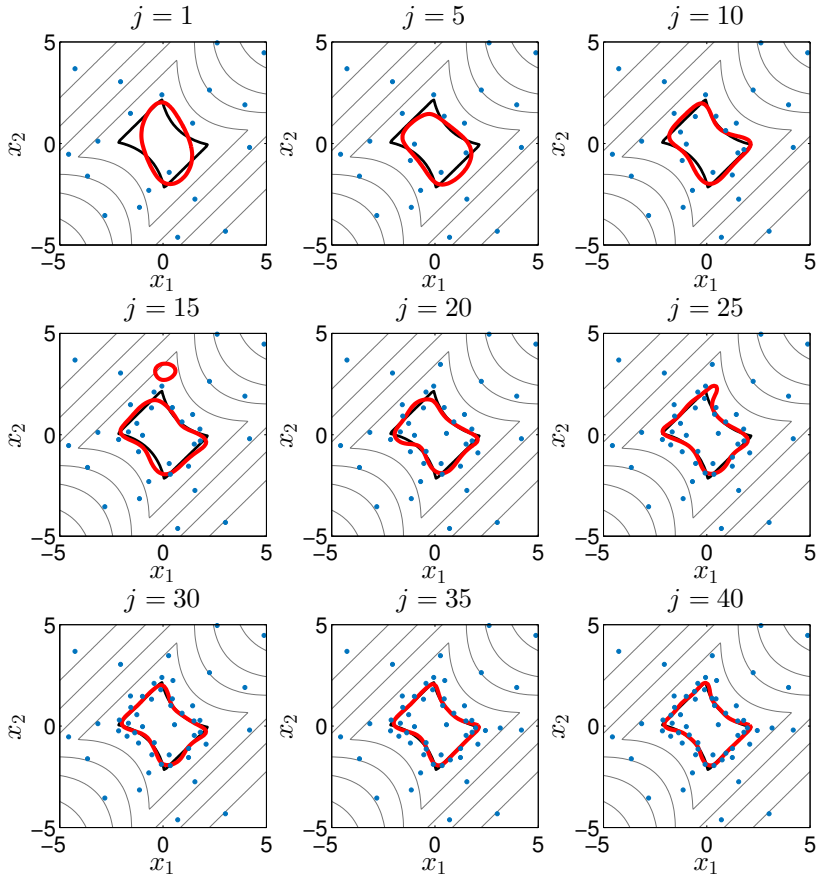


Figure 1: The performance of GPSS on the four branch series system. The true limit state contour, $y_c = 2$ is in black and that estimated by the GP - in red. The titles of the tiles show different stages of the algorithm. The number of samples in each tile is $n = \{20, 24, 29, 34, 40, 45, 50, 55, 60\}$.

The function has the form:

$$\eta(\mathbf{x}) = \frac{10^2}{2}\left[a\phi(\mathbf{x}_A) + b\phi(\mathbf{x}_B) + c\phi(\mathbf{x}_C) + d\phi(\mathbf{x}_D)\right] \tag{15}$$
$$\mathbf{x}_A = 10\,(\mathbf{x} - 1/4)$$
$$\mathbf{x}_B = 10\,(\mathbf{x} - 3/4)$$
$$\mathbf{x}_C = [10\,(x_1 - 3/4)\,, 10\,(x_2 - 3/4)]$$
$$\mathbf{x}_D = [10\,(x_1 - 1/3)\,, 10\,(x_2 - 5/6)]$$

In Eq. (15) $\mathbf{x} = [x_1, x_2] \in [0, 1]^2$ and $\phi(\cdot)$ is the standard normal PDF. The fractions in the expressions for $\mathbf{x}_A \ldots \mathbf{x}_D$ are location parameters and can be chosen arbitrarily. The constants $a$, $b$, $c$ and$d$ are used to select how many peaks belong to the failure domain. For this experiment $a = b = 1$, $c = 0.85$ and $d = 1.1$, meaning that peak $c$ is just outside of failure level set as $y_c = 7.9$. The associated failure probability is $p_F = 7.27 \times 10^{-3}$.

