Implementation of the AK-MCS algorithm for structural reliability

by
Juan Camilo Lugo Rojas

Advisor
Dr. techn. Diego Andrés Alvarez Marín

Submitted in partial fulfillment of the requirements
for the degree of

Civil Engineer

Department of Civil Engineering
Faculty of Engineering and Architecture
Universidad Nacional de Colombia, Sede Manizales

February, 2022

To those who have been there.

Abstract

A study of the AK-MCS method is presented, which seeks to optimize the determination of failure probabilities by means of Monte Carlo Simulations. A review of key concepts of structural reliability is made, in order to contextualize the field of application of the studied algorithm. A theoretical review of the Gaussian Process Regression is made to be able to understand the AK-MCS algorithm. Finally, several models are implemented, comparing the performance of AK-MCS with that of classical Monte Carlo Simulations.

Keywords: Monte Carlo Simulation, Structural Reliability Analysis, Krigin Regression, Active learning methods.

Resumen

Se presenta un estudio del método AK-MCS, el cual busca optimizar la determinación de probabilidades de falla mediante las Simulaciones de Monte Carlo. Se hace una revisión de conceptos clave de confiabilidad estructural, con el fin de contextualizar el campo de aplicación del algoritmo estudiado. Para poder entender AK-MCS, se hace una revisión teórica de las regresiones mediante procesos gaussianos. Finalmente, se implementan varios modelos, comparando el desempeño del algoritmo estudiado, con el de las simulaciones de Monte Carlo clásicas.

Palabras clave: Simulación de Monte Carlo, Análisis de confiabilidad estructural, Regresión Kriging, Métodos de aprendizaje activo.

Contents

Chapter 1

Introduction

The study of the behavior of a structure given the properties of its materials, its geometric configuration, and the loads to which it is subjected is known as structural analysis. Within this field, it is of particular interest to determine when a structure exceeds certain levels of distress that can lead to collapse, considerably affect its performance, or affect its serviceability, among other criteria. These are known as limit states [Melchers and Beck, 2018].

The determination of whether these limits have been exceeded has been addressed from different perspectives throughout history. When the variables involved are considered to take unique values, we have what is known as the deterministic approach. There is a natural extension of the latter, which consists of considering that the values of these variables present variations derived from their intrinsic uncertainty, this being known as the probabilistic approach [Ditlevsen and Madsen, 1996].

The term reliability is commonly defined as the complement of the probability of failure (failure being understood as the exceeding of a predefined limit state). In other words, reliability corresponds to the probability that the structure will function properly during a period of time [Melchers and Beck, 2018].

The use of reliability methods in structural engineering has several advantages, as stated in [Lemaire, 2009]. Among them, there are:

- offer a more realistic processing of uncertainties, allowing for a better understanding of these in the response of the structures,
- allow an optimal distribution of materials and resources in each component as well as in the entire structure,
- permit the evaluation of the regulations established in building codes

(e.g. safety factors), which can sometimes appear arbitrary due to the ignorance of the uncertainties associated with the phenomena

As a result, reliability analysis methods are being used more and more, given the stringent performance requirements, safety margins, and competitiveness of the market. They are particularly useful when dealing with unconventional structures for which there is insufficient empirical information. [Choi et al., 2006].

Several methods have been developed to estimate reliability, based on different principles. Each of them has advantages and disadvantages according to the nature of the problem, the number of variables involved and how they interact with each other, among other aspects. One of the most robust methods for reliability estimation is the Monte Carlo Simulation (MCS). The concept behind MCS is relatively simple, and it is a very powerful method, but its main drawback is that it can become computationally expensive [Nowak and Collins, 2000].

Various methods have been derived from MCS, trying to overcome the problem of its computational cost, while preserving its strengths. In this document we study one of those methods, known as AK-MCS for Active learning reliability method combining Kriging and MCS, which attempts to obtain results equivalent to those given by MCS in a more efficient manner [Echard et al., 2011].

1.1 *Objectives*

The main objective of this final project is to implement the algorithm proposed by [Echard et al., 2011] that employs MCS and a Kriging metamodel to asses structural reliability. In order to satisfactorily do this, many specific objectives have to be achieved, which include:

- Learn the Monte Carlo Simulation method.
- Learn the Kriging regression algorithm.
- Completely understand the article.
- Implement the previously mentioned methods in Python.
- Review articles that aim to improve the performance of the AK-MCS algorithm.
- Evaluate the performance of the algorithm, comparing it with the basic MCS.

Chapter 2

Some concepts of Structural Reliability

This chapter defines the concept of probability of failure and how it is calculated, and briefly reviews some methods to contextualize the subsequent work. It is based mostly on [Hurtado, 2013] and [Melchers and Beck, 2018].

2.1 Reliability problem formulation

The reliability problem is based on the performance function $G(\boldsymbol{x})$ defined as:

$$G(\boldsymbol{x}) = R(\boldsymbol{x}) - S(\boldsymbol{x}) \tag{2.1}$$

where $\boldsymbol{x}$ is the vector of input variables, in which are included the geometric and material properties, as well as the acting loads, and in general all the variables taken into consideration that affect the behavior of the structure; $S(\boldsymbol{x})$ is a load effect acting on the structure, and $R(\boldsymbol{x})$ is its capacity to withstand it. The limit state equation corresponds to $G(\boldsymbol{x}) = 0$, i.e. when the acting and resisting forces are equal. A violation of the limite state occurs when $G(\boldsymbol{x}) \leq 0$, otherwise (when $G(\boldsymbol{x}) > 0$), we say that the structure is safe.

For complex problems, the performance function usually cannot be expressed as in 2.1, but in the form:

$$G(\boldsymbol{x}) = r(\boldsymbol{x}) - \hat{r} \tag{2.2}$$

where $r(\boldsymbol{x})$ is an implicit function and $\hat{r}$ is a critical threshold. $r(\boldsymbol{x})$ can be, for instance, the result obtained from a finite element analysis.

The objective is to determine the probability of failure p_f of the structure:

$$p_f = P[G(\boldsymbol{x}) \leq 0] \tag{2.3}$$

Example: Cantilever beam

Consider the beam shown in figure 2.1, with rectangular cross section of base b and height h. A performance function can be defined as follows:

$$G(\boldsymbol{x}) = R - \underbrace{\frac{6PL}{bh^2}}_{\sigma_b} \tag{2.4}$$

where σ_b is the maximum bending stress to which the beam is subjected and R is the beam resistance. Then we have that if $G(x) \leq 0$, i.e. if $R \leq \sigma_b$, the beam will fail according to this criterion.

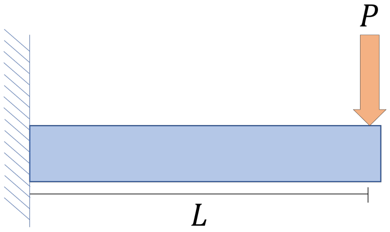


Figure 2.1: Cantilever beam.

2.2 Realiability assessment approaches

2.2.1 Deterministic approach

The deterministic approach is the traditional one. In this approach, the uncertainties in the forces and resistances are covered by assuming that the variation of these forces and resistances have limits that are not exceeded. Given this simplicity, it is usually necessary to overestimate the magnitudes in order to provide greater safety. This is usual in design codes when considering safety factors in the limit state equations. It is widely used, especially in simple and common problems. However, this required safety margin implies the need to assume higher costs.

