A Reliability-Based Design Optimization Method for Geotechnical Engineering Problems Based On Active-Learning Kriging Metamodeling

Geo-Engineering MSc Thesis

Muhammad Rayyan

TU Delft

A RELIABILITY-BASED DESIGN OPTIMIZATION METHOD FOR GEOTECHNICAL ENGINEERING PROBLEMS BASED ON ACTIVE-LEARNING KRIGING METAMODELING

by

Muhammad Rayyan
5032334

in partial fulfillment
of the requirements for the degree of

Master of Science
in Geo-Engineering
at the
Delft University of Technology

To be defended publicly on Monday, 29th August 2022 at 09:00 CEST

Supervisors: Dr. ir. A.P. (Bram) van den Eijnden
Prof. Dr. M.A. (Michael) Hicks
Dr. ir. R.C. (Robert) Lanzafame

Muhammad Rayyan: *A Reliability-Based Design Optimization Method for Geotechnical Engineering Problems Based On Active-Learning Kriging Metamodeling* (2022)

to obtain the degree of Master of Science at the Delft University of Technology

The work in this thesis was made in the:

Geo-Engineering Track
Faculty of Civil Engineering & Geoscience
Delft University of Technology (TU Delft)

An electronic version of this thesis is available at http://repository.tudelft.nl/.
Front and back covers: Typical dyke structures in the Netherlands (www.wallpaperflare.com).

ABSTRACT

Using an expensive-to-evaluate numerical model, such as a finite element method (FEM) model, is deemed unavoidable in solving modern geotechnical engineering problems. At the same time, the application of reliability analysis in dealing with uncertainties (e.g. soil properties) is increasing rapidly. This could pose a time-wise problem for an FEM model since reliability analysis normally takes much more than only one realization (function call) of the model. It becomes a bigger problem when a design optimization process is taking place. More often than not, design optimization is performed by a "trial-and-error" method in practice, which the process itself would even take longer just to give engineers the "sense" of achieving an optimal design (in terms of safety and economy). Therefore, the actual optimality of the design is not systematically proven and quantified. This research proposes a novel reliability-based design optimization (RBDO) method by combining existing theories regarding active-learning Kriging-based Monte Carlo Simulation (AK-MCS) and (1+1)-Covariance Matrix Adaptation evolution scheme ((1+1)-CMA-ES). To achieve accuracy and efficiency, the method consists of four enrichment stages. These enrichment stages ensure the method accurately and efficiently predicts the optimal design combination by considering the reliability constraint. The chosen case study is the reinforcement design of the Starnmeer polder dyke in the Netherlands, which is simulated as an FEM model. Within a limited number of function calls, the proposed RBDO method could accurately predict the optimal dimensions of the dyke that delivers the targeted reliability index. The reliable performance of the proposed method is further demonstrated by solving three analytical optimization problems.

Keywords: RBDO, Kriging, AK-MCS, MCS, (1+1)-CMA-ES, geotechnical reliability, FEM, dyke reinforcement

ACKNOWLEDGEMENT

Alhamdulillah, I would like to thank Allah SWT for giving me the strength to complete my master thesis. Apart from numerical modeling, I have been intrigued by the application of reliability analysis in geotechnical engineering problems since the first day I heard about it. I find it to be far superior and makes much more sense compared to the traditional deterministic approach that civil engineers use in general. Unfortunately, its application takes forever to conclude when one deals with a numerical model (FEM). Therefore, when I was introduced to the concept of Kriging metamodel and reliability-based design optimization (RBDO) to deal with a FEM model, I started to feel the urge and passion to find a way to make it applicable in practice with a common personal computer.

Firstly, I would like to express my gratitude to Dr. ir. Bram van den Eijnden as my main supervisor, for inspiring and introducing me to the topics of Kriging metamodel and RBDO (among others). Bram has been guiding me to achieve the skills needed to complete the master thesis even far before the start of the thesis itself. Since I am also highly interested in solving engineering problems through computer programming, I am so glad and grateful that Bram was there. Personally for me, it feels like I was learning from Bram throughout my whole master's study, which I enjoyed every second of it. I hope someday I could repay my debt and continue working under his guidance.

Furthermore, I would like to thank my supervisors, Prof. Dr. Michael Hicks and Dr. ir. Robert Lanzafame. Michael has introduced me to the concept of reliability analysis in geotechnical engineering and provided valuable support during the progress meetings. Robert has been giving me valuable technical and non-technical inputs throughout the entire master thesis that allowed me to have a different perspective on understanding the topic.

Last but not least, I would like to extend my gratitude to my wife Rosi, my parents, my brother and sister, and my family for all of the love and support, my employer PT PP for all of the support and opportunity, and The Foundation Justus & Louise van Effen for making it possible for me to do the master study. I would also like to thank all of my colleagues, friends, and everyone who has helped me to complete my master's study and made it a valuable experience.

I hope this master thesis would give contributions to those who are seeking to learn or further understand the application of reliability analysis and RBDO in the field of geotechnical engineering.

Muhammad Rayyan
Delft, July 2022

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ACRONYMS

LIST OF SYMBOLS

$\mathbf{A}$	Covariance matrix Cholesky decomposition
β	Reliability index
β^-	Lower bound of β
β^+	Upper bound of β
β	Supporting parameters for (1+1)-CMA-ES
$\mathbf{C}$	Covariance matrix
$\mathfrak{C}_q$	Monte carlo population for a set of design variables
$c(\mathbf{d})$	Cost function based on design variables d
c	Supporting parameters for (1+1)-CMA-ES
c_c	Supporting parameters for (1+1)-CMA-ES
c_P	Supporting parameters for (1+1)-CMA-ES
c'	Soil effective cohession (kPa)
c_{cov}^-	Supporting parameters for (1+1)-CMA-ES
c_{cov}^+	Supporting parameters for (1+1)-CMA-ES
$\mathbb{D}$	Design variables domain
d	Supporting parameters for (1+1)-CMA-ES
$\mathbf{d}$	Design parameter
δ_{P_f}	The discrepancy of $\hat{P}_f$ against $\bar{P}_{f,k}$
η	Convergence criterion
η_q	Convergence of the first local enrichment
$\bar{\eta}_q$	First local enrichment convergence threshold
η_{P_f}	Convergence of the second local enrichment
$\bar{\eta}_{P_f}$	Second local enrichment convergence threshold
E	Modulus of elasticity
f_j	Soft constraint function
f_{stop}	Optimization convergence criterion
$f_{\mathbf{z}}$	The distribution of environment parameters $\mathbf{x}$
γ	Unit weight
γ'	Effective unit weight
g_j	Constraint function for optimization process
g_k	k-th Limit state function
$\hat{g}_p$	Metamodel predictor
$\mathbf{G}_k$	MCS output prediction vector of the real performance function
$\mathcal{G}$	Monte carlo response
$G(x)$	Real performance function
$\hat{G}(x)$	Metamodel approximate of a performance function
H	A design variable in the case study
$\mathbf{K}$	Kernel matrix
$\mu_{\hat{G}}$	Kriging mean, the best prediction of Kriging process
μ_x	A randomly selected mean to generate x candidates in evaluating global enrichment convergence
ν	Poisson's ratio
N	Number of function call
N_{mc}	The size of Monte Carlo population
n	Total number of design variable
n_f	Total number of soft constraint
n_g	Total number of limit state functions (e.g. multiple modes of failure)
n_s	Total number of soft constraint
n_h	Total number of hard constraint
P_f	Probability of failure
$\hat{P}_f$	Metamodel prediction for probability of failure

P_{succ}	The success probability estimate for (1+1)-CMA-ES iteration
P_{target}	Supporting parameters for (1+1)-CMA-ES
$\bar{P}_{f,k}$	Probability of failure target for k-th limit state function
P_f^-	Lower bound of P_f
P_f^+	Upper bound of P_f
ρ_M	Matern correlation function
ϕ'	Soil effective internal friction angle
Q_α	The quantile of a certain set of design variables
$q_{\alpha k}$	Quantile of a certain set of design variables
q_α^-	Quantile's lower bound
q_α^+	Quantile's upper bound
σ	(1+1)-CMA-ES global step size
σ_0	Initial (1+1)-CMA-ES global step size
$\sigma_{\hat{G}}(x)$	Kriging variance, indicating how certain a Kriging prediction is
θ	Kernel hyperparameter
$U(x)$	Learning function U value to determine the next training data
U_{comp}	The best U of several probability constraints'
$\mathbf{v}_j$	Constraint vector
$\mathbf{w}_j$	Adaptation vector to update matrix $\mathbf{A}$
W	A design variable in the case study
$\mathbf{x}$	(1+1)-CMA-ES parent parameters
$\mathbf{X}(\mathbf{d})$	Input vector based on independent design variables
z	a random standard-normally parameters to generate an offspring in (1+1)-CMA-ES
$\mathbf{y}$	(1+1)-CMA-ES offspring
$\mathbf{Z}$	Input vector of environmental variables (variables other than the design variables)

1 INTRODUCTION

1.1 BACKGROUND AND MOTIVATION

The Finite Element Method (FEM) application is an integrated part of modern geotechnical engineering problem-solving. The increased use of FEM to accurately solve various geotechnical engineering problems is deemed "unavoidable" these days. However, the application of a FEM-based analysis in geotechnical engineering is, more often than not, expensive to evaluate (in terms of computational effort and time). On the other hand, despite its relatively recent development, the application of reliability-based design and analysis of geotechnical engineering structures is increasing steadily. Reliability analysis is very useful in quantifying uncertainties, therefore it is perfectly suitable to be applied in a field of study that deals with uncertainties on a regular basis, such as geotechnical engineering. Unfortunately, despite the rapid advancement of technology, FEM-based analyses of geotechnical engineering problems still normally take more than 20 seconds (if not minutes) to conclude. This means that a reliability analysis by using a Monte Carlo Simulation (MCS) with 10^6 realizations would take more than 231 days to conclude! On the other hand, a geotechnical engineering structure (or any engineering structure in this matter) needs to be designed as safely and economically as possible. Therefore, it is a common practice for a civil engineer to perform the classic trial-and-error approach to get the most (or "deemed" as the most) optimal design (in terms of safety and economy). This trial-and-error process itself could be time-consuming since it is not systematically performed. Moreover, a trial-and-error process could not quantify the accuracy (or confidence) level achieved by the chosen design since it mostly relies on the "hunch" or "engineering judgment" of the engineer himself.

Therefore, for a specific geotechnical engineering problem, there is a need in performing a systematic optimization method to achieve the most economical design possible while also fulfilling a certain reliability target within an acceptable computation time. To overcome the problem, an effective Reliability-Based Design Optimization (RBDO) method that can be applied to an FEM-based geotechnical engineering problem is needed. The application of an advanced metamodeling-based RBDO method to an FEM-based geotechnical engineering problem seems very promising in future practice. However, despite the development of metamodeling-based RBDO is becoming more and more well-established in other engineering practices, its use in geotechnical engineering has not been really proven yet as the case references are still relatively limited.

The FEM model to be analyzed in this research is a dyke reinforcement by the means of berm extension, which is inspired by a case study of the Netherlands' Starnmeer Polder Dyke reinforcement as analyzed in Hicks et al. [2019] (Figure 1.1). The FEM model will be simulated with PLAXIS 2D (Brinkgreve et al. [2020]). The goal is to make a design that makes the berm extension dimensions (W & H) to be as small as possible while also fulfilling its reliability target at the same time.

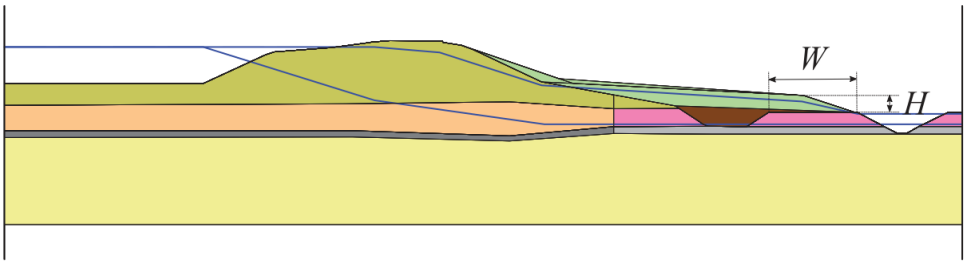

Figure 1.1: The model case study: Starnmeer Polder dyke reinforcement, the Netherlands (Hicks et al. [2019])

1.2 RESEARCH QUESTIONS AND OBJECTIVE

Based on the problems described above, the following main question and sub-questions are formulated for this research:

"How to efficiently and accurately perform a design optimization of a dyke reinforcement problem with an expensive FEM-based performance function while also fulfilling the reliability target?"

- *How to combine a metamodel and an optimization process into an RBDO method?*
- *How to achieve the accuracy of the RBDO method?*
- *How to achieve the efficiency of the RBDO method?*
- *What is the optimal design for the dyke reinforcement based on the proposed RBDO method?*

Based on the questions formulated above, the objective of this research is: **to optimize an expensive FEM-based dyke reinforcement design by using an RBDO method consisted of an active-learning metamodel combined with an optimization scheme to fulfill the reliability target while making the computation effort inexpensive and time-efficient.**

1.3 RESEARCH OVERVIEW

To solve the research questions mentioned above, a novel RBDO method will be introduced in this research. The method is developed by combining the existing well-known metamodeling and optimization theories with strategies that further improve the accuracy and efficiency of the method. This development is realized through algorithms implemented in Python programming language, while the FEM case study is modeled in PLAXIS 2D.

The proposed RBDO method works by constructing a metamodel based on a certain number of enrichments (or "training data" from the original model), which is then used to perform performance function evaluations in place of the expensive original FEM model evaluations (or "function calls"). Moreover, the optimization method will be used to locate the optimal design variables. Since the main aim of the RBDO method is to find an optimal set of design variables within a limited number of function calls for the FEM-based case study, the method is developed by putting accuracy and efficiency as the main consideration. The accuracy will be obtained by selecting the right amount of enrichment for the metamodel, while the efficiency will be obtained by performing the enrichment in the right points (or in the right "place").

The research is structured as the following. Chapter 2 discusses the literature study of existing related RBDO approaches. The chapter also points out the differences between this research and existing RBDO researches. Chapter 3 explains the proposed RBDO method by introducing the metamodel and the optimization process, and further explains how to couple them. The chapter also explains the enrichment stages and other strategies which are developed to achieve accuracy and efficiency. Moreover, Chapter 4 demonstrates the accuracy and efficiency of the proposed RBDO method compared to other existing RBDO methods by solving existing analytical problems. This chapter also shows the generality of the method, therefore, it can be applied to geotechnical engineering problems other than the case study. Furthermore, the RBDO method is applied to the case study in Chapter 5. The chapter discusses how the RBDO obtain the optimal design of the case study's noisy FEM model within a limited number of function calls. To sum up, Chapter 6 summarizes the RBDO method and its performance, including the advantages and disadvantages. Finally, the algorithms of the proposed RBDO method (based on Chapter 3) are provided in Appendices A and B, therefore, one can have a better understanding of the method.

2 LITERATURE STUDY

The existing recent research regarding metamodeling and RBDO will be discussed in the following subsections. Furthermore, the literature study will also provide the reasons why the methods performed in this research are chosen.

2.1 METAMODELING

Metamodel (also known as "surrogate model"), in short, is a simpler function (model) that is constructed to "imitate" the complex and expensive real performance function to a certain accuracy. Metamodel is constructed based on a set of "training data", which is normally called as Design of Experiments (DOE). Therefore, it is expected of the metamodel to be more accurate when it has more data in the DOE. Selecting the right DOE is crucial in constructing an accurate and efficient metamodel. With the right metamodel, an expensive-to-evaluate performance function's (like an FEM model) response could be predicted to a certain accuracy within orders of magnitude less computational time. This makes the application of reliability analysis of an expensive model to be much less time-consuming. Moreover, the metamodel-based reliability analysis framework could be further improved to accommodate an accurate and efficient RBDO application. An illustration of a metamodel prediction can be seen in Figure 2.1.

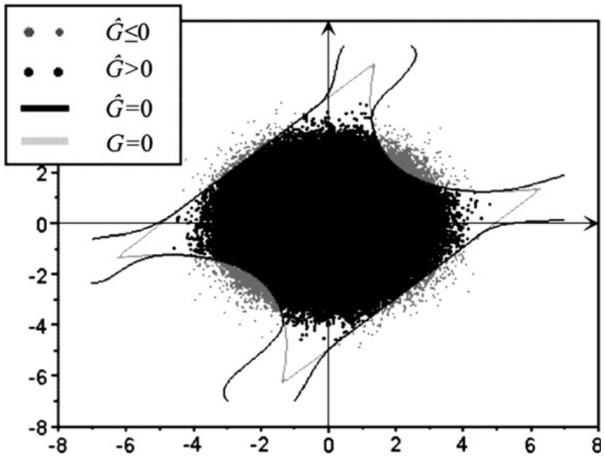

Figure 2.1: An illustration of a metamodel $\hat{G}$ that tries to re-create a real performance function G (Echard et al. [2011]).