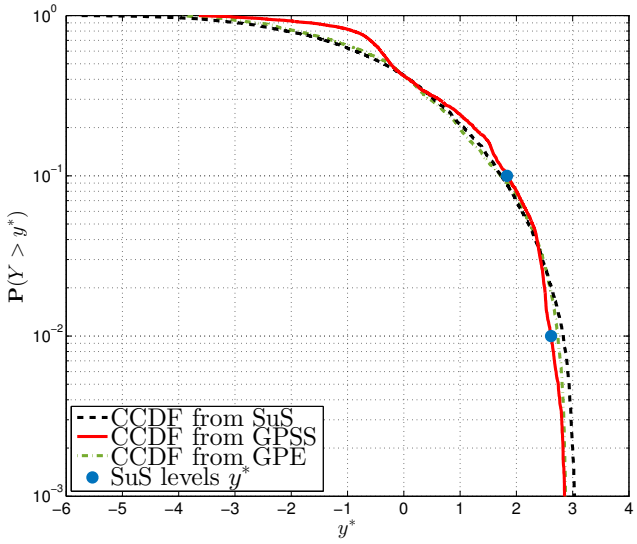


Figure 2: Complementary CDF estimates for the four branch system. Each curve is based of two levels of SuS (blue dots) and 3000 samples per level.

The GPE was initially trained with $n = 20$ LHS points. The variance threshold was set to, $\varepsilon = 10^{-5}$. Figure 3 shows the process of discovery of the failure domain. The plots in the first row show the contour values of the intermediate levels $y^{GP}_{c_{(j-1)}}$ in Eq. (11) (red line), because $y_c$ was not accessible at these GPE stages. During most of the process points were added in all three modes. However from level $j = 16$ to the end the algorithm could not improve the approximations in modes $a$ and $b$ any further and stopped sampling from them. This is reflected in the last row of Figure 3 where the green dots show the new samples in each consecutive frame - it can be seen that samples are only being added to mode $d$. This feature of the algorithm 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. As before, some statistics about the probability of failure were calculated from 100 runs of SuS. The mean 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 $\Delta p_F = 2.7\%$, respectively. It is interesting to note that the same figures based on a 100 runs of direct Monte Carlo simulation with $100000$ samples were $\delta^{DMC}_{p_F} = 3.7\%$ and $\Delta p^{DMC}_F = 0.7\%$. Figure 4 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 points from the GPE that belong to $F$.

### 5.3 10D wing weight model

The last experiment that was conducted is based on a high dimensional model for estimating the weight of the wing of a general aviation aircraft. The functional expression is:

$$0.036 S_w^{0.758} W_{fw}^{0.0035} \left( \frac{A}{\cos^2 \Lambda} \right)^{0.6} q^{0.006} \lambda^{0.04} \left( \frac{100tc}{\cos \Lambda} \right) (N_z W_{dg})^{0.49} + S_w W_p \tag{16}$$

The nomenclature of Eq. (16) is somewhat involved and can be found in [2]. For the purposes of this paper the dimensionality of the problem is of interest. GPSS was ran on the model with 5000 samples per level for SuS and variance threshold, $\varepsilon = 10^{-5}$. Since it was known a priori

that more points will be added to the training sample for the GPE, the initial DOE consisted of $60$ LHS points. The procedure took $108$ iterations with a final data point count of 168 samples.