In this case, the result of the formula 2.3 is either 1 or 0, since the values of the variables are considered fixed, thus the inequality is either fulfilled or not.

Example: Cantilever beam (cont.)

In order to design the beam in such a way that it does not violate the limit state equation considering this approach, design parameters are selected such that $G(\boldsymbol{x}) > 0$. For safety reasons, in the equation 2.4, σ_b is replaced by $\hat{\sigma}_b = \lambda\sigma_b$, being $\lambda \geq 1$ the safety factor, which is determined empirically based on previous experience with similar problems.

Although the limit state is not violated, failure may still occur, due to the uncertainty that is ignored.

2.2.2 *Partial probabilistic approach*

The historical record of events that expose structures to large loads, such as earthquakes, floods, tornadoes, etc., shows that they have a certain frequency according to their magnitude. For instance, during one day there may be multiple low-scale earthquakes, but every few years there is one of considerable destructive power.

The partially probabilistic approach handles these return periods when considering the maximum loads to which a structure will be subjected during certain period of time. This allows the selection of design parameters in accordance with the expected service life of the structure, seeking a balance between the probability of failure and the costs that may be incurred.

It can be seen that some randomness is considered when accounting for the variability over time of the loads; however, this approeach does not acknowledge that even at a given instant there is uncertainty.

> **Example: Cantilever beam (cont.)**
> In this case, to estimate $\hat{\sigma}_b$, the historical records of the values taken by the applied load P are considered to study the frequency of events in which it takes very high values. Then a value of P is selected, attempting to maintain a good balance between the costs and the risk of failure.
>
> There is a return period associated with the magnitude of P chosen, which accounts for the probability that the structure will fail during a certain period of time.

2.2.3 *Probabilistic approach*

The variables involved in the performance function can be considered random. For this purpose, through statistical data and experience, the distribution that each of them follows is determined.

Let consider the case when $G(\boldsymbol{x})$ can be expressed as in 2.1. It is possible to obtain probability density functions $f_R(x)$ and $f_S(x)$ that describe R and S respectively. Since the limit state is violated when $G(\boldsymbol{x}) \leq 0$, the probability of failure can be roughly represented by the overlap of $f_R(\boldsymbol{x})$ and $f_S(x)$ as shown in figure 2.2.

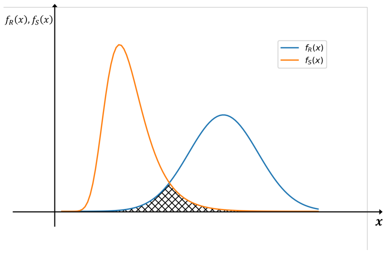


Figure 2.2: Representation of $p_f = P\left[R(\boldsymbol{x}) \leq S(\boldsymbol{x})\right]$

When $f_R(x)$ and $f_S(x)$ are independent, the probability of failure can be expressed as:

$$p_f = \int_{-\infty}^{\infty} F_R(x) f_S(x) \mathrm{d}x \tag{2.5}$$

which is known as a convolution integral, where $F_R(x)$ is the probability that $R \leq x$, so by considering its product with $f_S(x)$ over all possible values of x, p_f is obtained.

In the general case, the probability of failure is given by:

$$p_f = \underset{G(\boldsymbol{x}) \leq 0}{\int \cdots \int} f_{\boldsymbol{x}}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \tag{2.6}$$

where $f_{\boldsymbol{x}}(\boldsymbol{x})$ is the joint probability density function of the considered variables $\boldsymbol{x}$. The solution of 2.6 can be tricky. For this purpose, several methods have been developed, some of which will be reviewed in the following section.

2.3 Reliabiliy assessment methods

The analysis methods used to estimate 2.6 can be divided into two main categories: analytic and synthetic. The former are based on the Taylor series expansion of the limit state function, while the latter require the generation of random samples of the variables associated with the model. In this section we will review some generalities of these two categories, their applicability and limitations, with a particular focus on the latter, to which the method reviewed in this document corresponds.

2.3.1 Analytic methods (FORM and SORM)

The *First* and *Second Order Reliability Methods* require the transformation of the vector $\boldsymbol{x}$, defined by their joint density function $f_{\boldsymbol{x}}$, into a set of normally distributed independent variables $\boldsymbol{u}$. The Taylor series expansion of $G(\boldsymbol{u})$ is considered, of first order when using FORM, and of second order in the case of SORM. Iteratively, it is obtained the denominated design point $\boldsymbol{u}^*$ by minimizing its distance from the origin such that $G(\boldsymbol{u}^*) = 0$. By considering the Taylor series expansion at the design point, a hyperplane is obtained, such that it divides the design space (the possible values taken by the input variables) into the failure and the safe domain.

As these methods require the calculation of the gradient of the performance function, in addition to the Hessian matrix in the case of SORM, the practicality of their application depends heavily on the form of the performance matrix, greatly increasing the complexity of the calculations when the dimensionality of the problem increases.

2.3.2 *Synthetic methods (Monte Carlo Simulation)*

Reliability analysis using MCS consists of generating samples in the design space that are used to evaluate the structural model and thus numerically estimate the probability of failure.

To generate the sample, random numbers are first generated. Several algorithms have been developed for this purpose, seeking to generate samples uniformly distributed between 0 and 1. Nowadays, the most widely used algorithm is the Mersenne Twister (see [Matsumoto and Nishimura, 1998]). Starting from the uniformly distributed numbers, the sample is generated following the given distributions. The most general technique for this is the inverse transform method. For some particular distributions there are more efficient methods, for example, for a normal distribution the Box and Muller algorithm (see [Box and Muller, 1958]) can be used.

The equation 2.6 can be written as:

$$p_f = \int \cdots \int I\left[G(\boldsymbol{x}) \leq 0\right] f_{\boldsymbol{x}}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} \tag{2.7}$$

where $I\left[G(\boldsymbol{x}) \leq 0\right]$ is an *indicator function* defined as:

$$I\left[G(\boldsymbol{x}) \leq 0\right] = \begin{cases} 1, & \text{if } G(\boldsymbol{x}) \leq 0 \\ 0, & \text{if } G(\boldsymbol{x}) > 0 \end{cases} \tag{2.8}$$

But equation 2.7 corresponds to the expected value of $I\left[G(\boldsymbol{x}) \leq 0\right]$, so considering a generated sample of n_{MC} elements, being $\hat{\boldsymbol{x}}_j$ an element of the sample, it follows that the probability of failure can be approximated as:

$$p_f \approx \frac{1}{n_{MC}} \sum_{j=1}^{n_{MC}} I\left[G(\hat{\boldsymbol{x}}_j) \leq 0\right] \tag{2.9}$$

This estimated value of p_f, since it depends on a random sample, is itself a random variable. A measure of how much its value can vary is the coefficient of variation, which is defined as:

$$\text{C.O.V}_{p_f} = \sqrt{\frac{1 - p_f}{p_f \, n_{MC}}} \tag{2.10}$$

Note that the C.O.V is smaller when the p_f is larger, as the relative variation it can have is smaller; and it can also be reduced by using a larger number of samples.