There have been numerous types of metamodel. One of the commonly applied one is Response Surface based on Polynomial Chaos expansion (Sudret and Kiureghian [2002] and Blatman and Sudret [2009]) which are used for their speed and high global interpolation accuracy. However these methods have complex and impractical definitions of the DOE (Echard et al. [2011]). Another well-known metamodel is the Support Vector Machine (SVM) (Claudio and Alí [2002]). SVM is a kernel-based metamodeling technique initially formulated for classification problems, and later extended to regression problems (Teixeira et al. [2021]). However, the selection of its parameters is relatively complex and less straightforward.

A well-established stochastic metamodel approach, Kriging (developed in the fifties and sixties by Danie G. Krige for geostatistics application), has been extensively applied in reliability analysis of geotechnical engineering practices due to its interesting features (Kentrop [2021] and van der Werf [2021] among many others). It is an exact interpolation method (also known as the Gaussian process

predictor), which means it could predict an exact performance function value of a point that belongs to the DOE. Moreover, thanks to its stochastic feature, Kriging could predict the performance function value of any points and the local variance (Kriging variance) of the prediction (Echard et al. [2011]). Therefore the local uncertainty of any prediction can be quantified from this variance (the higher the variance is, the less certain the prediction is). Furthermore, Kriging model is particularly well-suited for strong non-linearity Teixeira et al. [2021], thus makes it especially superior over the Response Surface, Polynomial Chaos Expansion, and Support Vector Machine when dealing with a strongly non-linear and noisy model response (van den Eijnden et al. [2021] and Teixeira et al. [2021]). Moreover, the strategy in van den Eijnden et al. [2021] to overcome noisy response of FEM model will also be considered in this research. More discussions regarding the problem of noisy data in the Gaussian Process (Kriging) metamodel can be found in Forrester et al. [2008]. Due to its advantages and proven effectiveness in geotechnical engineering applications, Kriging metamodel will be implemented in this research.

The reliability method to be combined with the Kriging metamodel is MCS. The MCS reliability analysis method is chosen since it is a robust sampling-based (or simulation-based) method. Despite its need for a large sample population and number of function calls, evaluating those function calls in a metamodel is not considered as a problem since its computational effort is orders of magnitude less than the actual model's. Moreover, sampling-based reliability methods are generally more accurate than approximation (or gradient-based) methods (Rayyan [2021]). The MCS is also easy to implement and applicable to a large domain of application (Echard et al. [2011]), including geotechnical engineering.

2.2 ACTIVE-LEARNING METHOD

Generally speaking, an active learning method extends the DOE by including more strategic training data to improve the accuracy of the metamodel. An example of an active learning metamodel application is illustrated in Figure 2.2. An active learning function is the function responsible in choosing these strategic data. Some well-known active learning methods have been developed in the last few decades to perform sufficient performance function evaluation. Their use is helpful in determining the best training data to be added to the DOE. Thanks to Kriging prediction and Kriging variance, active learning methods could be perfectly combined with Kriging metamodeling.

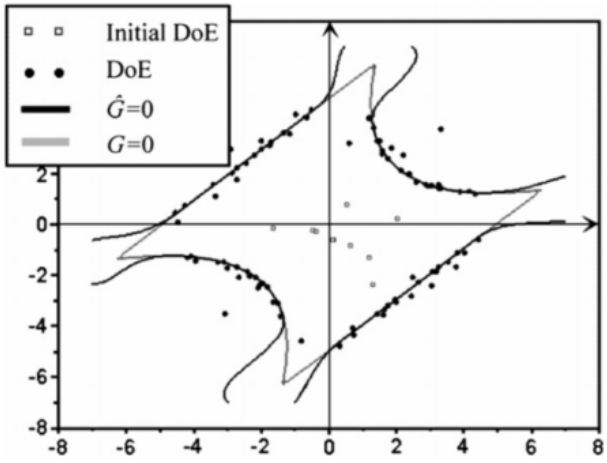

Figure 2.2: An illustration of active learning application in selecting the most strategic DOE to improve the metamodel (Echard et al. [2011]). In this case, the most strategic DOE should be located close to the limit state function line.

A learning function called Expected Feasibility Function (EFF) was proposed by Bichon et al. [2008] to estimate the next best training data by creating a function that provides an indication of how well the true value of the response is expected to satisfy a certain constraint (performance function's limit state, $G(X) = 0$), which is then used to define an active learning reliability analysis method called Efficient Global Reliability Analysis (EGRA). The function evaluates the data in the region around the limit state. The data with the maximum EFF value is then chosen as the new addition to the DOE. This method is inspired by Efficient Global Optimization (EGO) from Jones et al. [1998] and the Kriging

contour estimation method from Ranjan et al. [2008]. Despite producing results that are far more accurate and efficient than Most Probable Point (MPP)-based approach such as First Order Reliability Method (FORM), EGRA was effectively tested for problems that only have two independent variables (its effectiveness against problems with larger dimensions was still being investigated at the time). It is worth noting that EFF was designed for EGRA (Bichon et al. [2008]), which approximates the whole performance function. Moreover, EFF has a different concept that does not guarantee it to perform well with Active Learning Kriging-based Monte Carlo Simulation (AK-MCS) since EFF evaluates the whole limit state while AK-MCS only evaluates the data (points) in the Monte Carlo population that is in close proximity with the limit state.

Contrary to EFF which evaluates the whole performance function, Echard et al. [2011] proposed a learning function called U that only evaluates a generated Monte Carlo population n_{MC}. The learning function U is defined in such a way that its value can predict whether the evaluated data (point) is close to the limit state $G(x) = 0$ (having a high potential of having the performance function's sign change from negative to positive and vice versa), having an important uncertainty (high Kriging variance), or both at the same time. These potentially "interesting" data (points) are defined in such a way to have low U values, therefore, the next best training data to be added in DOE expansion is the data that has the lowest U value. The learning function U is formulated as Equation 2.1.

$$U(x) = \frac{|\hat{G}(x)|}{\sigma_{\hat{G}}(x)} \tag{2.1}$$

In a more recent development, van den Eijnden et al. [2021] took the concept of learning function U further to accommodate the noisy and incomplete performance function with an adaptive Importance Sampling reliability analysis method, where the samples have unequal weights (contrary to MCS samples that have equal weights). This learning function is called the learning function UNIS. In that case, a set of input data is defined as incomplete when it fails to deliver a complete final output (e.g. the geotechnical model fails when it is still in the construction stage).

Moreover, Zhaoyan et al. [2015] introduced the H learning function which is built based on information entropy theory. The information entropy of a metamodel $\hat{G}(x)$ describes the degree of disorder of $\hat{G}(x)$, and it can be used to quantitatively judge the uncertainty of $\hat{G}(x)$. The prediction is more certain when the information entropy is lower. Furthermore, some learning functions not only consider misclassification but also the influences of neighbor candidates (Sun et al. [2017]). However, these elaborate learning functions are less straightforward compared to the learning function U.

In this MSc thesis research, the FEM-based model is defined as a complete model, which means the evaluation is based on the model's final stage (no failure is assumed during the construction stages). Furthermore, since the chosen metamodel is AK-MCS (where every sample has an equal weight), the straightforward and more natural learning function U will be applied to this research. Other more elaborate learning functions have not shown any significant superior performances compared to the learning function U Bourinet [2018].

2.3 OPTIMIZATION METHOD

Reliability-based design optimization is a relatively rich and well-established field of research. In this section, we are reviewing some of the more recent and popular methods. There are a few approaches to formulating an RBDO process, one of the well-known formulations in engineering practice is (Dubourg et al. [2011]):

$$\mathbf{d^*} = arg \min_{\mathbf{d}\in\mathcal{D}} c(\mathbf{d}) \begin{cases} f_j(\mathbf{d}) \leq 0 & for\ j = 1, ..., n_f \\ \mathbb{P}(g_k(\mathbf{X}(\mathbf{d}), \mathbf{Z}) \leq 0) \leq \bar{P}_{f,k} & for\ k = 1, ..., n_g \end{cases} \tag{2.2}$$

Where $c(\mathbf{d})$ is the cost function to be minimized based on the design variable $\mathbf{d} = (d_1, ..., d_{n_d}) \in \mathcal{D}$, where n_d is the total number of design variables and $\mathcal{D}$ is the admissible domain for the design variables. Moreover, n_f is the number of constraints f_j (soft constraint) that bound the admissible design space in $\mathcal{D}$, and n_g is the total number of limit-state function g_k. Furthermore, $\mathbb{P}(g_k(\mathbf{X}(\mathbf{d}), \mathbf{Z}) \leq 0) \leq \bar{p}_{f,k}$ is the probability of failure constraint (hard constraint) of g_k with input vector $\mathbf{X(d)}$ and $\mathbf{Z}$ (for

design and environmental variables respectively) to be lower than the target probability of failure $\bar{P}_{f,k}$. Parameter **X(d)** is the input vector based on design variables **d**, and **Z** is the environment variables (independent variables other than the design variables).

In order to solve the RBDO problem defined in Equation 2.2, numerous methods have been developed (including classical and advanced techniques).

2.4 CLASSICAL RBDO METHODS

Based on the main solving strategy, the classical RBDO methods are commonly categorized into the followings (Aoues and Chateauneuf [2010]; Bourinet [2018]; Moustapha and Sudret [2019]):

1. **Double loop-approach.**
   Also known as the Two-level RBDO approach. In this method, the problem is solved in two nested optimization problems. The outer loop explores the design space (deals with cost optimization) while the inner loop solves the reliability assessment problem. Two of the well-known formulations in this category are Reliability Index Approach (RIA) (Enevoldsen and Sørensen [1994]) and the Performance Measure Approach (PMA) (Tu and Choi [1999]) approaches. However, this method requires enormous computational cost due to the performance functions to be evaluated and approximation errors due to limit state non-linearity (Aoues and Chateauneuf [2010]).

2. **Single loop-approach.**
   Also known as the Mono-level RBDO approach. This method alleviates the computational burden of the nested approach as in the Double loop-approach. It aims to solve the RBDO problem in a single loop. In order to do so, the probabilistic constraints are replaced by the optimality conditions or by reformulating the RBDO problem. One of the optimality conditions is Karush-Kuhn-Tucker (KKT) Kuschel and Rackwitz [1997]. Some of the well-known methods are the Single Loop Single Vector and Single Loop Approach (SLA) (Chen et al. [1997]; Liang et al. [2007]). However, based on a numerical benchmark carried by Aoues and Chateauneuf [2010], these methods (such as SLA and KKT) often fail to converge when the starting point of the optimization problem is far from the optimal solution. The lack of robustness in the single loop-approach (judged from the existing methods developed for this approach) is also observed when the target reliability indexes are large or when the design variables are the mean of the random parameters Moustapha and Sudret [2019].

3. **Decoupled approach.**
   The Decoupled approach performs optimization and reliability analysis sequentially, therefore, the reliability method is independent of the optimization algorithm. In fact, an approximate deterministic optimization problem is solved using information from a previous reliability analysis (Moustapha and Sudret [2019]). Some of the most popular methods in this category are reformulating the RBDO problem into a deterministic semi-infinite optimization problem (Royset et al. [2001]) and Sequential Optimization and Reliability Assessment (SORA) (Du and Chen [2004]), which transforms the RBDO problem into a sequence of deterministic optimization and reliability cycles that heavily relies on the inverse of FORM (as in PMA). Recently, Zhang et al. [2021] introduced a decoupled RBDO called Reliability Index Function Approximation by Adaptive Double-loop Kriging (RIFA-ADK) that use adaptive Gradient-enhanced Kriging (GEK) to take reliability sensitivity into account in addition to reliability index. The setbacks of the method are it is unsuitable for problems with a very small variation coefficient of the random design variable. However, decoupled approaches suffer similar drawbacks as the single loop-approaches. Approximation errors in early cycles may mislead the searching algorithm in the wrong direction, especially when the initial design is far from the optimal solution.

Since most of the classical methods heavily rely on approximation methods (e.g. FORM), they have low accuracy against problems that are highly non-linear or have multiple failure regions (despite there have been many contributions to improve their accuracy). However, since the recent developments of metamodeling techniques, the approximation methods in classical optimization methods can be replaced by metamodel-assisted simulation methods and further be improved.

2.5 ADVANCE METAMODEL-BASED SIMULATION RBDO METHODS

In order to overcome the well-known setbacks (especially inaccuracy) caused by the approximation methods, simulation methods can be applied. Numerous advanced simulation methods have been developed in the last decade, e.g. Stochastic Subset Optimization (SSO) (Taflanidis and Beck [2008]) that iteratively identifies subsets of the original design space with a high likelihood of containing the optimal design, and kernel density estimation to directly approximate the objective function instead of working with subset Jia and Taflanidis [2013]. These methods have generally brought a substantial gain in model evaluation savings, i.e., going from $10^8 - 10^9$ for a direct double loop-approach with crude MCS to $10^4 - 10^5$ calls to the performance function (Moustapha and Sudret [2019]). However, the number of function calls by these methods is still considered as expensive and time-consuming when one deals with a complex model (e.g. an FEM-based model). Therefore, further development in metamodel-based RBDO is needed to overcome such problems. An example of a metamodel-based RBDO application can be seen in Figure 2.3.

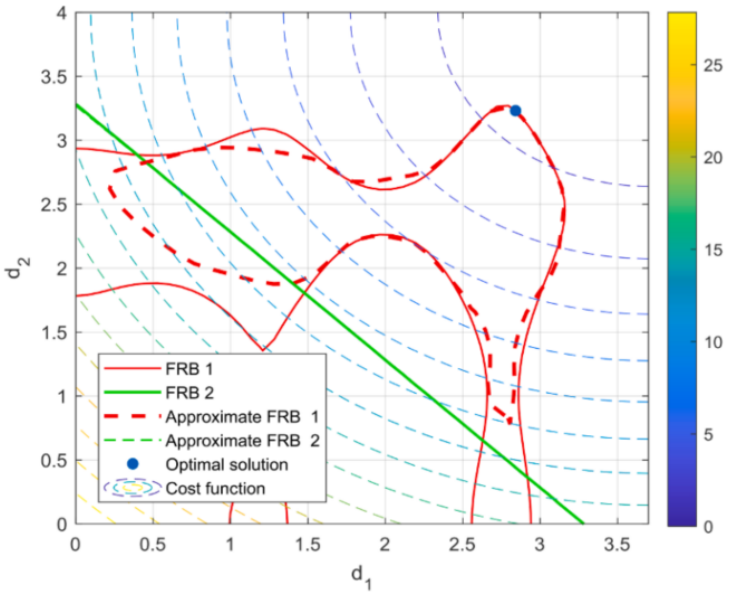

Figure 2.3: An illustration of a metamodel-based RBDO application (Zhang et al. [2021]). FRB = Feasible Region Boundary.

There are numerous approaches to combining a metamodel with an RBDO process depending on the various types of metamodel and optimization method. One of the approaches is to directly approximate the relationship between given design variables and the corresponding reliability index β (or probability of failure P_f). Some of the contributions to this approach can be seen in Foschi et al. [2002] and Lehký et al. [2018].

A less computational effort approach would be to directly create an approximation of the performance function and use it for reliability analysis. One of the obvious approaches is to build a distinct metamodel locally used for each reliability analysis in the inner loop of the two-level approach (to calculate the reliability index). Some of the contributions to this scheme are the double-loop approach with a second-order response surface model built around the MPP Agarwal and Renaud [2004] and the use of the neural network to compute an MCS in a double-loop approach Papadrakakis et al. [2005].

Another explored approach is building a single surrogate model that can be used to assess the failure probabilities considering multiple design choices, thus the main challenge is to build an accurate surrogate model over a large area. Some of the methods that use this approach are described in Chen et al. [2015] with a Kriging model for the whole design space combined with SORA, and Li et al. [2016] with Kriging combined with importance sampling for the whole design space. However, an optimal design is normally located in a smaller region (sub-region). Therefore, there is no need for a metamodel to be highly accurate globally. The most efficient approach would be to build a metamodel in a sub-region (often referred to as the *trust region*) of the space instead of the global one. Therefore, the metamodel could be specifically enhanced around this sub-region (instead of scattering the DOE in a wider but less important region). One of the important contributions to the trust region-based method

is the use of dynamic Kriging models on local windows associated with MCS for the RBDO problem's solution (Lee [1997]).

The augmented reliability space was introduced by Au [2005] to allow one to rigorously solve problems where all combinations of deterministic/random and design/environmental variables can be considered (Moustapha and Sudret [2019]). Therefore, the design and reliability space can be explored simultaneously by the metamodel. Some of the considerably important contributions are Dubourg et al. [2011] (that combines augmented space, adaptive Kriging, and subset simulation) and Moustapha et al. [2016] that formulates a quantile-based RBDO combined with (1+1) Covariance Matrix Adaptation-Evolution Strategy (CMA-ES) (Arnold and Hansen [2012]) and AK-MCS (focusing on the deviation number method developed by Echard et al. [2011]) that allows the design optimization into the direction of the space that decreases the cost function significantly (while still fulfilling the performance criteria).

The method developed by Moustapha et al. [2016] uses a two-stage DOE enrichment to reduce the Kriging epistemic uncertainty around the limit-state surface and improve the accuracy of the quantile estimates along the optimization iteration. It was successfully applied in the case of an automotive side-member subsystem under a frontal impact (in the field of a car body design). However, its application for high-dimensional cases (with more than 20 variables) still requires further work. Moreover, this method can not guarantee the solution to precisely converge in the correct optimal sub-region (despite managing to converge to a certain degree of precision depending on the number of iterations).