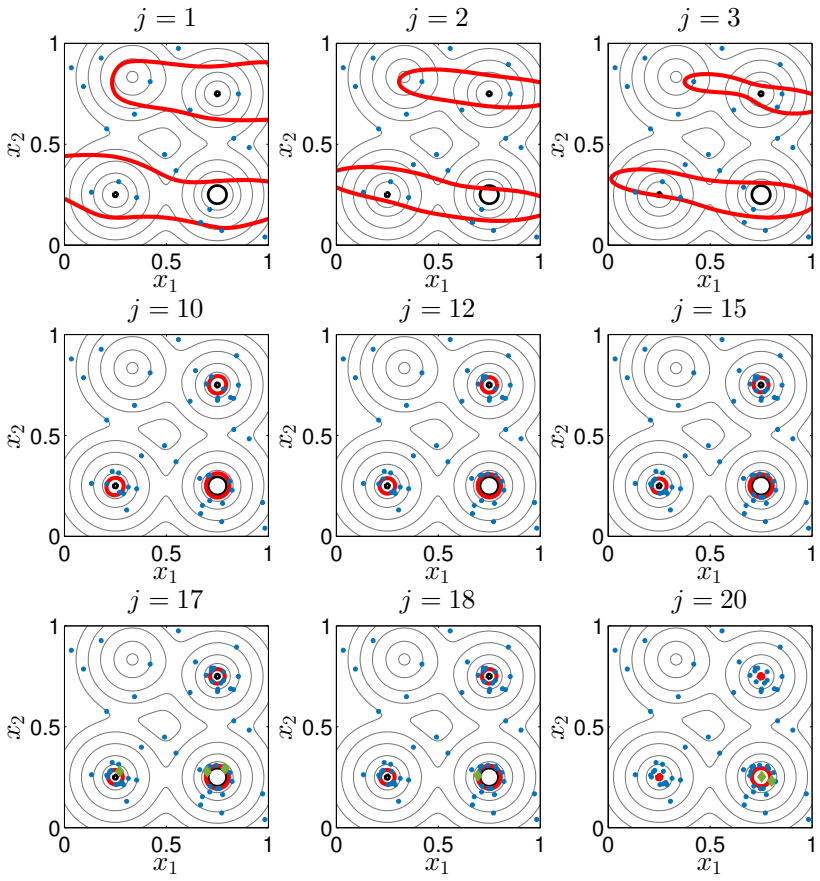


Figure 3: The performance of GPSS on the mixture function. The title in each tile shows the level of the algorithm. The red contours in the first row correspond to the intermediate levels from Eq. (11) at the specified level. The green diamonds in the last row show the samples added since the previous tile. The number of samples in each tile is $n = \{20, 22, 24, 44, 50, 59, 62, 63, 64\}$.

Figure 5 (left-centre) shows a set of diagnostics from the first and the last fit of the GPE for $100$ LHS data points. The left column shows the observed functional response versus predictions generated from the GPE. The error bars reflect the $95\%$ credible interval for each prediction. In the centre column the individual prediction errors (IPE) are plotted as suggested in [30]. The IPE should have a *student-t* distributions and should therefore lie in the interval $[-2, 2]$, $95\%$ of the time. There is not much of a difference between the two sets of plots which suggests that the global quality of the surrogate has not changed significantly - the GPE is a good global predictor before and after GPSS. The same cannot be said about the local approximation of the failure region.

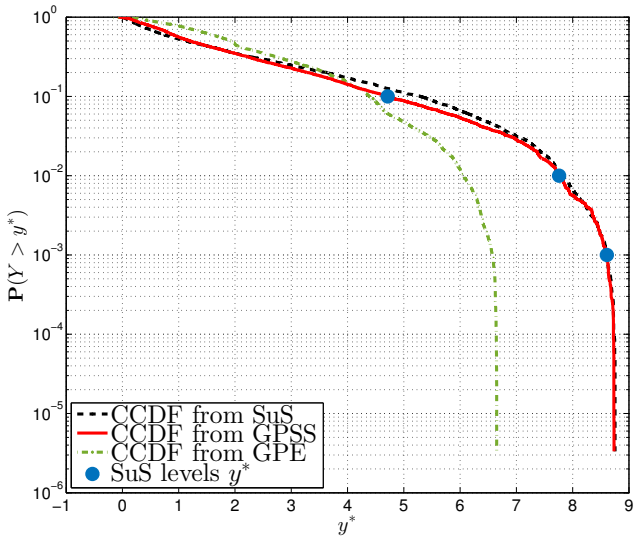


Figure 4: Complementary CDF of the mixture function according to the GPE (green), GPSS (red) and SuS (black). The blue circles show the intermediate levels of SuS. Each level was populated with 3000 samples.