Among the main advantages of MCS is that it makes no assumptions about the limit state function, so it is not necessary to perform any kind of transformation to the design space. In addition, it does not present as many drawbacks as other methods when dealing with high dimensionality problems. However, its main drawback is its computational cost. When having very complex limit state functions, it is necessary to evaluate it too many times to obtain an accurate enough result. Variations of the method are continuously being developed to deal with this problem.The AK-MCS algorithm, which is explained in the next chapter, is one of these variations.

Chapter 3

The AK-MCS method

One of the techniques used to deal with the high computational cost of MCS is to try to approximate the value taken by the performance function at the points of the sample, thus avoiding the costly task of evaluating this function at every point. AK-MCS, which stands for an active learning reliability method combining Kriging and Monte Carlo Simulation, is a method that performs such estimation by using a Gaussian Process Regression. The method is exposed in [Echard et al., 2011], on which this chapter is mostly based, together with [Rasmussen and Williams, 2006].

In this chapter, the procedure for performing regressions with Gaussian processes is first reviewed, then the steps to be followed during the method are described, and finally some application examples are shown.

3.1 Preliminaries

3.1.1 Gaussian Identity of conditional distributions

The joint probability density of the multivariate Gaussian distribution is given by:

$$p(\boldsymbol{x}|\boldsymbol{m},\boldsymbol{\Sigma}) = (2\pi)^{D/2}|\boldsymbol{\Sigma}|^{-1/2}\exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{m})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{m})\right) \quad (3.1)$$

where $\boldsymbol{x}$ is a vector of length D, $\boldsymbol{m}$ is the mean vector, and $\boldsymbol{\Sigma}$ is the covariance matrix, which is symmetric and positive definite. To say that $\boldsymbol{x}$ follows a Gaussian distribution with mean $\boldsymbol{m}$ and covariance $\boldsymbol{\Sigma}$ one writes $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{m},\boldsymbol{\Sigma})$.

Let $\boldsymbol{x}$ and $\boldsymbol{y}$ be jointly Gaussian random vectors

$$\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_x \\ \boldsymbol{\mu}_y \end{bmatrix}, \begin{bmatrix} \boldsymbol{A} & \boldsymbol{C} \\ \boldsymbol{C}^T & \boldsymbol{B} \end{bmatrix} \right) \tag{3.2}$$

then the conditional distribution of $\boldsymbol{x}$ given $\boldsymbol{y}$ is:

$$\boldsymbol{x}|\boldsymbol{y} \sim \mathcal{N}\left(\boldsymbol{\mu}_x + \boldsymbol{C}\boldsymbol{B}^{-1}(\boldsymbol{y} - \boldsymbol{\mu}_y),\ \boldsymbol{A} - \boldsymbol{C}\boldsymbol{B}^{-1}\boldsymbol{C}^T \right) \tag{3.3}$$

The proof of this fact can be detailed in [Bishop, 2006], section 3.1.

3.1.2 Gaussian Process regression (or Kriging)

A Gaussian process, is a collection of random variables, any finite number of which have a joint Gaussian distribution. It is completely specified by its mean function and covariance function. To say that a real process $f(\boldsymbol{x})$ is Gaussian with mean function $m(\boldsymbol{x})$ and covariance function $k(\boldsymbol{x}, \boldsymbol{x'})$ one writes:

$$f(\boldsymbol{x}) \sim \mathcal{GP}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x'})) \tag{3.4}$$

that is, every value of $f(\boldsymbol{x})$ at a location $\boldsymbol{x}$ is a random variable that follows a Gaussian distribution. Usually, for simplicity of notation, the mean function is considered to be zero.

Gaussian processes can be used to make predictions. Let $(\boldsymbol{X}, \mathbf{f}) = \{(\boldsymbol{x}_i, \mathrm{f}_i) | i = 1, \ldots, n\}$ be a set of observations (or training data) and $\boldsymbol{X}_*$ a set of n_* input values for which we have to determine the value of f_* (or test data). Moreover, suppose that the observations f are noisy, i.e. the exact value of f is not known but that of $\boldsymbol{y} = \mathbf{f} + \boldsymbol{\epsilon}$. Assuming that the noise $\boldsymbol{\epsilon}$ is additive independent identically Gaussian distributed with variance σ_n^2, the covariance on the observations becomes

$$\mathrm{cov}(y_p, y_q) = k(\boldsymbol{x}_p, \boldsymbol{x}_q) + \sigma_n^2 \delta_{pq} \tag{3.5}$$

where δ_{pq} is a Kronecker delta which is 1 when $p = 1$ and 0 otherwise. Then one can write the joint distribution as:

$$\begin{bmatrix} \boldsymbol{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}) + \sigma_n^2 \boldsymbol{I} & \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}_*) \\ \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}) & \boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}_*) \end{bmatrix} \right) \tag{3.6}$$

where $\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X}_*)$ denotes the $n \times n_*$ matrix of the covariances evaluated at all pairs of training and test points, and similarly for $\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})$, $\boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X})$ and $\boldsymbol{K}(\boldsymbol{X}_*, \boldsymbol{X}_*)$.

According to 3.3, given that the distribution of $\boldsymbol{y}$ is known, the distribution of $\mathbf{f}_*$ can be calcutated as follows:

$$\mathbf{f}_*|\boldsymbol{X},\boldsymbol{y},\boldsymbol{X}_* \sim \mathcal{N}(\bar{\mathbf{f}}_*, \operatorname{cov}(\mathbf{f}_*)) \tag{3.7}$$

where

$$\bar{\mathbf{f}}_* = \boldsymbol{K}(\boldsymbol{X}_*,\boldsymbol{X})\left[\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}) + \sigma_n^2\boldsymbol{I}\right]^{-1}\boldsymbol{y} \tag{3.8}$$

$$\operatorname{cov}(\mathbf{f}_*) = \boldsymbol{K}(\boldsymbol{X}_*,\boldsymbol{X}_*) - \boldsymbol{K}(\boldsymbol{X}_*,\boldsymbol{X})\left[\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}) + \sigma_n^2\boldsymbol{I}\right]^{-1}\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}_*) \tag{3.9}$$

Note that the values of $\bar{\mathbf{f}}_*$ are linear combinations of the observations; this is called a linear predictor. Also, note that the variances $\operatorname{cov}(\mathbf{f}_*)$ do not depend on the observations; it is the difference between two terms: the first term $\boldsymbol{K}(\boldsymbol{X}_*,\boldsymbol{X}_*)$ is the covariance given by the predifined function $k(\boldsymbol{x},\boldsymbol{x}')$, and from this is subtracted a positive term representing the information that the observations give about the function.

The function $k(\boldsymbol{x},\boldsymbol{x}')$, which gives the ratio of the closeness between two points, characterizes the regrssion model. A usual choice is the Radial-basis function, also known as squared-exponential, defined as:

$$k(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2l^2}\right) \tag{3.10}$$

with the parameter l being the characteristic length-scale. The value of this parameter is chosen by maximizing the log marginal likelihood given by:

$$\log p(\mathbf{f}|\boldsymbol{X}) = -\frac{1}{2}\boldsymbol{y}^T(\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}) + \sigma_n^2\boldsymbol{I})^{-1}\boldsymbol{y} - \frac{1}{2}\log|\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}) + \sigma_n^2\boldsymbol{I}| - \frac{n}{2}\log 2\pi \tag{3.11}$$

3.2 *Step-by-step of the method*

The AK-MCS method consists of the following stages:

1. **Generation of a Monte Carlo population in the design space.** According to the involved random variables, this population named S of n_{MC} points is generated.