Furthermore, there is a tremendous amount of methods developed to solve RBDO problems. To accommodate these methods, Moustapha and Sudret [2019] have recently proposed a global and unified framework to solve a wide range of RBDO problems. This framework separates adaptive surrogate modeling, reliability analysis, and optimization into different non-intrusive blocks (the analyst could freely choose the method for each block independently with the other blocks).

The work of Moustapha et al. [2016], Arnold and Hansen [2012], and Echard et al. [2011] would be some of the main inspirations in developing a novel method to solve the problems faced in this research. The novel RBDO method introduced in this thesis aims to overcome the setbacks caused by the quantile-based method introduced in Moustapha et al. [2016].

2.6 DIFFERENCES BETWEEN THE CURRENT RESEARCH AND EXISTING WORKS

Some of the main differences between the research to be performed (the novel RBDO method) in this MSc thesis compared to the previous existing works are (more details in Chapter 3):

- AK-MCS performed in Echard et al. [2011] used a conservative stopping condition threshold for learning function U value, in this research those thresholds will be slightly relieved along the process to give the optimization process some "freedom" in searching for the optimal point.
- AK-MCS performed in Echard et al. [2011] evaluated analytical problems as the performance function, while this research uses a geotechnical engineering FEM model.
- AK-MCS performed in Echard et al. [2011] randomly generate the initial DOE population. This research uses a more evenly-spread initial DOE which is generated by using Latin Hypercube Sampling (LHS) or uniform distribution, and a user-defined preferred starting point.
- Quantile-based RBDO method applied in Moustapha et al. [2016] uses 2 enrichment stages, while this research uses a novel RBDO method with 4 stages to ensure the accuracy of the result (precise convergence). The optimization process in this research relies on different convergence criteria and approaches compared to the one in Moustapha et al. [2016]. Especially in the coupling of the reliability and optimization analysis.
- References on (1+1)-CMA-ES of Arnold and Hansen [2012] optimizes polynomial objective functions, while in this research, the application will be upon a geotechnical engineering FEM problem (where the (1+1)-CMA-ES application in the field is still relatively difficult to find).
- Some modifications of (1+1)-CMA-ES based on Arnold and Hansen [2012] are made to improve the coupled process of the reliability analysis-optimization scheme.

3 METHOD DEFINITION

As mentioned in Chapter 2, the proposed RBDO method in this research will be developed by using the works of Echard et al. [2011], Moustapha et al. [2016], and Arnold and Hansen [2012] as the starting point. This chapter will discuss how the proposed RBDO method is being defined by combining Kriging metamodeling, Quantile-based MCS, and (1+1)-CMA-ES.

3.1 RELIABILITY ANALYSIS

Reliability analysis is an analysis in calculating how much the probability of an unwanted event occurring. An unwanted event in this research is the failure of a geotechnical structure. Failure is evaluated through a limit state function (also known as a performance function), normally written as $g(X)$, where X is the random variables vector that has its own distributions. The limit state function is defined in such a way that a failure occurs when $g(X) < \bar{g}$, where $\bar{g}$ is a certain threshold (normally taken as zero or one in geotechnical engineering practice). The probability of failure (P_f) is determined by calculating the cumulative distribution function of $g(X) < \bar{g}$. Mathematically, the probability of failure can be written as equation (3.1) where $f_X(X)$ is the joint probability density function of $g(X)$.

$$P_f = \int_{g(X)<\bar{g}} f_X(X)\, dX \tag{3.1}$$

MCS (normally referred to as "crude MCS") is considered as the most robust method for calculating P_f, however, it requires an enormous number of function calls. The basic formulation of MCS in solving Equation 3.1 is described in Equation 3.2.

$$P_f = \frac{N_{failure}}{N_{mc}} \tag{3.2}$$

Where $N_{failure}$ is the total number of failures and N_{mc} is the size of the sampling population (or "MCS population"). Since P_f is normally a really small number, it is a very common way to express the reliability by a reliability index (β). The reliability index is defined as the inverse of the cumulative standard normal distribution function of P_f, as can be written in equation (3.3).

$$\beta = -\Phi^{-1}(P_f) \tag{3.3}$$

Moreover, the size of N_{mc} has to be large enough depending on the (expected) value of P_f to ensure that the result is deemed as accurate (or consistent) enough. The correlation between N_{mc} and P_f can be expressed in terms of the coefficient of variation δ_{P_f} (Equation 3.4). In this research, N_mc is deemed as large enough when $\delta_{P_f} \leq 0.1$.

$$\delta_{P_f} = \sqrt{\frac{1 - P_f}{N_{mc} P_f}} \leq 0.1 \tag{3.4}$$

3.2 KRIGING METAMODELING

As discussed in Section 2.1, the approximation of a real performance function $G(x)$ by a metamodel $\hat{G}(\boldsymbol{x})$ can be expressed by Equation 3.5.

$$\hat{G}(\boldsymbol{x}) \approx G(\boldsymbol{x}) \tag{3.5}$$

Such metamodel $\hat{G}(\boldsymbol{x})$ is defined as a Gaussian Process (GP) model if "for any $L \geq 1$ and any choice of $\boldsymbol{x}_1, ..., \boldsymbol{x}_L$, [...] the vector $(\hat{G}(\boldsymbol{x}_1), ..., \hat{G}(\boldsymbol{x}_L))$ has a multivariate normal distribution" (Santner et al. [2003]). Accordingly, $\hat{G}(\boldsymbol{x})$ is here written as a GP, defined by its mean function $m(\boldsymbol{x})$ (Equation 3.6) and its covariance function or kernel $k(\boldsymbol{x}, \boldsymbol{x'}|\theta)$ (Equation 3.7).

$$m(\boldsymbol{x}) = \mathbb{E}[\hat{G}(\boldsymbol{x})] \tag{3.6}$$

$$k(\boldsymbol{x}, \boldsymbol{x'}|\theta) = \mathbb{E}[(\hat{G}(\boldsymbol{x}) - m(\boldsymbol{x}))(\hat{G}(\boldsymbol{x'}) - m(\boldsymbol{x'}))] \tag{3.7}$$

such that the GP at a finite number of locations $\boldsymbol{x}$ is given by Equation 3.8.

$$\hat{G}(\boldsymbol{x}) = m(\boldsymbol{x}) + \boldsymbol{K}^{1/2}\xi \tag{3.8}$$

with kernel matrix $\boldsymbol{K} = k(\boldsymbol{x}, \boldsymbol{x'}|\theta)$ and standard normal multivariate $\xi \sim \mathcal{N}(0, \boldsymbol{I})$. Hyperparameters θ are the internal parameters that define the shape of the kernel (van den Eijnden et al. [2021]). For simple Kriging, such that $m(\boldsymbol{u}) = 0$ (where $\boldsymbol{u}$ is standard-normally distributed parameters), the performance function metamodel can be formulated as a GP with Gaussian prior shown in Equation 3.9.

$$\hat{G} = \hat{G}(\boldsymbol{u}) \sim \mathcal{N}(0, k(\boldsymbol{u}, \boldsymbol{u}|\theta)) \tag{3.9}$$

When a certain number of data on $\hat{G}(\boldsymbol{u})$ are known, the metamodel response vector can be split into known data $\hat{g}_t$ (subscript "t" means training, which is the DOE) and unknown data $\hat{g}_p$ (Subscript "p" means prediction) as can be formulated in Equation 3.10 (van den Eijnden et al. [2021]).

$$\begin{bmatrix} \hat{g}_p \\ \hat{g}_t \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} \boldsymbol{K}_{pp} & \boldsymbol{K}_{tp}^T \\ \boldsymbol{K}_{tp} & \boldsymbol{K}_{tt}^T \end{bmatrix}\right) \tag{3.10}$$

Rewriting leads to $\hat{g}_p \sim \mathcal{N}(\mu_{\hat{G}}, \sigma^2_{hatg})$, with the best estimate $\mu_{\hat{G}}$ (Kriging mean) and variance $\sigma^2_{\hat{G}}$ (Kriging variance) defined as Equations 3.11 and 3.12 (van den Eijnden et al. [2021]). Therefore, as the number of DOE increases, the size of matrix $\boldsymbol{K}$ will also increase in the order of 2. Consequently, the prediction takes longer to compute as the number of DOE increases.

$$\mu_{\hat{G}} = \boldsymbol{K}_{tp}^T \boldsymbol{K}_{tt}^{-1} \hat{g}_t \tag{3.11}$$

$$\sigma^2_{\hat{g}} = \boldsymbol{K}_{pp} - \boldsymbol{K}_{tp}^T \boldsymbol{K}_{tt}^{-1} \boldsymbol{K}_{tp} \tag{3.12}$$

The selection of kernel $k(\boldsymbol{u}, \boldsymbol{u}|\theta)$ can be based on the expected behavior of the approximated function and is generally expressed in terms of an a-priori variance and the Matèrn correlation function $\rho_M(\boldsymbol{u} - \boldsymbol{u'}|\theta, \mu)$, with θ representing the internal parameters (hyperparameters) and μ is a shape function controlling the GP smoothness. More explanations regarding Kriging metamodel can be found in Rasmussen and Williams [2006], Forrester et al. [2008], Pedregosa et al. [2011], van den Eijnden et al. [2021], Echard et al. [2011] and the references mentioned therein.

3.3 NOISY FEM AND METAMODEL RESPONSE

It is a known fact that FEM computation generates noisy responses due to numerical errors, limited allowed numbers of solver iterations, and the precision of complex numerical computation (among

others). Despite the scale of this error being small compared to the computed result, it can have a strong impact on the use of the model response (van den Eijnden et al. [2021]). In this research, noisy FEM responses may cause a serious problem in obtaining convergences. Noise in the response prevents convergence (of the Kriging optimization) and "confuses" the training algorithm. Therefore a problem such as overfitting in Kriging prediction could happen. To make matters worse, adding more DOE to a noisy FEM response leads to an increase in uncertainty due to overfitting. Such a situation does not occur in an analytical model, where adding more DOE actually reduces the uncertainty (Kriging variances).

To overcome the noisy response problem, a noise component will be added to the kernel. A noise component leads to a better representation of the underlying performance function and metamodel prediction that includes the uncertainty in the DOE (van den Eijnden et al. [2021]). The kernels (Matern and noise component) considered in this research are based on Rasmussen and Williams [2006]. An example of the use of a noise component in the kernel can be observed in Figures 3.1 and 3.2.

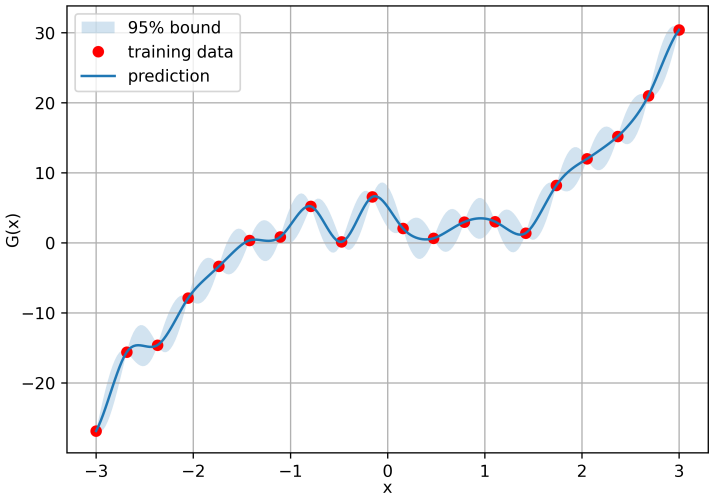

Figure 3.1: Metamodel prediction without a noise component of a noisy model response. There are some overfitting occurrences in the metamodel prediction because the metamodel is "forced" to make an exact prediction at the training data.

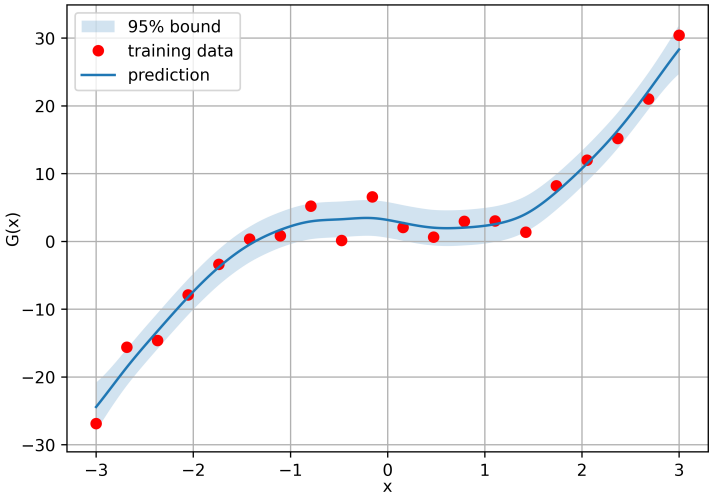

Figure 3.2: Metamodel prediction with a noise component of a noisy model response. Note that due to the noise component, the Kriging prediction is smoother because the prediction is allowed to have some "error tolerance" at the training data locations.

It can be seen from Figure 3.2 that the metamodel with a noise component could "smoothen" the prediction and ignores the "sudden jump" in the noisy training data (or performance function response). Meanwhile, overfitting occurred in 3.1 where all of the metamodel predictions are "forced"

to go through the noisy performance function response, and no "error tolerance" at the training data locations. Moreover, the noise term (k_{wn}) addition is defined as Equation 3.13 (van den Eijnden et al. [2021]. Unfortunately, in some cases, adding a noise component into the kernel may also heighten the Kriging variance (as can be observed in Figure 3.2. Therefore, the convergence criteria may be relaxed during the RBDO process to anticipate this "new minimum" Kriging variance.

$$k_{wn}(u, u') = \sigma_{wn}^2 \delta_{uu'} + k(u, u'|\sigma^2, \theta) \tag{3.13}$$

The noise variance σ_{wn}^2 is now added to the hyperparameter optimization scheme. The noise term is added to the kernel when it is applied to the training data (i.e. in matrix $\boldsymbol{K}_{tt}$ in Equation 3.12. More details regarding noisy metamodel response and the noise component application to a kernel can be found in Forrester et al. [2008] (for noisy metamodel response) and van den Eijnden et al. [2021] & Rasmussen and Williams [2006] (for noise component application in a kernel).

3.4 (1+1)-COVARIANCE MATRIX ADAPTATION-EVOLUTION SCHEME

The optimization method implemented in this research is based on the (1+1)-CMA-ES for constrained optimization by Arnold and Hansen [2012]. The main idea of the process is to approximate the directions of the local normal vectors of the constraint boundaries by accumulating steps that violate the respective constraints, and to then reduce variances of the mutation distribution in those directions (Arnold and Hansen [2012]). The target is to optimize (in this case, to minimize) the cost function (also known as the objective function) by accommodating the constraints defined in Equation 2.2. As explained in Section 2.3, these constraints consist of soft and hard constraints. Soft constraints normally define the domain of design variables (e.g. upper and lower limits) while the hard constraints are related to the reliability target. The cost function is a function that consists of design variables that are being optimized, thus its output is expected to be as optimum as possible (or in this case, to be as low as possible).

With each optimization iteration, an offspring (one set of design variables) is generated, thus it is called "(1+1)". The optimization continues through iterations until an offspring that fulfills the convergence criterion and satisfies the constraints defined in Equation 2.2 is found. The offspring $\boldsymbol{y}$ from each (1+1)-CMA-ES iteration is generated through Equation 3.14 below.

$$\boldsymbol{y} = \boldsymbol{x} + \sigma \boldsymbol{A} \boldsymbol{z} \tag{3.14}$$

Where $\boldsymbol{x}$ is the parental candidate solution (offspring of the previous iteration), σ is the global step size of the optimization strategy, $\boldsymbol{z} \in \mathbb{R}^n$ is a set of standard normally-distributed random numbers, n is the total number of design variables, and $\boldsymbol{A}$ is an $n \times n$ Cholesky decomposition matrix of a covariance matrix $\boldsymbol{C}$ such that $\boldsymbol{C} = \boldsymbol{A}\boldsymbol{A}^T$. Since matrix $\boldsymbol{C}$ has to be positive definite, an identity matrix with a size of $n \times n$ can be used as an initial matrix $\boldsymbol{C}$, therefore, the initial matrix $\boldsymbol{A}$ would also be an $n \times n$ identity matrix. Based on Equation 3.14, it can be seen that the generation of a new offspring $\boldsymbol{y}$ is carried by updating σ and matrix $\boldsymbol{A}$.

Moreover, the optimization process of (1+1)-CMA-ES can be observed by a simplified flowchart as displayed in Figure 3.3, and a more elaborate explanation can be found in Algorithm 1 (Appendix A).

This optimization process has to start from a feasible set of initial design variables. However, throughout the iteration process, some offspring $\mathbf{y}$ may fall into unfeasible regions in some iterations. When such things happen, the (1+1)-CMA-ES will perform some sort of a "course correction" through constraint evaluation in steps 3 and 4 in Figure 3.3. Each step of Figure 3.3 is further explained in Sections 3.4.1 to 3.4.10.