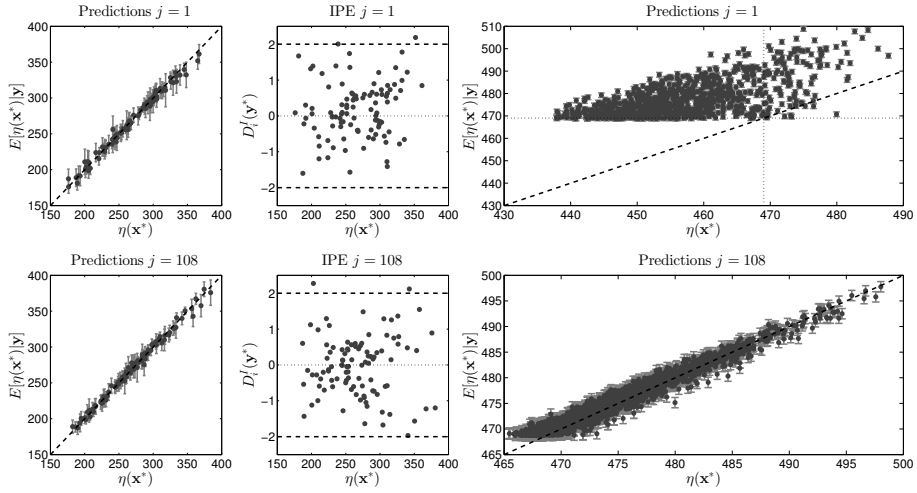


Figure 5: Diagnostics for the quality of the GPE before (top left-centre) and after (bottom left-centre) GPSS. The correlation between observation and prediction at 100 data points is shown on the left. The distribution of IPE is shown in the centre. The correlation between observation and prediction at points originally lying in $F$ before and after GPSS, top right and bottom right, respectively. In all cases mean and $95\%$ credible interval are given in dark grey dots and light grey error bars, respectively. Dashed lines on top mark $F$ on both axes.

Figure 5 (right) shows a comparison between the prediction of function response, originally in the failure domain, before (top) and after (bottom) GPSS. Results are obvious - the correlation between observed and predicted values is much higher. The mean probability of failure was $\bar{p}_F = 1.97 \times 10^{-5}$. The corresponding c.o.v. and relative error were $\delta_{p_F} = 21.8\%$ and $\Delta p_F = 11.4\%$, respectively. These were calculated based on 100 runs of SuS. Finally, Figure 6 shows a comparison between the complementary CDF curves from the GPE (green), GPSS (red) and

SuS (black).