2. **Definition of the initial design of experiments (DoE).** The initial DoE consists of a random selection of N_1 points of S. It is preferred to be small, adding at each iteration only the point that improves the metamodel the most. It is suggested to use a dozen points.

3. **Computation of the Kriging model.** The Kriging regressor is trained with the performance function G evaluated on the initial DoE. A model with an squared-exponential kernel is used.

4. **Prediction by Kriging and estimation of the probability of failure.** Predictions of the performance function over the Monte Carlo population, $\widehat{G}(x_i)$ for $i = 1, ..., n_{MC}$ are made with the previously trained Kriging model. Then, the estimated probability of failure $\widehat{p_f}$ is obtained by calculating the ratio of the points $x_i \in S$ such that $\widehat{G}(x_i) \leq 0$, i.e.:

$$\widehat{p_f} = \frac{n_{\widehat{G}(x_i) \leq 0}}{n_{MC}} \tag{3.12}$$

5. **Identification of the best next point in S to be evaluated on the performance function.** At this stage, the learning function is computed for each point of S to determine the next point that should be added to the DoE to improve the most the metamodel.

6. **Evaluation of the stopping condition.** The stopping condition associated to the learning function is evaluated for the point selected in the previous stage. If the criterion is met, we skip to Stage 8. Otherwise, we continue with Stage 7.

7. **Update of the design of experiments.** The point identified at stage 5 is added to the current DoE, such that $N_{i+1} = N_i + 1$. Then, the method goes back to Stage 3.

8. **Computation of the coefficient of variation of the probability of failure** If the stopping condition is satisfied, the metamodel is said to be accurate enough on the sign of the performance function on S. Subsequently, it is checked if S is large enough to obtain a low coefficient of variation on the estimation of p_f. If the coefficient of variation given by 2.10 is lesser than 5%, AK-MCS stops and the last estimation of the probability of failure is considered as the result. In other case, it continue to Stage 9.

9. **Update of the population.** S is increased with other n_{MC} points from the design space generated in the same way that in Stage 1, and the method proceeds back to stage 3.

3.3 *Learning Functions*

The learning functions are used to decide the next point of the population to be included in the DoE. In order to do that, they give a score to each x_i of S according to the value $\widehat{G}(x_i)$ and its variance $\sigma^2_{\widehat{G}}(x_i)$ given by the Kriging regressor. Every learning function comes along with a learning criterion and with a stopping condition on the obtained scores.

In general terms, since what is really important for the estimation of the failure probability is whether the limit state is violated or not, the learning functions prioritize the points close to the limit state that have a high variability.

3.3.1 *Expected feasibility function (EFF)*

Proposed in [Bichon et al., 2008]. It provides an indication of how well the true value of $\widehat{G}(x)$ is expected to satisfy the equality $G(x) = a$. In AK-MCS we have $a = 0$ and $\epsilon = 2\sigma_{\widehat{G}}(x)$. The learning criterion of $EFF(x)$ is the maximum value, so the best next point to add to DoE is $x^* \in S$ such that $EFF(x^*) = \max(EFF(\boldsymbol{x}))$. The stopping condition is $EFF(x^*) \leq 0.001$.

It is defined as:

$$\begin{aligned} EFF(x) = \left(\widehat{G}(x) - a\right) & \left(2\Phi(C) - \Phi(C^+) - \Phi(C^-)\right) \\ -\sigma_{\widehat{G}}(x) & \left(2\phi(C) - \phi(C^+) - \phi(C^-)\right) \\ +\epsilon & \left(\Phi(C^+) + \Phi(C^{)} - \right) \end{aligned} \tag{3.13}$$

where

$$\begin{aligned} C =& \frac{a - \widehat{G}(x)}{\sigma_{\widehat{G}}(x)} \\ C^+ =& \frac{(a+\epsilon) - \widehat{G}(x)}{\sigma_{\widehat{G}}(x)} \\ C^- =& \frac{(a-\epsilon) - \widehat{G}(x)}{\sigma_{\widehat{G}}(x)} \end{aligned}$$

and Φ is the standard normal cumulative distribution and ϕ the standard normal density distribution.

3.3.2 *Learning function U*

Proposed in [Echard et al., 2011]. Considering that in MCS only the sign of the performance function is important, this function selects as the next best point of S to be added to the DoE the one that has the higher potential risk of crossing the separator $\widehat{G}(x) = 0$. Then, the best next point to add to the DoE is $x^* \in S$ such that $U(x^*) = \min(U(\boldsymbol{x}))$. The stopping condition is $U(x^*) \geq 2$.

It is defined as:

$$U(x) = \frac{\left|\widehat{G}(x)\right|}{\sigma_{\widehat{G}}(x)} \tag{3.14}$$

As can be noted, $U(x^*)$ indicates the distance in standard deviations $\sigma_{\widehat{G}}(x)$ between $\widehat{G}(x)$ and the limit state.

3.3.3 Learning function H

Proposed in [Lv et al., 2015]. This function is based on the information entropy theory and it measures the uncertainty of $\widehat{G}(x)$. The best next point to add to the DoE is $x^* \in S$ such that $H(x^*) = \max(H(\boldsymbol{x}))$. The stopping condition that is $H(x^*) \leq 0.5$.

It is defined as:

$$\begin{aligned} H(x) = \Bigg| \ln\left(\sqrt{2\pi}\sigma_{\widehat{G}}(x) + \frac{1}{2}\right) \left[\Phi\left(\frac{D^-}{\sigma_{\widehat{G}}(x)}\right) - \Phi\left(\frac{-D^+}{\sigma_{\widehat{G}}(x)}\right)\right] \\ - \left[\frac{D^-}{2}\phi\left(\frac{D^-}{\sigma_{\widehat{G}}(x)}\right) + \frac{D^+}{2}\phi\left(\frac{-D^+}{\sigma_{\widehat{G}}(x)}\right)\right]\Bigg| \end{aligned} \tag{3.15}$$

where

$$\begin{aligned} D^+ &= 2\sigma_{\widehat{G}}(x) + \widehat{G}(x) \\ D^- &= 2\sigma_{\widehat{G}}(x) - \widehat{G}(x) \end{aligned}$$

3.4 Application examples

Some application examples are presented, which follow the setup:

- Definition of the design space, indicating the number of variables involved, together with the distribution that each one follows.
- Statement of the performance function $G(\boldsymbol{x})$ to be evaluated.
- Presentation of the probability of failure $\widehat{p_f}$ obtained with each learning function and the number of calls N_{call} to the performance function, comparing them with MCS.

The exercises are solved using the implementation of the Gaussian Process Regressor from the Python library *scikit-learn* ([Pedregosa et al., 2011]), using an squared-exponential kernel. The default optimizer is modified, increasing the number of maximum iterations, and also using a noise in the observations of $\sigma_n^2 = 1 \times 10^{-7}$, in order to avoid numerical issues.

3.4.1 Example 1: Illustration of how the learning functions work

The first example is a simple problem with just one variable, that follows a normal distribution with mean 0 and standard deviation 2, and the performance function considered is:

$$G(x) = \sin x \tag{3.16}$$

The problem is solved only with the learning function U, and the metamodel is initialized with just 5 points in the initial DoE.