3.4.1 Step 1

Generation of an offspring $\boldsymbol{y}$ as explained in Equation 3.14.

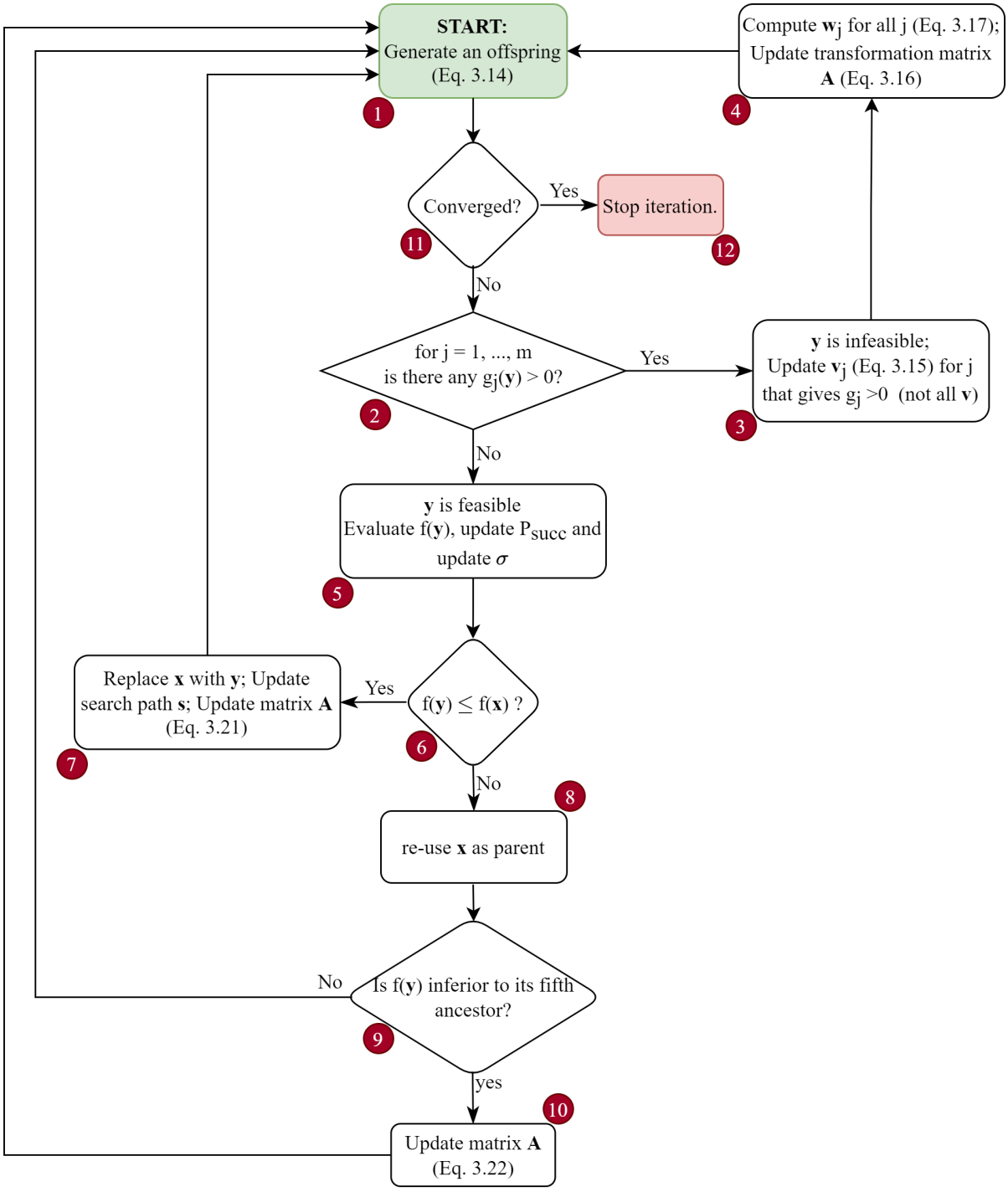

Figure 3.3: The flowchart of (1+1)-CMA-ES optimization process.

3.4.2 Step 2

Check if the current offspring $\boldsymbol{y}$ is feasible by evaluating the constraint functions $g_j(\boldsymbol{y})$ (where m is the total number of constraints (both soft and hard constraints). By the default setting, $\boldsymbol{y}$ is unfeasible if at least one of the constraint functions is larger than zero ($g_j(\boldsymbol{y}) > 0$).

3.4.3 Step 3

If offspring $\boldsymbol{y}$ is unfeasible, constraints vector $\boldsymbol{v}_j$ will be updated by using Equation 3.15. Note that the update is only needed for $\boldsymbol{v}_j$ element that corresponds to its particular constraint function $\boldsymbol{g}_j$.

$$\boldsymbol{v}_j \leftarrow (1 - c_c)\boldsymbol{v}_j + c_c \boldsymbol{A}\boldsymbol{z} \tag{3.15}$$

3.4.4 Step 4

Moreover, matrix $\boldsymbol{A}$ will be updated by using Equation 3.16.

$$\boldsymbol{A} \leftarrow \boldsymbol{A} - \frac{\beta}{\sum_{j=1}^{m} \mathbb{1}_{g_j(y)>0}} \sum_{j=1}^{m} \mathbb{1}_{g_j(y)>0} \frac{\boldsymbol{v}_j \boldsymbol{w}_j^T}{\boldsymbol{w}_j^T \boldsymbol{w}_j} \tag{3.16}$$

with

$$w_j = \boldsymbol{A}^{-1} v_j \tag{3.17}$$

Where $\mathbb{1}_{g_j(y)>0}$ equals one if $\boldsymbol{g_j(y)} > 0$ and zero otherwise. Since offspring $\boldsymbol{y}$ is unfeasible here, the iteration will be stopped after step 4 and a new iteration will be started from step 1. In the new iteration, an offspring will be generated by using the same $\boldsymbol{x}$ from the previous (unfeasible) iteration with an updated matrix $\boldsymbol{A}$.

3.4.5 Step 5

If the $\boldsymbol{y}$ evaluated in step 2 is feasible, the success probability estimate P_{succ} and the global step size σ will be updated by Equations 3.18 and 3.19 respectively.

$$P_{succ} \leftarrow (1 - c_P)P_{succ} + c_P \mathbb{1}_{f(\boldsymbol{y}) \leq f(\boldsymbol{x})} \tag{3.18}$$

$$\sigma \leftarrow \sigma \exp\left(\frac{1}{d} \frac{P_{succ} - P_{target}}{1 - P_{target}}\right) \tag{3.19}$$

Where $\mathbb{1}_{f(\boldsymbol{y}) \leq f(\boldsymbol{x})}$ equals one if $f(y) \leq f(x)$ and zero otherwise.

3.4.6 Step 6

Check if $f(\boldsymbol{y}) \leq f(\boldsymbol{x})$. Since the target of this research is to find the lowest cost function, therefore it is considered a "success" (or $\boldsymbol{y}$ is "superior" over $\boldsymbol{x}$) if the offspring $\boldsymbol{y}$ gives a lower cost function value than its parent $\boldsymbol{x}$ (a "success" when $f(\boldsymbol{y}) < f(\boldsymbol{x})$).

3.4.7 Step 7

If $f(\boldsymbol{y}) \leq f(\boldsymbol{x})$, offspring $\boldsymbol{y}$ will be used as a "parent" $\boldsymbol{x}$ for the next iteration. Moreover, the search path $\boldsymbol{s}$ and matrix $\boldsymbol{A}$ will be updated by using Equations 3.20 and 3.21 respectively. The iteration stops here.

$$\boldsymbol{s} \leftarrow (1-c)\boldsymbol{s} + \sqrt{\boldsymbol{c}(2-\boldsymbol{c})}\boldsymbol{A}\boldsymbol{z} \tag{3.20}$$

$$\boldsymbol{A} \leftarrow \sqrt{1-c_{cov}^{+}}\boldsymbol{A} + \frac{\sqrt{1-c_{cov}^{+}}}{||\boldsymbol{w}||^2}\left(\sqrt{1+\frac{c_{cov}^{+}||\boldsymbol{w}||^2}{1-c_{cov}^{+}}}-1\right)\boldsymbol{s}\boldsymbol{w}^T \tag{3.21}$$

3.4.8 Step 8 & 9

If $f(\boldsymbol{y}) > f(\boldsymbol{x})$, $\mathbf{x}$ will be re-used as the parent variable in the next iteration. moreover, check if $f(\boldsymbol{y})$ is "inferior" to its fifth ancestor. In this case, if the fifth ancestor has a lower cost function than $f(\boldsymbol{y})$, the current $\boldsymbol{x}$ will be re-used as $\boldsymbol{x}$ of the next iteration (the current offspring $\boldsymbol{y}$ will be ignored as it is deemed as "unsuccessful" compared to its fifth ancestor). The fifth ancestor or the fifth "grandparent" is the fifth most recent offspring. Similarly, if $\boldsymbol{y}$ is an offspring, then $\boldsymbol{x}$ is the first ancestor (or simply a "parent"). If $f(\boldsymbol{y})$ is "superior" to its fifth ancestor, then the iteration will be stopped here.

3.4.9 Step 10

If $f(\boldsymbol{y})$ is indeed "inferior" to its fifth ancestor, then matrix $\boldsymbol{A}$ will be updated by using Equation 3.22.

$$\boldsymbol{A} \leftarrow \sqrt{1+c_{cov}^{-}}\boldsymbol{A} + \frac{\sqrt{1+c_{cov}^{-}}}{||\boldsymbol{z}||^2}\left(\sqrt{1-\frac{c_{cov}^{-}||\boldsymbol{z}||^2}{1-c_{cov}^{-}}}-1\right)\boldsymbol{A}\boldsymbol{z}\boldsymbol{z}^T \tag{3.22}$$

3.4.10 Step 11

After a certain number of iterations, the (1+1)-CMA-ES will come to a sub-region where the optimal design point is located (optimal sub-region). When the iteration is at (or around) this optimal sub-region, the difference between the parent parameter $\mathbf{x}$ and offspring parameter $\mathbf{y}$ is very small (also the case with some other parameters, e.g. global step size σ). Therefore, in this sub-region, the cost function differences between iterations are insignificant. At this point, the user could decide on when to stop the optimization process based on this cost function difference.

The optimization process will be stopped when the cost function difference f_Δ is lower than threshold f_{stop} (Equations 3.23 and 3.24). Based on the author's practice, the value for f_{stop} can be taken in a range of 10^{-5} to 10^{-10} (see Section 3.4.13).

$$f_\Delta = \frac{|f(\boldsymbol{y}) - f(\boldsymbol{x})|}{f(\boldsymbol{x})} \tag{3.23}$$

Optimization process is converged when:

$$f_\Delta \leq f_{stop} \tag{3.24}$$

3.4.11 Supporting parameter settings

Moreover, the supporting parameters mentioned in Sections 3.4.1 to 3.4.10 are defined as the following Equations 3.25 to 3.32.

$$d = 1 + \frac{n}{2} \tag{3.25}$$

$$c = \frac{2}{n+2} \tag{3.26}$$

$$c_P = \frac{1}{12} \tag{3.27}$$

$$P_{target} = \frac{2}{11} \tag{3.28}$$

$$c_{cov}^{+} = \frac{2}{n^2 + 6} \tag{3.29}$$

$$c_{cov}^{-} = \min\left(\frac{0.4}{n^{1.6} + 1}; \frac{1}{2||z||^2 - 1}\right) \tag{3.30}$$

$$c_c = \frac{1}{n + 2} \tag{3.31}$$

$$\beta = \frac{0.1}{n + 2} \tag{3.32}$$

However, Equation 3.30 would become a problem if the (standard-normal randomly) generated value of z is small enough such that Equation 3.30 would return a negative value. Especially if the resulting c_{cov}^{-} is lower than (-1), because it will make a few terms on the right-hand side of Equation 3.22 to be irrational. Therefore, compared to the original (1+1)-CMA-ES proposed by Arnold and Hansen [2012], a new z will be re-sampled in this research whenever Equation 3.30 returns a negative c_{cov}^{-} value.

3.4.12 Iteration parameters

Since the optimization will be coupled with a reliability analysis method, the reliability analysis and metamodel accuracy assessment will be performed for every one iteration of the (1+1)-CMA-ES process. Therefore there are some parameters from the previous iteration that are needed for the next iteration. These parameters are the covariance matrix $\boldsymbol{A}$, global step-size σ, cost function value $c(\boldsymbol{d})$, constraint vector $\boldsymbol{v}$, the success probability estimate $\boldsymbol{P}_{succ}$, search path vector $\boldsymbol{s}$, the last fifth ancestor $f(\boldsymbol{d}^{i-5})$, offspring parameter $\boldsymbol{y}$, and parent parameter $\boldsymbol{x}$. Note that the "currently-evaluated" set of design variables $\boldsymbol{d}$ is the latest offspring $\boldsymbol{y}$. Moreover, in the original (1+1)-CMA-ES by Arnold and Hansen [2012], all of the iteration parameters are updated until the iteration ends. However, during the RBDO process, most of these parameters will be reset to their initial values whenever a DOE enrichment is performed. Furthermore, the last fifth ancestor $f(\boldsymbol{d}^{i-5})$ will be used as a "guidance" during the RBDO process (even though it was obtained from the optimization process with the "outdated" metamodel). A more elaborate explanation can be found in Section 3.5.4.

3.4.13 Convergence criterion

Based on the convergence criterion defined in Step 10 (Section 3.4.10, the accuracy of the optimization process can be determined. If f_{stop} is smaller, the optimization process will be more accurate. However, smaller f_{stop} will require more iteration and enrichment points (enrichment stages will be discussed from Section 3.5.3 onward. Thus smaller f_{stop} requires more time to conclude, especially when the number of DOE is getting higher (more DOE requires more time to optimize the metamodel's hyperparameter function). Moreover, a higher f_{stop} value may lead the (1+1)-CMA-ES optimization to converge faster. Early optimization convergence (due to a high value of f_{stop}) often leads to a false sub-region as it is still possible for the optimization to progress further.

It is found that the selection of f_{stop} value depends on the cost function formulation. For a cost function that gives a wider range of results, a slight change in the design variables may give a really small f_{Δ}. Therefore, a smaller f_{stop} is advised. However, for a cost function that gives a narrower range of results, it is advised to use a higher f_{stop}. For example, there are two different cost functions. The

cost function definitions are defined in Equation 3.33 (c_1 and c_2 for the first and second cost function example respectively). Note that c_1 has a narrower range of results compared to c_2.

$$\begin{cases} c_1 = 0.5x_1x_2 & for\ x_1 = 2 \leq x_1 \leq 7\ ;\ 0 \leq x_2 \leq 2 \\ c_2 = x_1x_2 & for\ x_1 = 150 \leq x_1 \leq 300\ ;\ 150 \leq x_2 \leq 300 \end{cases} \tag{3.33}$$

Examples of c_1 and c_2 cost function differences between iterations are shown in Tables 3.1 and 3.2.

Table 3.1: Cost function 1 (c_1) differences between iterations.

x_1	x_2	c_1	f_Δ (%)
5.02	1.22	3.06	-
5.03	1.23	3.09	1.0205

Table 3.2: Cost function 2 (c_2) differences between iterations.

x_1	x_2	c_2	f_Δ (%)
244.16	243.56	59467.61	-
244.17	243.57	59472.49	0.0082

It can be seen from Tables 3.1 and 3.2 that for small changes in x_1 and x_2 (with a magnitude of 0.01), c_1 gives more than 100 times higher f_Δ than c_2. Therefore, it is advised to use a higher f_{stop} for a similar problem as c_1 (e.g. $f_{stop} = 10^{-5}$) and a lower f_{stop} for a similar problem as c_2 (e.g. $f_{stop} = 10^{-8}$ or lower).

3.5 A FOUR-STAGE-ENRICHMENT RBDO METHOD

This chapter will discuss how the proposed RBDO method is defined, including the coupling of AK-MCS and (1+1)-CMA-ES.