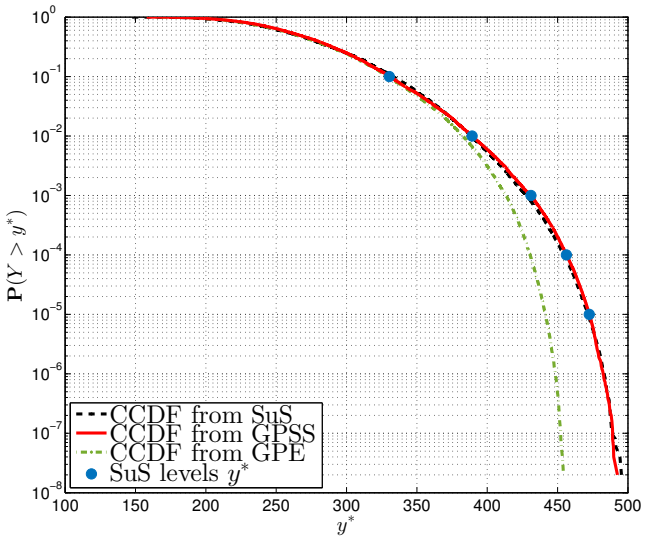


Figure 6: Complementary CDF of the mixture function according to the GPE (green), GPSS (red) and SuS (black). The blue circles show the intermediate levels of SuS. Each level was populated with 5000 samples. The global quality of the GPE before GPSS is good only in the high probability regions.

The CCDF curve from the first GPE level summarizes the diagnostics from Figure 5 - the original surrogate provides a good approximation in the high probability regions of the performance function, but fails to capture its behaviour in the rare regions.

## 6 CONCLUSIONS

In this paper an adaptive algorithm for reliability analysis, called GPSS, was presented. GPSS is a combination between Gaussian process emulation and subset simulation which allows an efficient characterization of rare events and estimation of the associated probabilities of failure. The reduction in the number of samples require by GPSS as compared to SuS and DMC is significant. The algorithm was validated with three benchmark problems. Future work, as planned by the authors includes reliability analysis of computationally expensive computer codes, where the advantages offered by GPSS can be appreciated.

## REFERENCES

[1] S. K. Au and J. L. Beck, "Estimation of small failure probabilities in high dimensions by subset simulation," *Probabilistic Engineering Mechanics*, vol. 16, no. 4, pp. 263–277, 2001.

[2] A. Forrester, A. Sobester, and A. Keane, *Engineering Design via Surrogate Modelling: A Practical Guide*. Wiley, 2008.

[3] P. Ranjan, D. Bingham, and G. Michailidis, "Sequential Experiment Design for Contour Estimation From Complex Computer Codes," *Technometrics*, vol. 50, no. 4, pp. 527–541, 2008.

[4] V. Picheny, D. Ginsbourger, O. Roustant, R. T. Haftka, and N.-H. Kim, "Adaptive Designs of Experiments for Accurate Approximation of a Target Region," *Journal of Mechanical Design*, vol. 132, no. 7, p. 071008, 2010.

[5] Z. Zhu and X. Du, "Reliability Analysis With Monte Carlo Simulation and Dependent Kriging Predictions," *Journal of Mechanical Design*, vol. 138, no. 12, p. 121403, 2016.

[6] T. H. Lee and J. J. Jung, "A sampling technique enhancing accuracy and efficiency of metamodel-based RBDO: Constraint boundary sampling," *Computers and Structures*, vol. 86, no. 13-14, pp. 1463–1476, 2008.

[7] V. Dubourg, B. Sudret, and J. M. Bourinet, "Reliability-based design optimization using kriging surrogates and subset simulation," *Structural and Multidisciplinary Optimization*, vol. 44, no. 5, pp. 673–690, 2011.

[8] G. Su, B. Yu, Y. Xiao, and L. Yan, "Gaussian Process Machine-Learning Method for Structural Reliability Analysis," *Advances in Structural Engineering*, vol. 17, no. 9, pp. 1257–1271, 2014.

[9] L. Zhang, Z. Lu, and P. Wang, "Efficient structural reliability analysis method based on advanced Kriging model," *Applied Mathematical Modelling*, vol. 39, no. 2, pp. 781–793, 2015.

[10] J. Bect, D. Ginsbourger, L. Li, V. Picheny, and E. Vazquez, "Sequential design of computer experiments for the estimation of a probability of failure," *Statistics and Computing*, vol. 22, no. 3, pp. 773–793, 2012.

[11] D. Jones, M. Schonlau, and W. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[12] B. Bichon, S. Mahadevan, and M. Eldred, "Reliability-Based Design Optimization Using Efficient Global Reliability Analysis," *50th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, no. May, p. 2261, 2009.