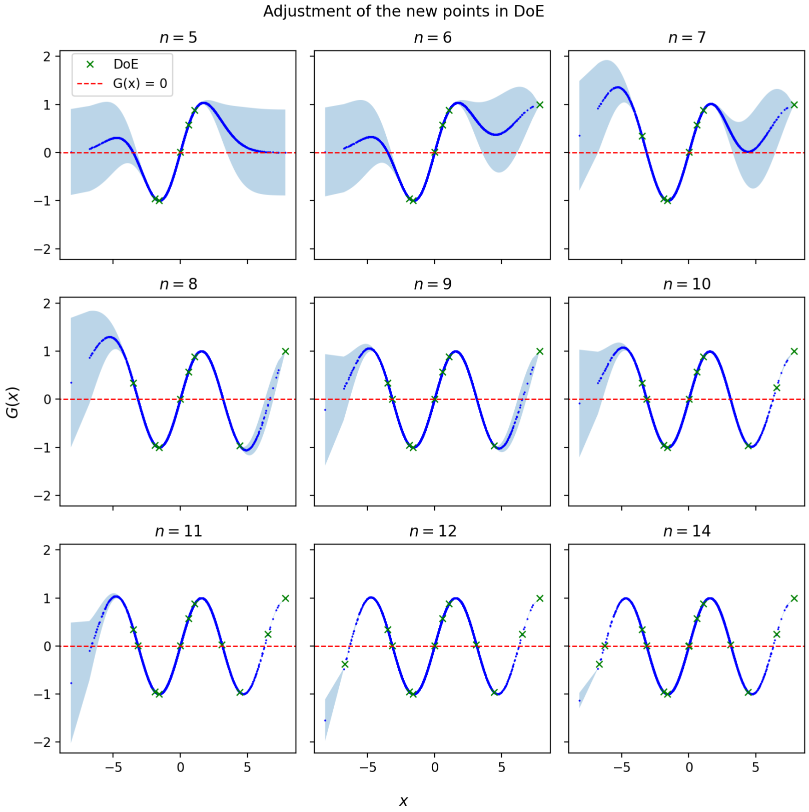


Figure 3.1: Prediction and standard deviation obtained with n points in the DoE.

Method	N_{call}	$\widehat{p_f}$ (C.O.V$_{\widehat{p_f}}$)	$\epsilon_{\widehat{p_f}}(\%)$
Monte Carlo	1×10^4	4.944×10^{-1}(1.01%)	-
AK-MCS+U	14	4.944×10^{-1}	0

Table 3.1: Results of example 1

In figure 3.1 it can be seen how the algorithm selects the next point to be added to the DoE such that it reduces the most the variance near to the limit $G(\boldsymbol{x}) = 0$. Even though there is still some considerable variance in some regions, it stops at $N = 14$ because this uncertainty is away from the limit, so it is unlikely to affect the estimation of the probability of failure. In fact, the results in table 3.1 reveal that a very accurate estimation is obtained.

3.4.2 *Example 2: series system with four branches*

Taken from [Echard et al., 2011]. Both random variables are standard normal distributed. The performance function is:

$$G(x_1, x_2) = \min \left\{ \begin{array}{c} 3 + 0.1(x_1 - x_2)^2 - \frac{(x_1+x_2)}{\sqrt{2}}; \\ 3 + 0.1(x_1 - x_2)^2 + \frac{(x_1+x_2)}{\sqrt{2}}; \\ (x_1 - x_2) + \frac{4}{\sqrt{2}}; \\ (x_2 - x_1) + \frac{4}{\sqrt{2}} \end{array} \right\} \tag{3.17}$$

with k taking two different values: 6 and 7.
From the results on tables 3.2 and 3.3 we can see that the learning function U obtained the most accurate estimations of $\widehat{p_f}$, being exactly the same of the MCS. In the case $k = 6$, the other two functions performed well, they only failed at clasifying one point. In the other one, with $k = 7$, EFF made some misclassifications, but the learning function H, although calling the performance function less times, had an error of 30%.

Method	N_{call}	$\widehat{p_f}$ (C.O.V$_{\widehat{p_f}}$)	$\epsilon_{\widehat{p_f}}$(%)
Monte Carlo	1×10^6	4.433×10^{-3}(1.5%)	-
AK-MCS+U	126	4.433×10^{-3}	0
AK-MCS+EFF	123	4.432×10^{-3}	0.02
AK-MCS+H	113	4.434×10^{-3}	0.02

Table 3.2: Results of example 2 with $k = 6$

Method	N_{call}	$\widehat{p_f}$ (C.O.V$_{\widehat{p_f}}$)	$\epsilon_{\widehat{p_f}}$(%)
Monte Carlo	1×10^6	2.161×10^{-3}(2.15%)	-
AK-MCS+U	103	2.161×10^{-3}	0
AK-MCS+EFF	107	2.156×10^{-3}	0.23
AK-MCS+H	65	1.5×10^{-3}	30.59

Table 3.3: Results of example 2 with $k = 7$

In the figure 3.2 the actual distribution of the two classes in the MC population is displayed, while in the figure 3.3 there is the distribution predicted by AK-MCS+U at several stages. Additionally, figure 3.3 give some insights about the update of the DoE, showing how the selected points tend to come from near the limit $G(\boldsymbol{x}) = 0$.

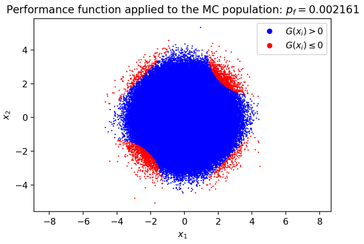


Figure 3.2: Example 2 with $k = 7$. Evaluation of the performance function on the MC population.

Figures 3.4 and 3.5 display how the probability of failure converges with each learning function, and how the selected points make the learning criteria tend to the stopping conditions. In both cases there are similar behaviors. The EFF is proned to converge more consistently to the stopping condition. The function U usually gets a good estimation well before finishing. The function H presents the most irregular results.