3.5.1 Quantile-based formulation

The hard constraint from Equation 2.2 can be re-formulated as Equation 3.34 (Moustapha et al. [2016]).

$$\begin{aligned} \mathbb{P}(g(\boldsymbol{X}(\boldsymbol{d}), \boldsymbol{Z}) \leq 0) \leq \bar{P}_f \Leftrightarrow \mathbb{P}(G(\boldsymbol{X}(\boldsymbol{d}), \boldsymbol{Z} \geq \bar{g} \leq \bar{P}_f, \\ \Leftrightarrow \mathbb{P}(G(\boldsymbol{X}(\boldsymbol{d}), \boldsymbol{Z}) \leq \bar{g}) \geq 1 - \bar{P}_f \end{aligned} \tag{3.34}$$

Where $\bar{g}$ is the *upper* threshold of the real model performance function G (thus failure when $G > \bar{g}$). Meanwhile in geotechnical engineering practices, $\bar{g}$ is normally assigned as a *lower* threshold, valued as zero or one (e.g. lower threshold for the factor of safety), thus failure when $G < \bar{g}$. The last expression of Equation 3.34 can be further expressed as a quantile Q_α, as defined in Equation 3.35.

$$Q_\alpha(\boldsymbol{d}; G(\boldsymbol{X}(\boldsymbol{d}), \boldsymbol{Z})) = \inf\{q \in \mathbb{R} : \mathbb{P}(G(\boldsymbol{X}(\boldsymbol{d}), \boldsymbol{Z}) \leq q) \geq \alpha\} \tag{3.35}$$

where $\alpha = 1 - \bar{P}_{f,k}$. The quantile Q_α can be used as a measure of reliability given a target failure probability $\bar{P}_{f,k}$. Therefore, considering Equations 3.34 and 3.35, the hard constraint of Equation 2.2 can be re-expressed as Equation 3.36.

$$\mathbb{P}(g_k(\boldsymbol{X}(\boldsymbol{d}), \boldsymbol{Z}) \leq 0) \leq \bar{P}_{f,k} \Leftrightarrow Q_\alpha(\boldsymbol{d}; G(\boldsymbol{X}(\boldsymbol{d}), \boldsymbol{Z})) \leq \bar{g} \tag{3.36}$$

Based on Equation 3.36 above, Equation 2.2 can be re-defined as Equation 3.37 (where $\bar{g}_k$ acts as a *lower* threshold).

$$\boldsymbol{d}^* = \underset{\boldsymbol{d}\in\mathcal{D}}{arg\,min}\, c(\boldsymbol{d}) \begin{cases} f_j(\boldsymbol{d}) \leq 0 & for\, j = 1, \cdots, n_s \\ Q_{\alpha k}(\boldsymbol{d}; G_k(\boldsymbol{X}(\boldsymbol{d}, \boldsymbol{Z})) \leq \bar{g}_k & for\, k = 1, \cdots, n_h \end{cases} \tag{3.37}$$

where $\alpha_k = 1 - \bar{P}_{fk}$ and n_s & n_h are the total number of soft and hard constraints respectively. Moreover, the upper and lower boundary of the quantiles will be used as a measure of the metamodel's accuracy. The quantiles for each set of design variables $\boldsymbol{x}^{(i)}$ mentioned in Equation 3.37 can be calculated in the following steps.

1. Sample the Monte Carlo population for the initial design variable candidates:

$$\mathfrak{C}_q(\boldsymbol{d}^{(i)}) = \left\{ (\boldsymbol{x}^{(j)}, \boldsymbol{z}^{(j)}),\ j = 1, \cdots, N_{mc} \right\} \tag{3.38}$$

   Where $\boldsymbol{x}$ is the set of design variables to be analyzed (at i-th iteration), and $\boldsymbol{z}$ is distributed based on the corresponding environmental variables, $\boldsymbol{z} \sim f_{\boldsymbol{z}}$. Moreover, N_{mc} is the size of the Monte Carlo population. Note that each $\mathfrak{C}_q(\boldsymbol{d}^{(i)})$ has one combination of $\boldsymbol{x}$ and N_{mc} combinations of $\boldsymbol{z}$.

2. Compute the MCS output prediction of the performance function ($\boldsymbol{\mathcal{G}}_k$) for $\mathfrak{C}_q(\boldsymbol{d}^{(i)})$:

$$\boldsymbol{\mathcal{G}}_k = \left\{ \mathcal{G}_k^{(j)} = G_k(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(j)}),\ j = 1, \cdots, N_{mc} \right\} \tag{3.39}$$

3. Sort the MCS output prediction in ascending order such that $\mathcal{G}_k^{(1)} \leq \mathcal{G}_k^{(2)} \leq \cdots \leq \mathcal{G}_k^{(N_{mc})}$

4. The quantile $Q_{\alpha k}$ of $\mathfrak{C}_q(\boldsymbol{d}^{(i)})$ corresponding to the k-th constraint is defined as:

$$Q_{\alpha k}(\boldsymbol{d}^{(i)}; G_k(\boldsymbol{x}(\boldsymbol{d}^{(i)}), \boldsymbol{z})) \equiv q_{\alpha k}(\boldsymbol{d}^{(i)}) = \mathcal{G}_{k\lfloor n_\alpha \rfloor} \tag{3.40}$$

   Where $\lfloor n_\alpha \rfloor$ denotes the$\mathcal{G}$ response that yields the largest integer smaller than α. For example, if $N_{mc} = 100$ and P_f target is 5% (thus $\alpha_k = 95\%$), then $N_{mc} \times \alpha_k = 95$, therefore the quantile is the 95^{th} smallest (or the 6^{th} largest) MCS response within N_{mc}. The Monte Carlo population N_{mc} has to be large enough to ensure the calculated quantile $Q_{\alpha k}$ is accurate (Equation 3.4).

Note that each set of design variables $\boldsymbol{x}^{(i)}$ has its own Monte Carlo population $\mathfrak{C}_q(\boldsymbol{d}^{(i)})$ from the environmental variable $\boldsymbol{z}^{(i)}$ and a unique quantile $Q_{\alpha k}$ (including the lower and upper boundary of the quantile). Moreover, the use of metamodel will significantly reduce the computation burden in calculating the quantiles, therefore the performance function response G from Equation 3.39 will be replaced by metamodel response $\hat{G}$ (note that the metamodel responses consist of Kriging mean $\mu_{\hat{G}}$ and Kriging variance $\sigma_{\hat{G}}$). These quantiles will be further used to select optimal training data for the DOE and to estimate the accuracy of the metamodel upon certain sets of design variables.

3.5.2 Metamodel improvement throughout the optimization process

The metamodel will be improved by enriching the DOE. Therefore, it is important to select the best and most strategic training points for the DOE enrichment in order to get a reliable metamodel. Moreover, improving a metamodel throughout the entire performance function's domain space requires a large amount of DOE (thus many function calls are needed). Consequently, the RBDO process becomes expensive. Improving the metamodel throughout the whole performance function's domain (whole region) is also not really necessary since the optimal point (for the design variables) that gives the lowest cost function is located in a certain sub-region. Therefore, it is more important and efficient to improve the metamodel in this optimal sub-region rather than adding more DOE and function calls than necessary throughout the entire region. In order to do so, the metamodel will be enriched in four different stages, namely:

1. Global enrichment stage

2. First local enrichment stage
3. Second local enrichment stage
4. Starting point enrichment stage

The connection between each enrichment stage can be observed in Figure 3.5. A more elaborate explanation can be found in Algorithm 2 (Appendix B). Based on the figure, the global enrichment, first local enrichment, second local enrichment, and starting point enrichment stages are performed in steps 4, 7, 16, and 11 respectively.

Moreover, the whole enrichment stages are performed in an augmented space. In this space, every set of design variables' coordinate has its own environmental parameters' Monte Carlo population and its own unique quantile value. A brief depiction of an augmented space for a problem with two design variables ($\boldsymbol{d}_1$ & $\boldsymbol{d}_2$) and one environmental parameter ($\mathbf{z}$) is shown in Figure 3.4.

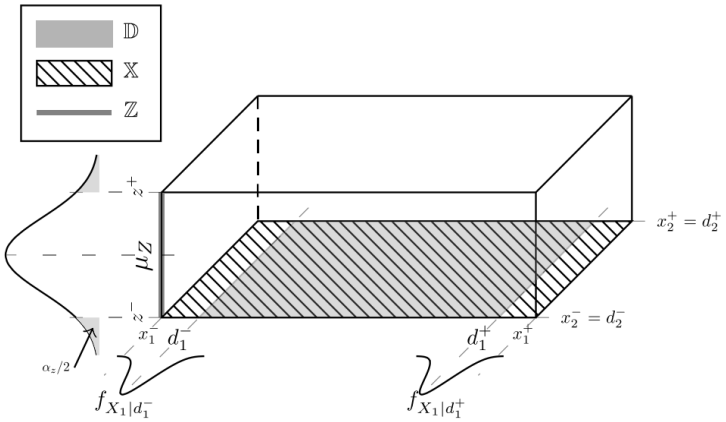

Figure 3.4: An illustration of augmented space (Moustapha and Sudret [2019]).

3.5.3 Global enrichment stage

After the initialization of the metamodel by using only several initial DOE, the metamodel is relatively inaccurate for the whole region of augmented space (or the whole performance function input region). The global enrichment aims to enrich the metamodel throughout the entire performance function input region (domain) with a less strict convergence criterion. Once it is globally converged to an acceptable less-strict criterion, the local enrichment and optimization process will be performed. Note that after the global enrichment, there is an expected residual uncertainty to the metamodel. This residual uncertainty will be reduced in the optimal sub-region by performing local enrichment. The global enrichment stage will be ended once the local enrichment stage began. The global enrichment is performed in the following steps:

1. Generate n_c random candidates for design variables (within the design variable domains):

$$\mathfrak{C} = \left\{ \boldsymbol{d}^{(1)}, \boldsymbol{d}^{(2)}, \ldots, \boldsymbol{d}^{(m)} \right\} \tag{3.41}$$

2. For each design $\boldsymbol{d}^{(i)}$, $i = \{1, \ldots, m\}$:
   (a) Generate the Monte Carlo population set required to compute the quantile:

$$\mathfrak{C}_q^{(i)} = \left\{ (\boldsymbol{x}_j, \boldsymbol{z}_j),\ j = 1, \ldots, N_{mc} \right\} \tag{3.42}$$

   (b) Compute the associated quantile $\hat{q}_\alpha\left(\boldsymbol{d}^{(i)}\right)$
   (c) Identify the input point of the quantile:

$$\left(\boldsymbol{x}_\alpha^{(i)}, \boldsymbol{z}_\alpha^{(i)}\right) = \left\{ (\boldsymbol{x}, \boldsymbol{z}) \in \mathfrak{C}_q^{(i)} : \hat{q}_\alpha\left(\boldsymbol{d}^{(i)}\right) = \mu_{\hat{G}}(\boldsymbol{x}, \boldsymbol{z}) \right\} \tag{3.43}$$

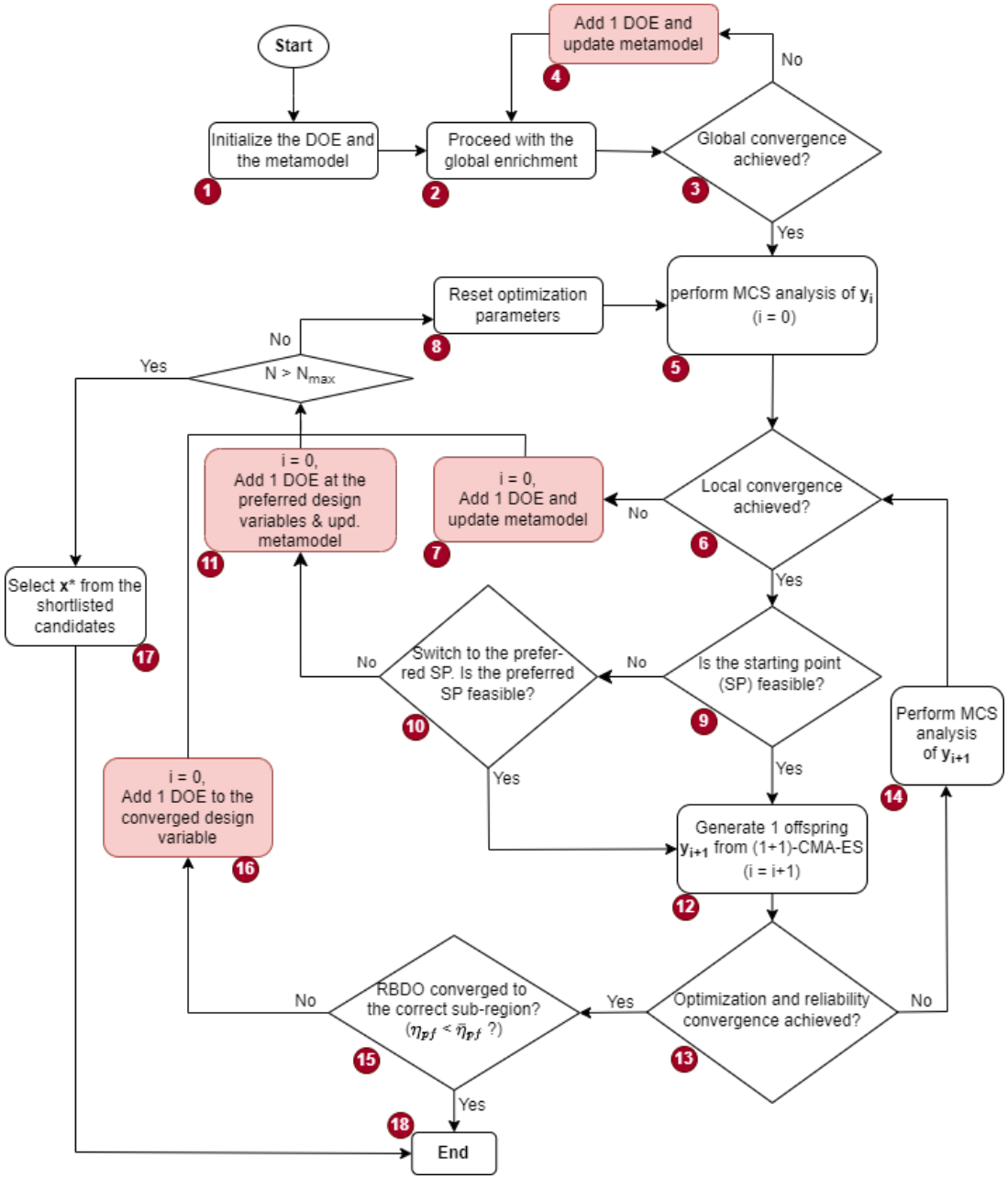

Figure 3.5: The general flowchart of the RBDO process. Enrichment stages are displayed in colored steps.

(d) Compute the modified deviation number based on U-learning function as mentioned in Equation 2.1:

$$\begin{aligned}\mathcal{U}\left(\boldsymbol{d}^{(i)}\right) &\equiv \boldsymbol{U}\left(\boldsymbol{x}_{\alpha}^{(i)}, \boldsymbol{z}_{\alpha}^{(i)}\right) \\ &= \frac{\left|\bar{g} - \mu_{\hat{G}}\left(\boldsymbol{x}_{\alpha}^{(i)}, \boldsymbol{z}_{\alpha}^{(i)}\right)\right|}{\sigma_{\hat{G}}\left(\boldsymbol{x}_{\alpha}^{(i)}, \boldsymbol{z}_{\alpha}^{(i)}\right)}\end{aligned} \tag{3.44}$$

3. The best point candidate to be added to the DOE is therefore defined as:

$$(\boldsymbol{x}_{next}, \boldsymbol{z}_{next}) = \arg \min_{(\boldsymbol{x}_{\alpha}^{(i)}, \boldsymbol{z}_{\alpha}^{(i)})} \mathcal{U}(\boldsymbol{d}) \tag{3.45}$$

Where $\mathfrak{C}_{\alpha} = \left\{\left(\boldsymbol{x}_{\alpha}^{(i)}, \boldsymbol{z}_{\alpha}^{(i)}\right),\ i = 1, ..., m\right\}$.

In the original U-learning function application by Echard et al. [2011], DOE enrichment stops when $min_{d\in\mathfrak{C}}\mathcal{U} \geq 2$. This convergence criterion is quite conservative since there is only a 5% chance of mistaking a safe design for a failed one (and vice-versa) with respect to all the points in $\mathfrak{C}$ (Moustapha et al. [2016]). However, since the main idea of global enrichment is only to get the metamodel just accurate enough for the optimization process to be applicable, but not too accurate therefore an efficient total number of function calls can be maintained. Moreover, the remaining "room for improvement" of the metamodel accuracy will be explored by focusing on the actual optimal sub-region where the optimal set of design variables resides. Therefore, the convergence criterion of $min\,\mathcal{U} \geq 2$ may be relaxed to a certain degree by defining it as Equation 3.46.

$$\eta = Card(\mathfrak{C}_2)/Card(\mathfrak{C}) \leq \bar{\eta}_{glo} \tag{3.46}$$

Where $Card(\mathfrak{C}_2)$ is the total number of $\mathcal{U}(d) \leq 2$ and $Card(\mathfrak{C})$ means the total number of design set candidates $\boldsymbol{d}$ in the global enrichment stages. Moreover, $\bar{\eta}_{glo}$ is the global convergence threshold which can be taken between 0.15 to 0.30 (note that the original criterion corresponds to $\bar{\eta}_{glo} = 0$). Note that each set of the design variables $\boldsymbol{d}$ has its own unique $\mathcal{U}$ value. The lower the global enrichment threshold $\bar{\eta}_{glo}$ is, the more function calls are needed to achieve the global enrichment convergence. Consequently, when Equation 3.46 is not fulfilled, $(\boldsymbol{x}_{next},\ \boldsymbol{z}_{next})$ from Equation 3.45 will be added as DOE.

3.5.4 First Local enrichment stage

After the global convergence is achieved, the metamodel is now deemed as globally-accurate enough while also having some residual uncertainties in some sub-region. These residual uncertainties can be dealt with altogether with the optimization process by updating the DOE only when (and where) necessary. The local enrichment will be performed in the following steps.