[13] B. Echard, N. Gayton, and M. Lemaire, "AK-MCS: An active learning reliability method combining Kriging and Monte Carlo Simulation," *Structural Safety*, vol. 33, no. 2, pp. 145–154, 2011.

[14] Z. Wen, H. Pei, H. Liu, and Z. Yue, "A Sequential Kriging reliability analysis method with characteristics of adaptive sampling regions and parallelizability," *Reliability Engineering and System Safety*, vol. 153, pp. 170–179, 2016.

[15] X. Huang, J. Chen, and H. Zhu, "Assessing small failure probabilities by AK-SS: An active learning method combining Kriging and Subset Simulation," *Structural Safety*, vol. 59, pp. 86–95, 2016.

[16] C. Tong, Z. Sun, Q. Zhao, Q. Wang, and S. Wang, "A hybrid algorithm for reliability analysis combining Kriging and subset simulation importance sampling," *Journal of Mechanical Science and Technology*, vol. 29, no. 8, pp. 3183–3193, 2015.

[17] Z. Sun, J. Wang, R. Li, and C. Tong, "LIF: A new Kriging based learning function and its application to structural reliability analysis," *Reliability Engineering & System Safety*, vol. 157, pp. 152–165, 2017.

[18] X. Yang, Y. Liu, and L. Gao, "Unified reliability analysis by active learning Kriging model combining with random-set based Monte Carlo simulation method," *Int. J. Numer. Meth. Engng*, vol. 108, pp. 1343—-1361, 2016.

[19] N. Gayton, J. Bourinet, and M. Lemaire, "CQ2RS: a new statistical approach to the response surface method for reliability analysis," *Structural Safety*, vol. 25, no. 1, pp. 99–121, 2003.

[20] S. Gupta and C. S. Manohar, "An improved response surface method for the determination of failure probability and importance measures," *Structural Safety*, vol. 26, no. 2, pp. 123–139, 2004.

[21] S. K. Au, "Augmenting approximate solutions for consistent reliability analysis," *Probabilistic Engineering Mechanics*, vol. 22, no. 1, pp. 77–87, 2007.

[22] D. Q. Li, K. B. Shao, Z. J. Cao, X. S. Tang, and K. K. Phoon, "A generalized surrogate response aided-subset simulation approach for efficient geotechnical reliability-based design," *Computers and Geotechnics*, vol. 74, pp. 88–101, 2016.

[23] V. Papadopoulos, D. G. Giovanis, N. D. Lagaros, and M. Papadrakakis, "Accelerated subset simulation with neural networks for reliability analysis," *Computer Methods in Applied Mechanics and Engineering*, vol. 223-224, pp. 70–80, 2012.

[24] J. M. Bourinet, F. Deheeger, and M. Lemaire, "Assessing small failure probabilities by combined subset simulation and Support Vector Machines," *Structural Safety*, vol. 33, no. 6, pp. 343–353, 2011.

[25] S. K. Au and Y. Wang, *Engineering Risk Assessment with Subset Simulation*. Wiley, 2014.

[26] A. Keane and P. Nair, *Computational Approaches for Aerospace Design*. Derby: John Wiley & Sons, 2005.

[27] J. Oakley, "Bayesian uncertainty analysis for complex computer codes," *Thesis*, 1999.

[28] W. Becker, *Uncertainty Propagation through Dependability Models*. PhD thesis, 2011.

[29] M. D. McKay, R. J. Beckman, and W. J. Conover, "Comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, no. 2, pp. 239–245, 1979.

[30] L. S. Bastos and A. O'Hagan, "Diagnostics for Gaussian Process Emulators," *Technometrics*, vol. 51, no. 4, pp. 425–438, 2009.

[31] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in *KDD-96 AIAA*, pp. 226–231, 1996.