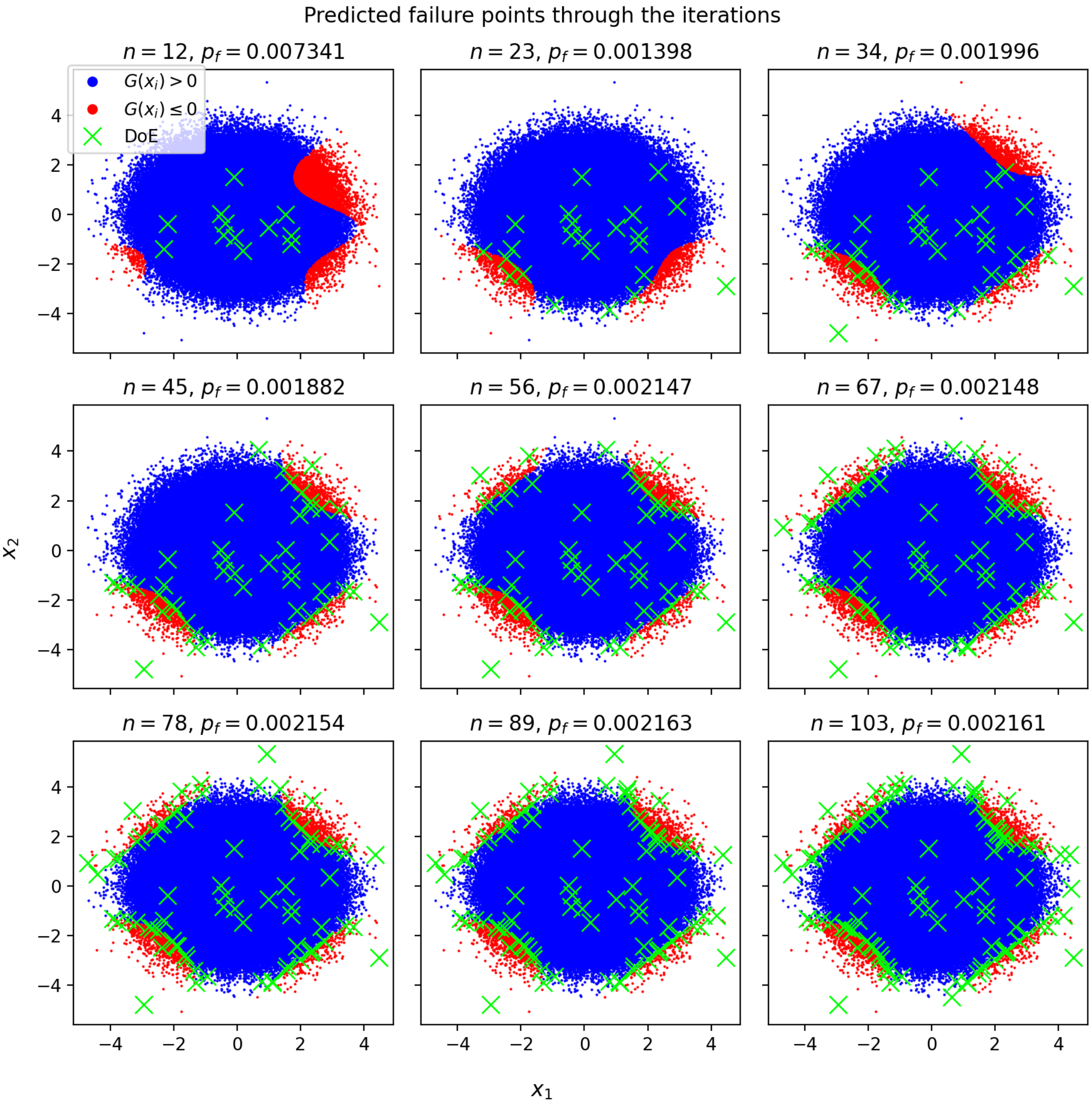
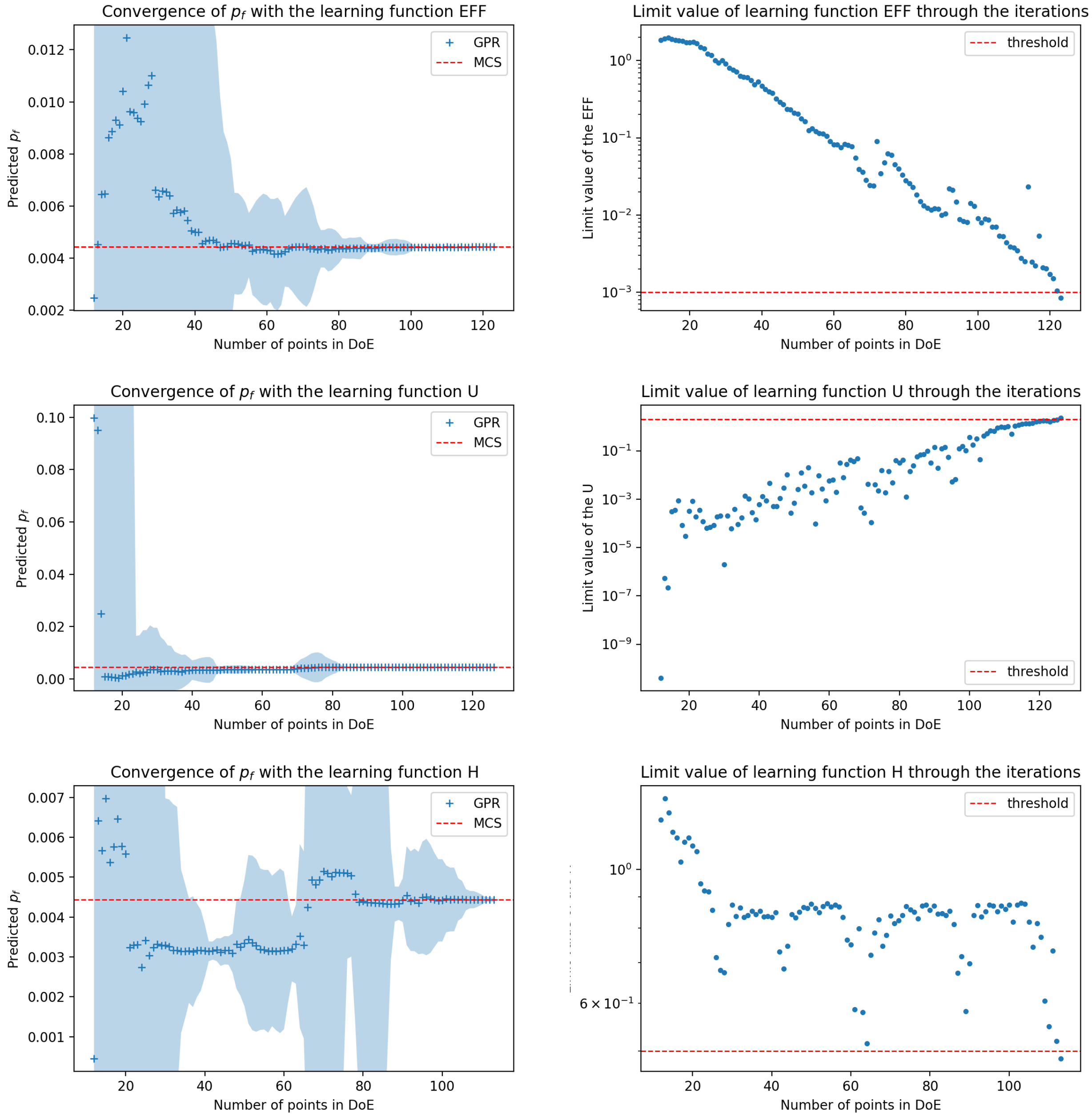


Figure 3.3: Example 2 with $k = 7$. Prediction made by AK-MCS+U at several stages.