1. Select a starting point for the optimization process $(i = 0)$. It is advised to choose a starting point $(\boldsymbol{x_0}, \boldsymbol{z_0})$ from a relatively high cost function to ensure the feasibility of the starting point (especially from the probability constraint point of view) since the (1+1)-CMA-ES has to start from a feasible point. Note that $\boldsymbol{x_0}$ is equal to $\boldsymbol{d}^{(i=0)}$, which is the set of design variables at the start of the optimization process. However, during the optimization process $(i > 1)$, it is not a problem to have an iteration in the unfeasible points region.

2. Generate a Monte Carlo population for $\boldsymbol{d}^{(i)}$ based on the environmental parameters by following Equation 3.42.

3. Compute the metamodel MCS prediction $\hat{\boldsymbol{G}}_k$ by using Equation 3.39.

4. Compute the probability of failure $\hat{P}_f$ based on $\hat{\boldsymbol{G}}_k$ by using Equation 3.2.

5. Compute the associated quantiles and input points of the corresponding quantiles (Equations 3.40 and 3.43).

6. Compute the lower and upper boundary of the quantiles ($\hat{q}_\alpha^-$ & $\hat{q}_\alpha^+$) by utilizing the Kriging variance:

$$\begin{aligned} \hat{q}_\alpha^- &= \mu_{\hat{G}} - 1.96\sigma_{\hat{G}} \\ \hat{q}_\alpha^+ &= \mu_{\hat{G}} + 1.96\sigma_{\hat{G}} \end{aligned} \tag{3.47}$$

where the following relation holds:

$$\hat{q}_\alpha^-(\boldsymbol{d}) \leq \hat{q}_\alpha(\boldsymbol{d}) \leq \hat{q}_\alpha^+(\boldsymbol{d}) \qquad \textit{for any } \boldsymbol{d} \in \mathbb{D} \tag{3.48}$$

7. The spread of $\hat{q}_\alpha^-$ & $\hat{q}_\alpha^+$ interval can be used as a measure of the local Kriging accuracy for quantile estimation. Therefore, the local enrichment convergence criteria for $\bar{g} \neq 0$ can be defined as:

$$\eta_q(\boldsymbol{d}) = \frac{\hat{q}_\alpha^+ - \hat{q}_\alpha^-}{\bar{g}} \leq \bar{\eta}_q \tag{3.49}$$

Moreover, for $\bar{g} = 0$, the convergence criteria can be proposed as:

$$\eta_q(\boldsymbol{d}) = \frac{\hat{q}_\alpha^+ - \hat{q}_\alpha^-}{6 \times STD[\mathcal{G}_k]} \leq \bar{\eta}_q \tag{3.50}$$

where $STD[\mathcal{G}_k]$ is the Kriging standard deviation over the Monte Carlo population of $\boldsymbol{d}^{(i)}$. The denominator in Equation 3.50 is different than the original local enrichment criteria proposed by Moustapha et al. [2016]. In the original criteria, the denominator is defined as $\sqrt{Var[\mathcal{G}_{MCS}]}$. The denominator in the original criteria is not a fair comparison to the denominator for $\bar{g} \neq 0$ since the denominator in Equation 3.49 is spanning to a wider length compared to the denominator of Equation 3.50, thus achieving a local convergence by using Equation 3.50 is stricter and requires much more function calls.

Moreover, other convergence criteria can also be considered (where it is applicable regardless of what the $\bar{g}$ value is). For instance, one could also measure the lower and upper boundary of the reliability index predicted by the metamodel ($\hat{\beta}^-$ & $\hat{\beta}^-$) which are calculated in the same manner as Equation 3.47, and compare it to a certain threshold as defined by Equation 3.51. Similarly, the same method could also be applied to the probability of failure $\hat{P}_f$ (Equation 3.52).

$$\eta_q(\boldsymbol{d}) = \frac{|\hat{\beta}^+ - \hat{\beta}^-|}{\hat{\beta}} \leq \bar{\eta}_{\beta q} \tag{3.51}$$

$$\eta_q(\boldsymbol{d}) = \frac{|\hat{P}_f^+ - \hat{P}_f^-|}{\hat{P}_f} \leq \bar{\eta}_{P_f q} \tag{3.52}$$

where $\bar{\eta}_{\beta q}$ and $\bar{\eta}_{P_f q}$ is the local enrichment convergence threshold for $\hat{\beta}$ and $\hat{P}_f$ respectively. For reference, $\bar{\eta}_{P_f q}$ is taken as 0.05 in van den Eijnden et al. [2021]. However, $\bar{\eta}_{P_f q} = 0.05$ is a conservative threshold during an optimization process, which means a large number of DOE may be needed before the optimization process converges.

8. If the convergence criteria is not fulfilled, the first local enrichment will be performed by adding a training data U_{comp} into the DOE as defined in Equations 3.53 and 3.54 (as proposed in Moustapha et al. [2016]).

$$\boldsymbol{U}_l(\boldsymbol{x}_k, \boldsymbol{z}_k) = \frac{|\mu_{\hat{G}_l}(\boldsymbol{x}_k, \boldsymbol{z}_k) - \hat{q}_{\alpha_l}|}{\sigma_{\hat{G}_l}(\boldsymbol{x}_k, \boldsymbol{z}_k)} \qquad for\, l = 1, \ldots, n_h;\ k = 1, \ldots, N_{mc} \tag{3.53}$$

$$U_{comp}(\boldsymbol{x}_k, \boldsymbol{z}_k) = \min_{l \in 1, \ldots, n_h} \boldsymbol{U}_l(\boldsymbol{x}_k, \boldsymbol{z}_k) \tag{3.54}$$

Since this research only has one hard constraint, $n_h = 1$, therefore the subscript l in Equation 3.53 can be ignored. Alternatively, to reduce computation load, Equation 3.53 can be replaced by Equation 3.55. Because it does not matter what the U value is since it is more important to find which U value is the lowest, and what is the corresponding design and environmental variables ($\mathbf{x}$ and $\mathbf{z}$) that gives the lowest U value.

$$\boldsymbol{U}_l(\boldsymbol{x}_k, \boldsymbol{z}_k) = \frac{|\mu_{\hat{G}_l}(\boldsymbol{x}_k, \boldsymbol{z}_k) - \bar{g}|}{\sigma_{\hat{G}_l}(\boldsymbol{x}_k, \boldsymbol{z}_k)} \qquad for\, l = 1, \ldots, n_h;\ k = 1, \ldots, N_{mc} \tag{3.55}$$

Moreover, Equation 3.55 is identical to Equation 3.44 and is a more natural resemblance to the U-learning function formula proposed in Echard et al. [2011]. It is advised to use Equation 3.55 in finding the local training data because it puts into account the difference between Kriging prediction $\mu_{\hat{G}}$ and $\bar{g}$, therefore, enrichment tends to be made around the limit state border where the probability of having $\mu_{\hat{G}} < \bar{g}$ is relatively high. This research uses Equation 3.55.

Generally, it is found that the convergence criteria that use quantile value as defined in Equations 3.49 and 3.50 to be less strict than the convergence criteria defined in Equations 3.51 and 3.52. It is advised to use the convergence criteria by using quantile values as defined in Equations 3.49 and 3.50 because they are less strict and allow the (1+1)-CMA-ES to have more "freedom" in searching the optimal region. Note that a strict convergence criterion will "force" the (1+1)-CMA-ES to add more DOE along the search path, thus more optimization re-start (and time) are needed. This also tends to limit the sub-regions explored by the metamodel when searching for the local optimal region. Therefore, in this research, a combination of $\bar{\eta}_q$ values is used. For instance, $\bar{\eta}_q = 1$ when $i = 0$ until $i = 250$, $\bar{\eta}_q$ =0.5 when $i = 251$ until $i = 500$, and etc. Note that i is the iteration number, as briefly explained in Figure 3.5.

When the local enrichment convergence criterion is achieved, the iteration process of (1+1)-CMA-ES is started (or continued) as can also be seen in Figure 3.5. Note that the training data selection in the local enrichment is using the same concept as the U-learning function as previously defined in Equations 3.44 and 3.45. Therefore, the input point of U_{comp} will be added to the DOE as a new training point for the first local enrichment. Note that the U-learning function mentioned in Equations 3.53 and 3.54 is applicable for a problem with more than one probability constraint (when $n_h > 1$).

Moreover, when a local enrichment is performed, the (1+1)-CMA-ES optimization process needs to be "re-started" by re-setting the iteration parameters mentioned in Section 3.4.12 to their initial values except for the parent parameter ($\boldsymbol{x}$) and the last fifth ancestor ($f(\boldsymbol{d}^{(i-5)})$). Therefore, the iteration process will be re-started from the last ($\boldsymbol{x}$) and the previous ($f(\boldsymbol{d}^{(i-5)})$) will be used to help the decision-making of the new iteration (the last fifth ancestor's role can be seen in step 8 of Figure 3.3). The re-start of the optimization process after a local enrichment is needed to keep the constraint function consistent during the process. Due to a local enrichment process, the metamodel is improved, therefore some design points that are feasible in the earlier or "outdated" metamodel may be unfeasible in the improved or "updated" metamodel (and vice-versa), thus there will be an inconsistency in the constraint function (especially the probability constraint). The inconsistency in the constraint function will "confuse" the (1+1)-CMA-ES iteration process since it heavily relies on the "history" that is recorded through the iteration parameters.

Furthermore, in order to satisfy the local convergence criteria defined in Equations 3.49 & 3.50, it was found that some sub-regions require a considerable amount of local enrichment points for the same set of design variable $\boldsymbol{x}$ (especially when a noise component is added to the metamodel, which could increase the Kriging variance). This could "waste" a considerable amount of function calls at sub-regions that are less important or (later discovered to be) far from the optimal sub-region. To

overcome this issue and maintain function call efficiency, the maximum allowed number of first local enrichments at every $\boldsymbol{x}$ will be set to 5. This means for each $\boldsymbol{x}$ that still fails to satisfy Equations 3.49 & 3.50 even after being enriched by 5 DOE, the iteration will be stopped and re-started from the latest preferred starting point (will be discussed in Section 3.5.6).

3.5.5 Second local enrichment stage

The first local enrichment stage helps the (1+1)-CMA-ES process to converge at a certain point that the metamodel deems as the optimal point. Note that this convergence happens when the following criteria are fulfilled:

1. At least one of the equations between Equations 3.49 to 3.52 is satisfied (the first local enrichment stage is converged).
2. Equation 3.24 is satisfied (the (1+1)-CMA-ES optimization process is converged).

However, this point is converged based on the metamodel's performance with a certain number of DOE (based on a certain number of enrichments performed so far). For example, if the targeted probability of failure ($P_{f\ target}$) is 5%, the (1+1)-CMA-ES optimization process will converge to a point where the metamodel returns $\hat{P}_f$= 5% during the first stage of local enrichment. After the first stage of local enrichment, there is no guarantee yet if the converged point is located in the correct sub-region. Note that the correct sub-region is where the crude MCS simulation returns $P_f = P_{f\ target}$ (Not $\hat{P}_f = P_{f\ target}$!). Therefore, a convergence achieved from the first stage of local enrichment may mislead the optimal design parameters output or give a "false sense of security".

In order to overcome the problem, an additional enrichment stage is introduced, namely the second local enrichment stage. The main purpose of this stage is to confirm if the first local enrichment stage converged in the correct sub-region. If it is not, a DOE with the same design variable as the convergence point will be added. This DOE will also help the CMA-ES to avoid the same "mistake" (incorrect sub-region) in future iterations. Moreover, this stage could identify an incorrect sub-region by checking the Kriging variance (or standard deviation $\sigma_{\hat{G}}$) of the sub-region. Ideally, a correct sub-region would be where the metamodel returns $P_f = P_{f\ target}$ and has zero (or very low) Kriging variance at the same time. The application of the second local enrichment stage can be explained in the following steps.

1. After the (1+1)-CMA-ES converged in the first local enrichment stage, compute P_f^+ and P_f^- as described in Equation 3.56.

$$\begin{aligned} \hat{P}_f^+ &= \mathbb{P}(\hat{g}_k + 1.96\sigma_{\hat{G}}) < \bar{g} \qquad for\ k = 1, \ldots, N_{mc} \\ \hat{P}_f^- &= \mathbb{P}(\hat{g}_k - 1.96\sigma_{\hat{G}}) < \bar{g} \qquad for\ k = 1, \ldots, N_{mc} \end{aligned} \tag{3.56}$$

2. Check the discrepancy δ_{P_f} of $\hat{P}_f$ compared to $\bar{P}_{f,k}$ by using Equation 3.57. To avoid excessive deployment of function calls, only a converged point that has $\delta_{P_f} \leq 5\%$ is considered a second local enrichment DOE.

$$\delta_{P_f} = \frac{|\hat{P}_f - \bar{P}_{f,k}|}{\bar{P}_{f,k}} \leq 5\% \tag{3.57}$$

3. Check η_{P_f} as described in Equation 3.58.

$$\eta_{P_f} = \frac{|\hat{P}_f^+ - \hat{P}_f^-|}{\hat{P}_f} \leq \bar{\eta}_{P_f} \tag{3.58}$$

The η_{P_f} value describes the accuracy of the metamodel prediction $\hat{P}_f$ in the converged sub-region (the lower η_{P_f}, the less Kriging variance in the sub-region is and thus the more accurate the $\hat{P}_f$ is). Note that Equation 3.58 is identical to Equation 3.52. As mentioned in Section 3.5.4, the value

of $\bar{\eta}_{P_f}$ can be taken in a range of 0.05 to 0.10. Consequently, the lower the $\bar{\eta}_{P_f}$ value (stricter), the more DOE are needed. The RBDO process will be stopped when Equation 3.58 is satisfied and thus the corresponding set of design variables **X(d)** that satisfies the Equation will be regarded as the optimal design variables.

4. If Equations 3.58 and 3.57 are not satisfied, a new DOE that has the lowest U value will be added from the MCS population ($\mathfrak{C}_i$) of the currently converged design variables (x_i). This U value can be calculated by using the exact formula as in Equations 3.53 and 3.54.

Eventually, when the convergence criterion of the first local enrichment stage is fulfilled, the converged design variables to be analyzed in the second local enrichment stage are the same as the design variables in the first local enrichment stage. Therefore Equations 3.53 (or 3.55 and 3.54 can be re-used to add a DOE from the current design variables' MCS population.

3.5.6 Starting point enrichment stage

As mentioned in Section 3.4, the (1+1)-CMA-ES process has to start from a feasible starting point that satisfies all of the constraints defined in Equation 2.2. As also mentioned in step 1 of Section 3.5.4, the RBDO process starts from a user-defined preferred starting point ($\boldsymbol{x_0}, \boldsymbol{z_0}$). This starting point should satisfy the probability constraint of the problem. After the initialization of the metamodel (initial metamodel construction from the initial DOE), the metamodel naturally returns a feasible P_f at the preferred starting point design variables. However, after a certain number of enrichments, there is a possibility for over-fitting to occur in the Kriging prediction. This over-fitting may cause the metamodel to return an unfeasible P_f at the preferred starting point (which is unrealistic). When such a problem arises, enrichment at the starting point will be performed. This enrichment will add a DOE (or more) at the preferred starting point in the following manner:

1. Evaluate the U values from the MCS population of ($\boldsymbol{x_0}, \boldsymbol{z_k}$; for $k = 1, \cdots, N_{mc}$) by using Equation 3.55.
2. Select the point ($\boldsymbol{x_0}, \boldsymbol{z_k}$) that gives the lowest U value as the new DOE.

Since $\boldsymbol{x_0}$ will always be a constant (user-defined), the corresponding set of environmental variables $\boldsymbol{z_k}$ that returns the lowest U value is the most important data here. Note that this enrichment method is almost identical to the methods described in the previous enrichment stages, the only difference lies in the design parameters ($\boldsymbol{d} = \boldsymbol{x_0}$), which is a user-defined value.

Furthermore, as mentioned in Section 3.2, Kriging prediction takes longer to conclude as the DOE size increases (e.g. DOE > 150). When the MCS population size N_{mc} is also relatively high (e.g. $N_{mc} \geq$ 80,000), each optimization iteration process requires a considerable amount of time to conclude. Therefore, repeating the iteration from the initial starting point will be deemed time-inefficient despite the number of function calls remaining the same. To overcome such a problem, the preferred starting point will be updated every time an enrichment is made. For convenience, the preferred starting point will be updated to be the same as the newest enrichment point (from the first or second local enrichment). Therefore, there will be more than one candidate for the preferred starting point. However, if the updated starting point is becoming unfeasible at the start of (1+1)-CMA-ES iteration, the preferred starting point will be switched back to the previous feasible enrichment point, or to the initial preferred starting point if there is no feasible enrichment point left (starting point with $P_f > \bar{P}_f$ will be eliminated). To avoid further inefficiency, each set of design variable $\boldsymbol{x}$ is only allowed to be assigned once as an updated preferred starting point. This strategy could avoid a considerable amount of iteration from the initial preferred starting point.

Updating the preferred starting point is also advantageous because the optimization process could spend "more time" in exploring the "suspected" optimal sub-region, thus leading to more first local or second local enrichment around this suspected optimal sub-region, which ultimately leads to a higher chance of obtaining optimization convergence. To avoid having excessive and misleading preferred starting point candidates, only a point that has a relatively low η_{P_f} (e.g. $\eta_{P_f} < 20$) will be considered here. Re-starting optimization iteration from a point that has a high value of η_{P_f} will increase the possibility of having a high value of η_q at the start of the iteration, thus increasing the chance of "wasting" function calls at the starting point (to satisfy Equation 3.49 or 3.50).