Convergence of p_f with the learning function EFF

Limit value of learning function EFF through the iterations

Convergence of p_f with the learning function U

Limit value of learning function U through the iterations

Convergence of p_f with the learning function H

Limit value of learning function H through the iterations

Figure 3.4: Results of example 2 with $k = 6$

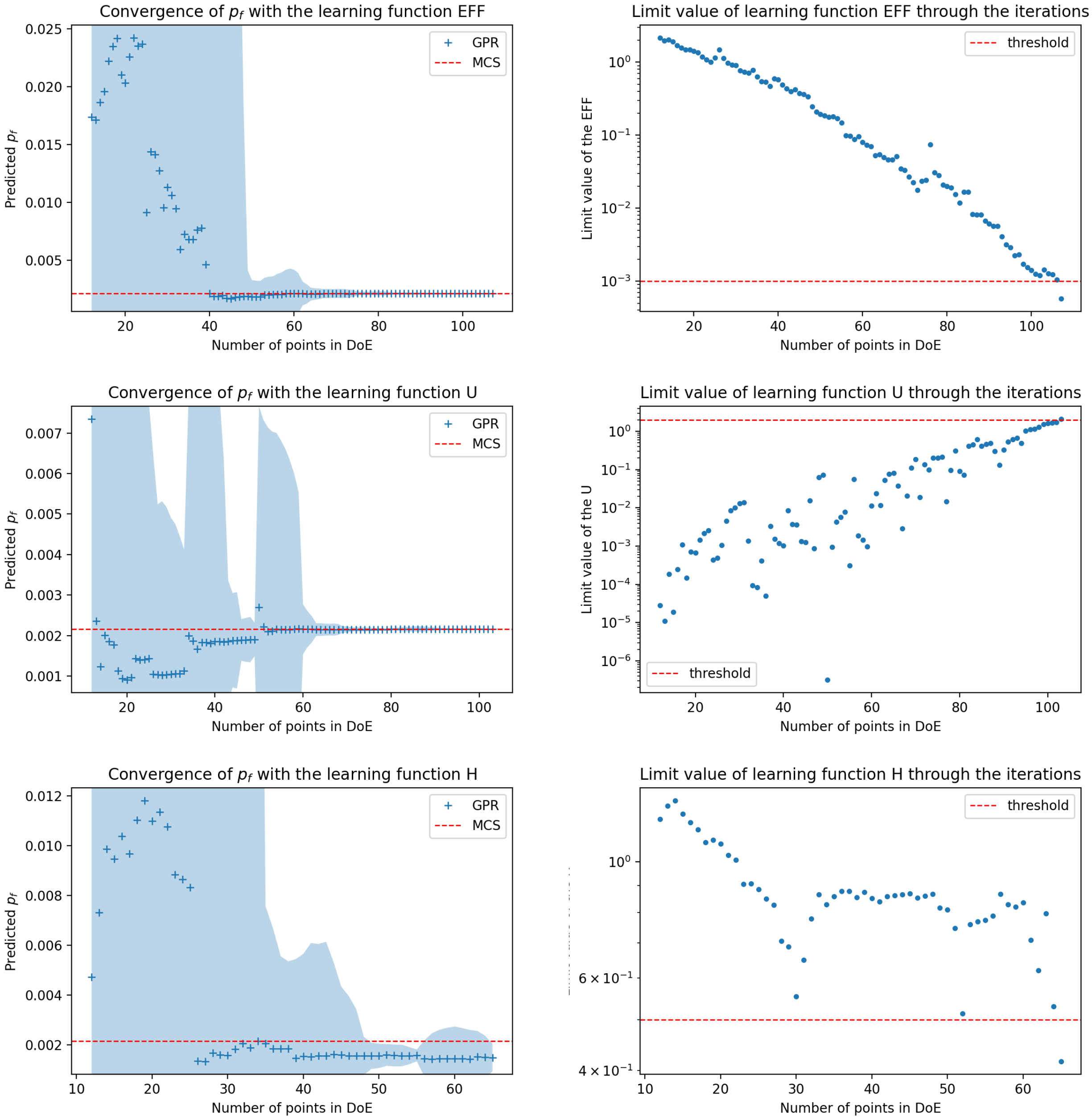


Figure 3.5: Results of example 2 with $k = 7$

3.4.3 *Example 3: dynamic response of a non-linear oscillator*

Taken from [Schueremans and Gemert, 2005]. It is a non-linear undamped single degree of freedom system, as the one shown in figure 3.6. The involved variables are listed in table 3.4.

$$G(\boldsymbol{x}) = 3r - \left| \frac{2F_i}{m\omega_0^2} \sin\left(\frac{\omega_0 t_1}{2}\right) \right| \tag{3.18}$$

where $\omega_0 = \sqrt{\frac{c_1+c_2}{m}}$

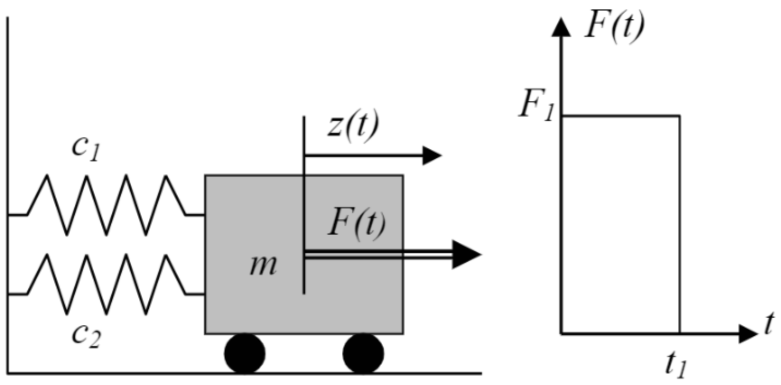


Figure 3.6: Example 3. Non-linear oscilator. Taken from [?]

Variable	P.D.F	Mean	Standard deviation
m	Normal	1	0.05
c_1	Normal	1	0.1
c_2	Normal	0.1	0.01
r	Normal	0.5	0.05
F_1	Normal	1	0.2
t_1	Normal	1	0.2

Table 3.4: Random variables of example 3

Table 3.5 contains the obtained results. In contrast with the other examples, in this case the function H performs quite well, obtaining the exact result, just as the function U, but with some more calls to the performance function.

In figure 3.7 it is evident that the function EFF met its stopping condition before converging to the solution, unlike the other two functions.

Method	N_{call}	$\widehat{p_f}$ (C.O.V$_{\widehat{p_f}}$)	$\epsilon_{\widehat{p_f}}$(%)
Monte Carlo	7×10^4	2.7814×10^{-2}(2.23%)	-
AK-MCS+U	60	2.7814×10^{-2}	0
AK-MCS+EFF	43	2.7671×10^{-2}	0.51
AK-MCS+H	66	2.7814×10^{-2}	0