Another strategy to save time and avoid having too many starting point enrichment is by assigning a less strict value of $\bar{\eta}_q$ at the starting point, e.g. $\bar{\eta}_q = 2.0$ when $i \leq 20$ and $\boldsymbol{x} = \boldsymbol{x}_0$. It is generally expected for the optimal design point to be located not too close to the starting point, therefore "wasting" function calls around the starting point sub-region is not really ideal. A high number of DOE enrichments is ideally performed around the (predicted or expected) optimal sub-region.

3.5.7 Maximum number of iterations and function calls.

In order to avoid an endless loops and keep the RBDO process time-efficient, the number of iterations i (as in Figure 3.5) and function calls N will be limited. When the number of maximum allowed iterations i_{max} is reached(in this research i_{max} is kept to 4000), a local enrichment will be made (first local enrichment stage) regardless of the reached η_q or f_Δ values. Therefore, the metamodel will be improved and the iteration should be able to converge faster with the newly updated metamodel. However, based on the author's experience, f_Δ is normally reached before approximately 1000-1500 iterations for f_{stop} value between 10^{-5} to 10^{-8}.

Moreover, when the maximum number of function calls N_{max} is reached, the RBDO process will conclude and select the best shortlisted design variable candidate as the final output. This selection is performed in the following steps:

1. Every time Equation 3.24 is fulfilled (or when the second local enrichment stage is about to be evaluated), the design variable $\boldsymbol{x}_j$ and its corresponding $\hat{P}_{f,j}$ will be recorded.
2. The discrepancy δ_{P_f} of $\hat{P}_{f,j}$ from $\boldsymbol{x}_j$ and $\boldsymbol{x}_{init}$ against the targeted $\bar{P}_{f,k}$ will be calculated by Equation 3.57.
3. If $\delta_{P_f} < 5\%$, $\boldsymbol{x}_j$ will be included in the shortlist.
4. After a few enrichments, there will be a few candidates of optimal design variable $\boldsymbol{x}$ in the shortlist.
5. When N_{max} is reached, $\bar{\eta}_q$ will be set to ∞. Therefore, the optimization process will find a converged point $\boldsymbol{x}_{last}$ based on the existing metamodel (based on N_{max} number of DOE). The $\hat{P}_f$ based on $\boldsymbol{x}_{last}$ will also be recorded to the same list altogether with $\boldsymbol{x}_j$ from the previous steps. A strategy could be devised to increase the possibility of finding the optimal point by setting $\bar{\eta}_q = \infty$ earlier, e.g. when $N = N_{max} - 2$, therefore the last 2 enrichment will be categorized as the second local enrichment (independent of $\bar{\eta}_q$).
6. The $\hat{P}_f$ and δ_{P_f} of all $\boldsymbol{x}_j$ and $\boldsymbol{x}_{init}$ in the shortlist will be evaluated with the updated metamodel (with N_{max} number of DOE).
7. The pair of design variable $\boldsymbol{x}_j$ that has the lowest δ_{P_f} will be selected as the final best design variables output.
8. In case there is more than one candidate of $\boldsymbol{x}_j$ that have the same δ_{P_f}, the final best design variables will be selected from $\boldsymbol{x}_j$ that has the lowest cost function.

4 METHOD IMPLEMENTATION

To give a better understanding of the proposed RBDO method (as defined in Chapter 3), its application will be discussed in this chapter. The RBDO method is performed through computer scripting using Python programming language. The Kriging metamodel is constructed using a modified version of Scikit-learn's Gaussian Process Regressor module (Pedregosa et al. [2011]), while the (1+1)-CMA-ES optimization and RBDO implementations are as Algorithms 1 and 2. The accuracy and efficiency of the RBDO method will be demonstrated on three well-known analytical (non-FEM) problems, therefore, its performance can be compared to other existing RBDO methods. Each example has different performance functions, cost functions, and constraint functions. Thus the generality of the proposed RBDO method can be investigated.

4.1 EXAMPLE 1: A SIMPLE COLUMN UNDER COMPRESSION

The problem is a simple rectangular column under a compression load F_{ser} with a rectangular cross-section $b \times h$ (as illustrated in Figure 4.1). This example is also used as one of the RBDO benchmark problems in Dubourg et al. [2011] and Moustapha et al. [2016]. The performance function of the column capacity is defined in Equation 4.1.

Figure 4.1: A simple illustration of Example 1 (b & h in mm).

$$g(\boldsymbol{x}, \boldsymbol{z}) = k\frac{\pi^2 E b h^3}{12L^2} \tag{4.1}$$

Where b & h is the dimension of the column, E is Young's elasticity modulus, L is the column's length, k is a factor that accounts for model uncertainty in the Euler force (to represent the effect of imperfections in the column geometry). Therefore, failure occurs when F_{ser} is bigger than the column capacity $g(\boldsymbol{x}, \boldsymbol{z})$, or in other words, failure when $g(\boldsymbol{x}, \boldsymbol{z}) < F_{ser}$ (which means $\bar{g} = F_{ser}$). The design variables are $\boldsymbol{x} = \{b, h\}^T$ and the environmental variables are $\boldsymbol{z} = \{k, E, L\}^T$. The aim here is to find $\boldsymbol{x}^* = \{\boldsymbol{b}^*, \boldsymbol{h}^*\}^T$, which are the optimal value of b & h (in mm) that give the lowest possible cross-section area of the column that still returns a target probability of failure of 5% ($\bar{P}_{f,k} = 5\%$). The distributions of the environmental parameters are listed in Table 4.1.

Table 4.1: Probabilistic parameter details of Example 1: a simple column under compression. Coefficient of variation, COV = standard deviation/mean.

Parameter	***Distribution***	***Mean*** (μ)	***COV*** (%)
k (-)	Lognormal	0.6	10
E (MPa)	Lognormal	10,000	15
L (mm)	Lognormal	3,000	1
F_{ser} (N)	Deterministic	1.4622×10^6	-

The problem is consisted of 2 design variables ($n = 2$) and 3 environmental variables ($z = 3$). Moreover, to quote Equation 2.2, the cost function c and constraint functions of the problem are defined in Equations 4.2 and 4.3 respectively.

$$c(b,\ h) = b \times h \tag{4.2}$$

subjects to:

$$\mathbf{x}^* = arg\ \min_{\mathbf{x}\in\mathcal{D}} c(\mathbf{x}) \begin{cases} h - b \leq 0 \\ 150\ mm \leq b \leq 350\ mm \\ 150\ mm \leq h \leq 350\ mm \\ \mathbb{P}(g_k(\mathbf{x}, \mathbf{z}) \leq 1) \leq 5\% \end{cases} \tag{4.3}$$

Based on Equation 4.3, there are 4 constraint functions ($m = 4$). The initial preferred starting point is chosen to be $(\boldsymbol{x_0}, \boldsymbol{z_0})$ = {b, h, k, E, L}^{T} = {325.1, 325.0, 0.6, 10000.0, 3000.0}^{T}, which is expected to fulfill all of the constraints defined in Equation 4.3. This preferred starting point may be updated during the RBDO process. Moreover, the MCS size is set to N_{mc} = 10,000 (in accordance to Equation 3.4). As for the initialization of (1+1)-CMA-ES, $\sigma_0 = 0.5$ is chosen. For the metamodel's kernel, Matérn 3/2 without noise component is considered here since the performance function is an analytical problem. The process will be started by selecting 10 initial DOE randomly (one of the points is located at the starting point) for initial metamodel construction. The inputs are summarized in Table 4.2. The results are presented in Table 4.3 and Figures 4.2 to 4.4.

Table 4.2: RBDO input summary of Example 1.

Parameter	***Input***
n	2
m	4
z	3
Metamodel kernel	Matérn 3/2 without noise component
$(\boldsymbol{x_0}, \boldsymbol{z_0})$	{325.1, 325.0, 0.6, 10000.0, 3000.0 }^{T}
Metamodel N_{mc}	10,000 (Expected δ_{P_f} = 0.044)
$\bar{P}_{f,k}$	5.00% (β = 1.645)
$\bar{\eta}_{glo}$ & $\bar{\eta}_{P_f}$	0.20 & 0.10
$\bar{\eta}_q$	{1, 0.5, 0.25, 0.1} for every 250 iteration
f_{stop}	10^{-8}
σ_0	0.5
$\bar{g}$	1.0
i_{max}	4000
N_{max}	100

Table 4.3: RBDO output summary of Example 1.

Parameter	***Output***
$\boldsymbol{x}^*$ (mm)	{238.454, 238.454}^{T}
Total N	23
Metamodel $\hat{P}_f$ ($N_{mc} = 10^5$)	5.00%
MCS P_f ($N_{mc} = 10^6$)	5.02%
MCS P_f error to $\bar{P}_{f,k}$	0.40%
Total iteration	3,450
Total function evaluation (metamodel)	3.45×10^7

Figure 4.2 shows the result of the RBDO process in the cost function contour. Each point in the contour represents a combination of b & h (note that each of these points has its own MCS population $\mathfrak{C}_i$). The figure also shows the position of the 10 initial DOE, where one of the points is selected to be located in the starting point. There are three global enrichment points (where the third one has the same

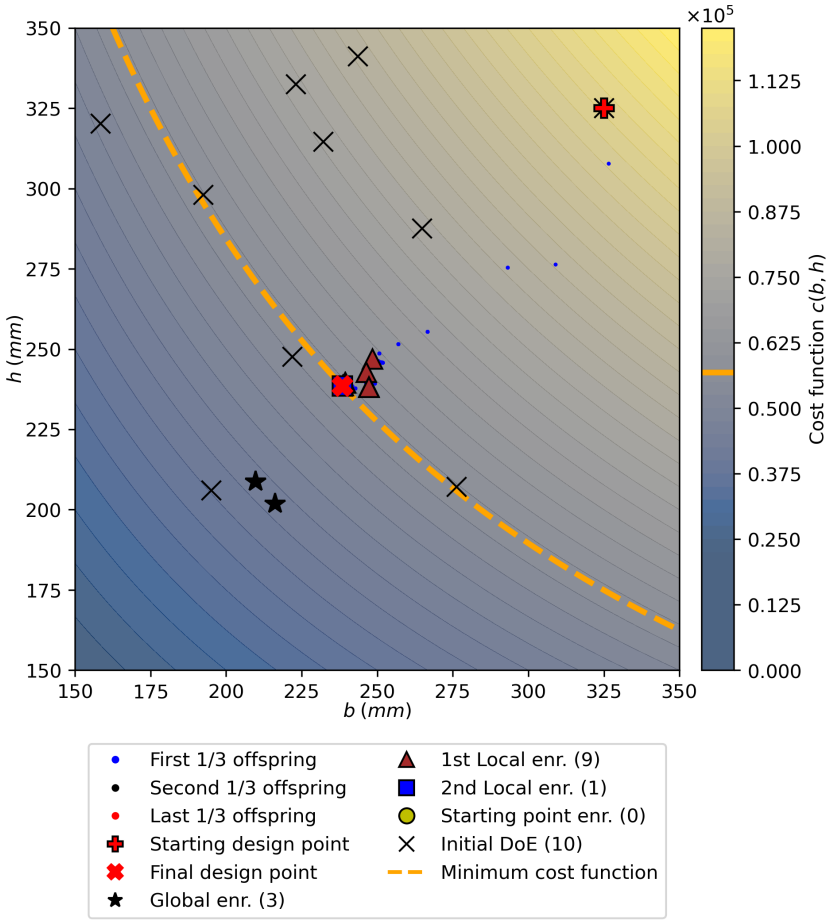

Figure 4.2: RBDO output of Example 1 with iteration history in cost function contours. The obtained $x^* = \{238.454$ mm, 238.454 mm$\}$ with the total number of function calls $N = 23$. Note that the last half of offspring were blocked by the enrichment markers because they started from an updated starting point.

design variables but different environmental variables, thus only 2 are visible in Figure 4.2 or 4.3), nine first local enrichment points, one second local enrichment point, and no starting point enrichment was made (which means the metamodel at the starting point was already feasible by only using the one point from the initial enrichment stage). Therefore, there are 23 enrichment points in total (23 function calls or $N = 23$). Note that a one-shot approach with a DOE of size 23 does not systematically converge to the optimal sub-region.

It can also be observed from Figure 4.2 that the first local enrichment was made along the "iteration track" of (1+1)-CMA-ES process and the iteration was continued from the last enrichment point. When it is not feasible to continue from the last enrichment point, the iteration will be restarted from the preferred starting point (x_0, z_0). However, in this case, thanks to the updated preferred starting point scheme as discussed in Section 3.5.6, the first local enrichment DOE was used as the updated preferred starting point close to the optimal sub-region. Therefore, no iteration points (offsprings) were visible from Figure 4.2 except for the first third of them, because they started from the last few local enrichments (the visibility was blocked by the local enrichment markers). As a consequence, the iterations were "forced" to spend more time around this "suspected" optimal sub-region, thus enhancing a higher probability of a faster convergence (with fewer function calls) at the optimal sub-region. Note that when an updated preferred starting point is no longer feasible, it will have one chance of performing a starting point enrichment. If it is still unfeasible after a starting point enrichment, it will be removed from the "preferred starting point list", except when there is only the initial predefined starting point that is left in the list.

The iteration points shown in Figure 4.2 (labeled as offspring) are grouped into 3 categories to show the timeline of the iteration, therefore the iteration track can be clearly seen (including the "retry", which is unfortunately not so clearly displayed in this case). For convenience, only the iteration tracks based on the last 2000 iterations are included (out of 3450 iterations). It can also be observed that the

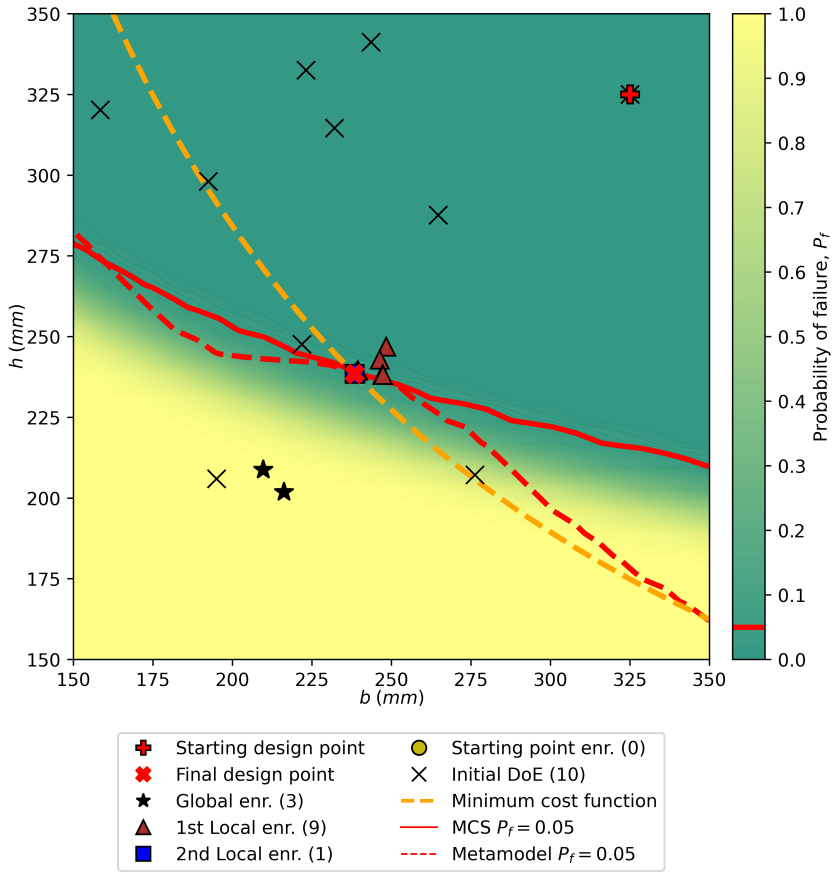

Figure 4.3: RBDO output of Example 1: enrichment points in crude MCS contour.

optimization process converged to an (turned out to be) incorrect optimal point (marked as the blue square in Figure 4.2 or 4.3) before it actually converged to the optimal sub-region (as marked by the red cross). Without the second local enrichment stage, the RBDO process may converge at the second local enrichment point (shown in Figure 4.3), which is an incorrect optimal sub-region (Fortunately in this example, the initial false prediction and the actual correct prediction of the optimal sub-region were located very close to each other). Therefore, the second local enrichment stage is very important in avoiding a "false sense of security" that could have been obtained by the RBDO process.

Figure 4.3 shows the enrichment points in the crude MCS's probability of failure $\hat{P}_f$ contours. Each b & h combination will return a $\hat{P}_f$ value. The red dashed line shows the contour where the metamodel returns $\hat{P}_f$ = 5%, and the red line is the contour where the crude MCS returns the actual P_f = 5%. The most ideal situation would be where the metamodel output is identical to the crude MCS output, or where the red dashed line lies at the exact same spot as the red line. Moreover, it can also be seen that the red dashed line deviates further from the red line at sub-regions farther from the converged point (red cross). This deviation happens due to the fact that the RBDO is only focusing on (enriching) the optimal sub-region, therefore no enrichment is made at the "less important" sub-regions. Note that the optimal sub-region is where the orange and red lines cross each other in Figure 4.3.