Table 3.5: Results of example 3

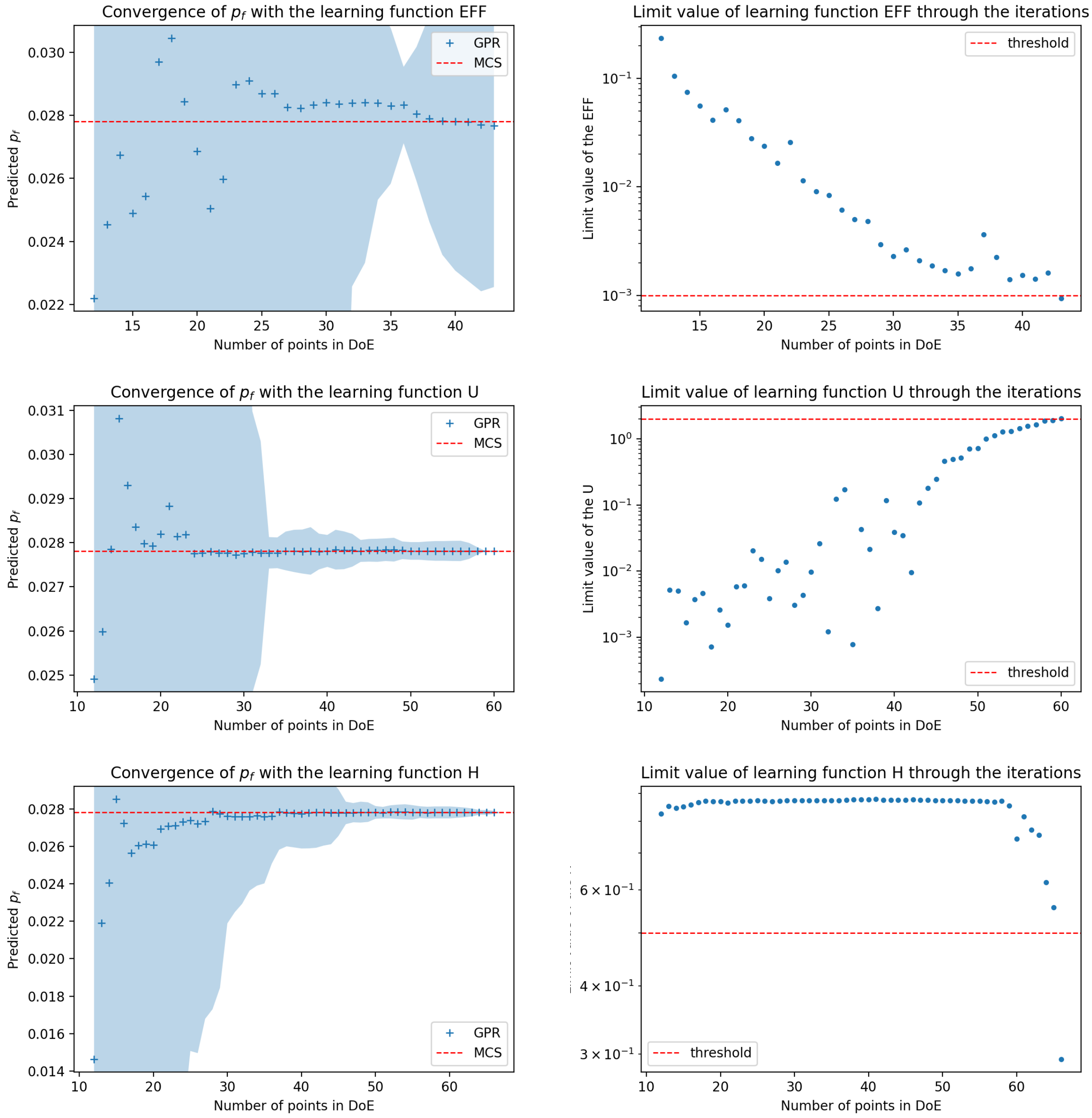


Figure 3.7: Results of example 3

Chapter 4

Conclusions

The AK-MCS algorithm proved to have several advantages, being a substantial improvement of the basic MCS method. It retains the advantages of the latter, improving its main disadvantage which is having to call the expensive performance function so many times. This improvement becomes more evident as the complexity of the performance function increases.

The choice of one learning function over another should be influenced by the nature and form of the problem. Although they are all based on the same parameters, the manner in which they determine the evolution of the prediction process varies by giving greater importance to different aspects of the design space and the performance function. Likewise, the choice of thresholds for the stopping conditions is of great importance, as they represent the usual trade-off between cost and benefit. It could be seen that in the worked examples, the suggested threshold of the learning function H was too high, yet for some particular problems it was low enough.

Given the simplicity of the nature of the method, its potential for improvement is very high. In principle, the formulation of new learning functions is a boundless task. Moreover, different ways of approaching the problem following the same principles have been studied. For example, in [Peijuan et al., 2017] geometrical considerations are presented that allow to improve the efficiency of the method, although limiting its applicability. In [Balesdent et al., 2013] work is done on a variant of the crude MCS. And as well as these, there are many modifications that are pending to be studied, and even proposed, that could further improve the qualities of the method.

Regarding the proposed objectives, the MCS method, its formulation and main characteristics were studied, as well as the kriging algorithm, after having made a contextualization in the field of reliability analysis to determine failure probabilities. The article [Echard et al., 2011] was studied in its entirety, and several examples were implemented, comparing the results given by AK-MCS with those obtained with MCS.

For future work, one can study new learning functions, and try to categorize them according to the type of problems for which each is most appropriate. The application of active learning methods combined with variations to the basic MCS algorithm could also be studied. The use of different kernels to define the covariance function is also an important field of study that is continually developing, so a review of these can be considered.

Bibliography

M. Balesdent, J. Morio, and J. Marzat. Krigin-based adaptive importance sampling algorithms for rare event estimation. *Structural Safety*, 44:1–10, 2013. DOI: 10.1016/j.strusafe.2013.04.001.

B. Bichon, M. Eldred, L. Swiler, S. Mahadevan, and J. McFarland. Efficient global reliability analysis for nonlinear implicit performance functions. *AIAA Journal*, 46(10), 2008. DOI: 10.2514/1.34321.

C. Bishop. *Pattern Recognition and Machine Learning*. Springer, first edition, 2006. ISBN 979-0387-31073-8.

G. Box and M. Muller. Note on the generation of random normal deviates. *The Annals of Mathematical Statistics*, 29(2):610–611, 1958. DOI: 10.1214/aoms/1177706645.

S. Choi, R. Grandhi, and R. Canfield. *Reliability-based Structural Design*. Springer, first edition, 2006. ISBN 9781846284441.

O. Ditlevsen and H. Madsen. *Structrual Reliability Methods*. Wiley, first edition, 1996. ISBN 0-471-96086-1.

B. Echard, N. Gayton, and M. Lemaire. AK-MCS: An active learning reliability method combining Kriging and Monte Carlo Simulation. *Structural Safety*, 33:145–154, 2011. DOI: 10.1016/j.strusafe.2011.01.002.

J. Hurtado. *Structural Reliability: Statistical Learning Perspectives*. Springer Berlin Heidelberg, first edition, 2013. ISBN 9783540409878.

M. Lemaire. *Structural Reliabiity*. Wiley, first edition, 2009. ISBN 978-1-84821-082-0.

Z. Lv, Z. Lu, and P. Wang. A new learning function for Kriging and its applications to solve reliability problems in engineering. *Computers and Mathematics with Applications*, 70:1182–1197, 2015. DOI: 10.1016/j.camwa.2015.07.004.

M. Matsumoto and T. Nishimura. Mersenne Twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1): 3–30, 1998. DOI: 10.1145/272991.272995.

R. Melchers and A. Beck. *Structural Reliability Analysis and Prediction*. John Wiley & Sons Ltd, third edition, 2018. ISBN 9781119266075.

A. Nowak and K. Collins. *Reliability of Structures*. McGraw-Hill, first edition, 2000. ISBN 0-07-048163-6.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Z. Peijuan, W. Ming, Z. Zhouhong, and W. Liqi. A new active learning method based on the learning function U of the AK-MCS reliability analysis method. *Engineering Structures*, 148:185–194, 2017. DOI: 10.1016/j.engstruct.2017.06.038.

C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, first edition, 2006. ISBN 0-262-18253-X.

L. Schueremans and D. Gemert. Benefit of splines and neural networks in simulation based structural reliability analysis. *Structural Safety*, 27: 246–261, 2005. DOI: 10.1016/j.strusafe.2004.11.001.