The RBDO process returns an optimal design $\boldsymbol{x}^*$ = {238.454 mm, 238.454 mm}^{T}. The analytical solution derived from Dubourg et al. [2011] gives $\boldsymbol{b}^*$ = $\boldsymbol{h}^*$ = 238.45 mm (Moustapha et al. [2016]). Moreover, the analytical solution is shown in Figures 4.2 and 4.3 as an orange dashed line, all b & h combinations along this orange dashed line will give an equal function value. Furthermore, at the converged final design point, the metamodel returns $\hat{P}_f$ = 5.00%. Meanwhile, at the converged design point, the crude MCS returns Pf = 5.02%. Note that there is a 0.40% discrepancy between $\hat{P}_f$ and P_f. This discrepancy can be further reduced by the following (among others):

- Lowering the $\bar{\eta}_{P_f}$ value (e.g. from 0.10 to 0.05).
- Increasing the maximum number of allowed iterations per set of DOE.
- Increasing the maximum number of allowed function calls (for a more complex problem).

- Increasing the MCS population N_{mc}.

However, lowering the $\bar{\eta}_{P_f}$ value will require more enrichment points, thus more function calls and iterations are also needed. In this example, a $\bar{\eta}_{P_f}$ value of 0.1 already gives a relatively accurate result compared to the analytical solution with an error of 0.40%. The evolution of $\bar{\eta}_q$, $\bar{\eta}_{P_f}$, and $\bar{\eta}_\beta$ (by using Equation 3.51) values are displayed in Figure 4.4.

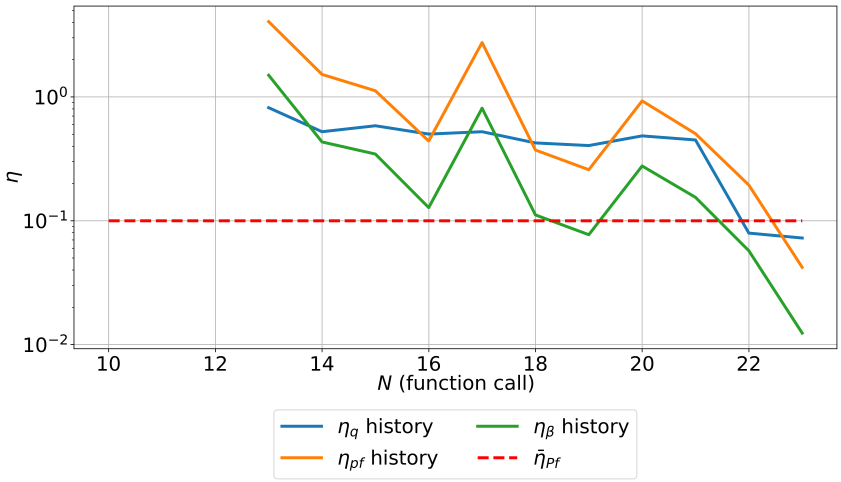

Figure 4.4: RBDO output of Example 1: evolution of $\bar{\eta}_q$, $\bar{\eta}_\beta$ and $\bar{\eta}_{P_f}$ values (y-axis in log. scale).

The η values recorded in Figure 4.4 are the η values of the last iteration before a new enrichment is made. Therefore, the record starts after the initial and global enrichment stage (metamodel initialization) since the iterations have not started yet during the initial and global enrichment stage. Based on Figure 4.4, all of the η values decrease with the increase in the number of function calls. However, there are a few sudden "jumps" in the η values along the iteration process. These jumps are caused when the $\hat{P}_f$ value of the last iteration is zero or really close to zero, therefore $\beta \approx \infty$. Or because the enrichment was made at a point that is far from the optimal sub-region. Figure 4.4 also shows that for the same number of N, the convergence parameters η_q, η_{P_f} and η_β have varying values between each other (e.g. when $\eta_q \approx 0.5$, $\eta_{P_f} \approx 3$). This shows that using η_{P_f} or η_β values in evaluating the convergence criteria would be more conservative (thus requires more N) compared to using η_q. Furthermore, the comparison of the result with other RBDO methods is displayed in Table 4.4.

Table 4.4: Example 1: result comparison with other RBDO methods.

Methods	$b^*(mm)$	$h^*(mm)$	$c(x)(mm^2)$	N	$c(x)$ ***error***$(\%)$
Analytical solution	238.450	238.450	56,858.40	-	-
Dubourg et al. [2011]	231.000	231.000	53,361.00	20	6.151
Moustapha et al. [2016]	239.120	239.120	57,178.37	18	0.563
This research	238.454	238.454	56,860.31	23	0.003

Since there is some randomness in the proposed RBDO method (e.g. the choice of MCS sample population or the generated (1+1)-CMA-ES' z value in Equation 3.14), the number of required N is varying between each run. Based on the author's experience, it takes approximately 17 to 30 numbers of N to get the converged (optimal) design output in this problem with the same settings, with the converged value of b^* & h^* ranging between 238.30 mm to 238.60 mm (cost function discrepancy ranging between 0.003% to 0.126%) for a chosen $\bar{\eta}_q = 0.1$. Table 4.4 shows that the proposed RBDO method gives a more accurate prediction of the optimal sub-region (less cost function discrepancy) compared to the method proposed in Moustapha et al. [2016] due to the second local enrichment stage (among other improvements). Furthermore, it is also worth noticing from Figure 4.2 that it takes 3,450 iterations to get the optimal sub-region. Therefore, since N_{mc} = 10,000, it takes 10,000 × 3,450 = 34,500,000 performance function evaluations to conclude. This would definitely take an enormous of time if a metamodel had not been used. Thanks to the metamodel, only 23 function calls are needed instead of 34,500,000!

To sum up, Figures 4.2 & 4.3 show that the main challenge in developing the RBDO method is to find the intersection between the orange dashed line and the red line by performing metamodel enrichment

efficiently in the best locations, therefore the correct optimal sub-region can be found with the fewest possible number of function calls.

4.2 EXAMPLE 2: A SHORT COLUMN UNDER OBLIQUE BENDING

This example deals with a short column structure with a rectangular cross-section $b \times h$ like in Example 1, only this time the column has a yield stress σ_y and is subject to an axial load F and biaxial bending moments M_1 & M_2 (Figure 4.5). The aim is to optimize b and h (or $\boldsymbol{x}^* = \{\boldsymbol{b}^*, \boldsymbol{h}^*\}^T$) to obtain the reliability target $\beta = 3$ (or $\bar{P}_{f,k} = 0.135\%$). The performance function of the model is defined in Equation 4.4. Note that the last term in Equation 4.4 ("+1") is added to accommodate $\bar{g} = 1$. Therefore failure occurs when $g(\boldsymbol{x}, \boldsymbol{z}) < 1$.

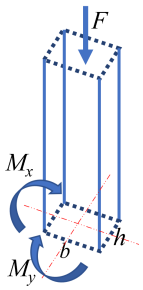

Figure 4.5: A simple illustration of Example 2 (b & h in mm).

$$g(\boldsymbol{x}, \boldsymbol{z}) = 1 - \frac{4M_1}{bh^2\sigma_y} - \frac{4M_2}{b^2h\sigma_y} - \left(\frac{F}{bh\sigma_y}\right)^2 + 1 \tag{4.4}$$

Therefore, the design variables are $x = \{b,\ h\}^T$ and the environmental variables are $\boldsymbol{z} = \{F,\ M_1,\ M_2,\ \sigma_y\}^T$ (thus $n = 2$ and $z = 4$). All of the design variable dimensions are in mm. The distributions of the environmental variables are displayed in Table 4.5.

Table 4.5: Probabilistic parameter details of Example 2: short column under oblique bending.

Parameter	***Distribution***	***Mean*(μ)**	***COV*** (%)
F (N)	Lognormal	2.5×10^6	0.20
M_1 (Nmm)	Lognormal	250×10^6	0.30
M_2 (Nmm)	Lognormal	125×10^6	0.30
σ_y (MPa)	Lognormal	40	0.10

The cost function and constraint functions are defined in Equations 4.5 and 4.6 respectively.

$$c(b,\ h) = b \times h \tag{4.5}$$

subjects to:

$$\mathbf{x}^* = arg\ \min_{\mathbf{x}\in\mathcal{D}} c(\mathbf{x}) \begin{cases} 100\ mm \leq b \leq 1000\ mm \\ 100\ mm \leq h \leq 1000\ mm \\ \mathbb{P}(g_k(\mathbf{x}, \mathbf{z}) \leq 1) \leq 0.135\% \end{cases} \tag{4.6}$$

Forty points of DOE will be generated to construct the initial metamodel. Since $\bar{P}_{f,k} = 0.135\%$, N_{mc} = 80,000 will be chosen to satisfy Equation 3.4 ($\delta_{P_f} \leq 0.1$). The value of $\bar{\eta}_{glo} = 0.20$ will be chosen to minimize global enrichment. The selected starting point is chosen at (x_0, y_0) = { 700, 700, 2.5×10^6, 250×10^6, 125×10^6, 40}. The rest of the settings are displayed in Table 4.6. The results are presented in Table 4.7 and Figures 4.6 to 4.8. The obtained optimal design variables are $\boldsymbol{x}^*$ = {328.630 mm, 595.150 mm}.

Table 4.6: RBDO input summary of Example 2.

Parameter	*Input*
n	2
m	3
z	4
Metamodel kernel	Matérn 3/2 without noise component
$(\boldsymbol{x_0}, z_0)$	$\{700, 700, 2.5\times 10^6, 250 \times 10^6, 125 \times 10^6, 40\}$
Metamodel N_{mc}	80,000 (Expected $\delta_{P_f} = 0.096$)
$\bar{P}_{f,k}$	0.135% ($\beta = 3.00$)
$\bar{\eta}_{glo}$ & $\bar{\eta}_{P_f}$	0.20 & 0.10
$\bar{\eta}_q$	$\{1, 0.5, 0.25, 0.1\}$ for every 250 iteration
f_{stop}	10^{-6}
σ_0	0.5
$\bar{g}$	1.0
i_{max}	4000
N_{max}	250

Table 4.7: RBDO output summary of Example 2.

Parameter	*Output*
$\boldsymbol{x}^*$ (mm)	$\{328.630, 595.150\}^T$
Total N	119
Metamodel $\hat{P}_f$ ($N_{mc} = 8 \times 10^4$	0.134%
MCS P_f ($N_{mc} = 10^6$)	0.135%
MCS P_f error to $\bar{P}_{f,k}$	0.07%
Total iteration	19,609
Total function evaluation (metamodel)	1.568×10^9

Figures 4.6 and 4.7 show that the optimal design variable contour line lies almost in parallel with the $\bar{P}_{f,k}$ target line. To overcome the challenge, as discussed in Section 3.5.6, the preferred starting point is updated to a few points (before it was removed if it is found to be unfeasible later on) closer to the "suspected" optimal sub-region. Figure 4.6 or 4.7 further shows that the second local enrichment points are "more crowded" around the optimal point compared to other sub-regions that have the same $\hat{P}_f$, thanks to the preferred starting point update. The optimization process concluded after 119 function calls. Moreover, Figure 4.7 also shows that the metamodel (shown as the red dashed line) was accurately constructed within the optimal sub-region, and is less accurate in sub-regions farther from the optimal point.

It can also be seen from Figure 4.7 that no global enrichment points were made. This is partly due to the locations and numbers of the initial DOE, which is chosen to be relatively high (40). In this case, where the P_f contours are parallel with the cost function contours, it was found that using less initial DOE may lead to many global and first local enrichment in less important sub-regions. Therefore, using more initial DOE may give the metamodel a "quicker judgment" at spotting a "suspected" optimal sub-region before "wasting" training data by investigating the (turned out to be) less important sub-regions. Further observation can be made from Figure 4.7 that all of the second local enrichment points lie along the $\bar{P}_{f,k}$ contour line, which means the RBDO method always converges at the targeted $\hat{P}_f$ contour line, which borders the feasible and unfeasible design domain. Some strategies that can be applied to obtain a faster convergence when the optimal cost function line lies almost in parallel with the probability constraint line are described below.

- As applied here, update the preferred starting point at a point closer to the "suspected" optimal point, but not too close to avoid starting at a (turned out to be) infeasible domain, as can be observed in Figure 4.7 (one of the yellow circle marks just outside the red line).
- Increase the $\bar{\eta}_q$ value to allow the optimization process to explore more possible optimal sub-region before it converges.
- Decrease f_{stop} value to delay the optimization convergence that allows the second local enrichment point to occur.

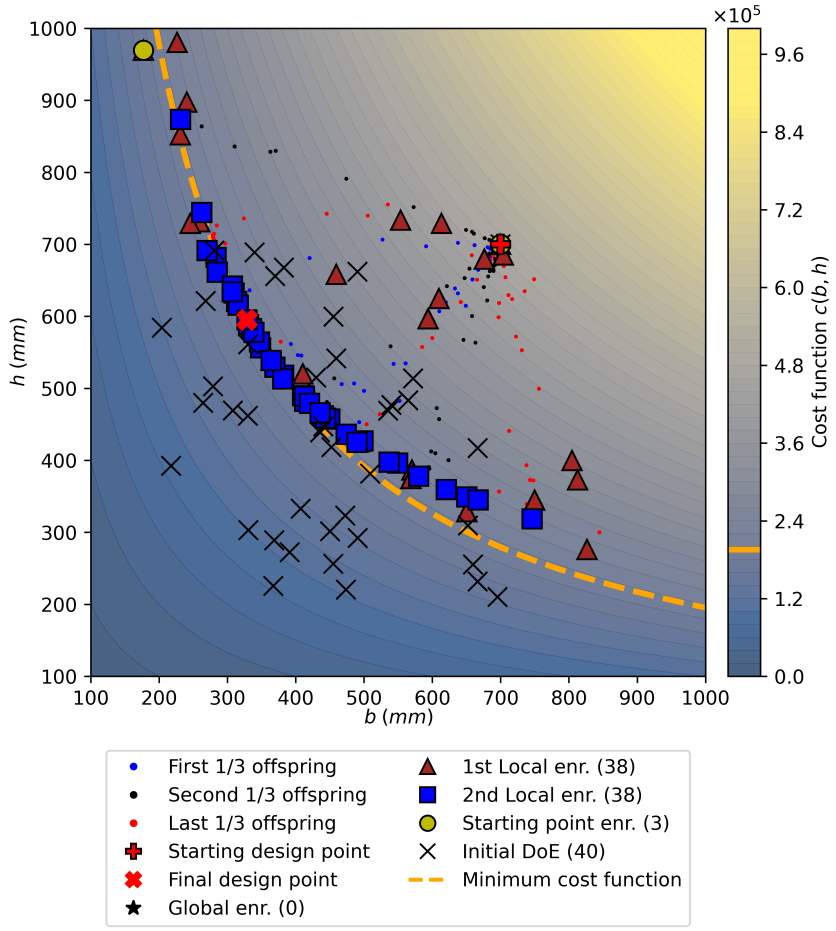

Figure 4.6: RBDO output of Example 2 with iteration history in cost function contours. The obtained $x^* = \{328.630 \text{ mm}, 595.150 \text{ mm}\}$ with total number of function calls $N = 119$.

- When it is possible/predictable, scatter the initial DOE around the expected optimal sub-region. Therefore, the required global and first local enrichment can be reduced.

Moreover, the convergence history of the problem is displayed in Figure 4.8. It can be seen from the figure that all of the η values are generally decreasing with the increase of N.

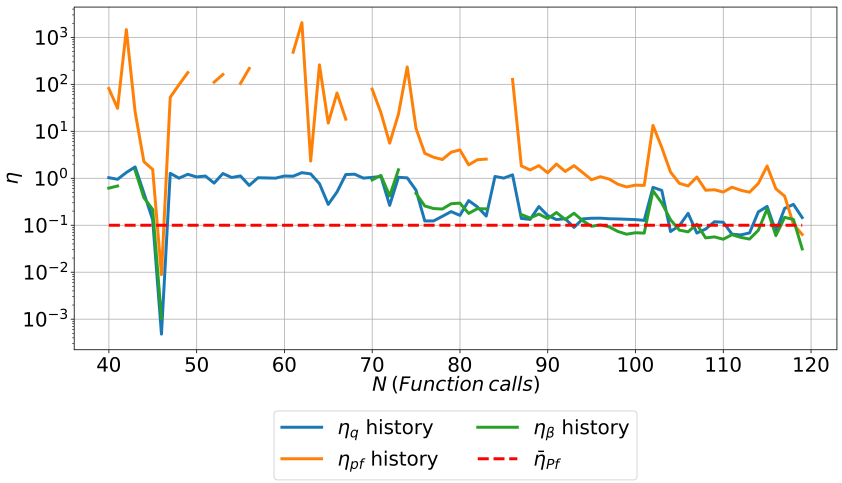

Figure 4.8: RBDO output of Example 2: evolution of $\bar{\eta}_q$, $\bar{\eta}_\beta$ and $\bar{\eta}_{P_f}$ values (y-axis in log. scale).

The comparison between the obtained result with the other RBDO methods is presented in Table 4.8.

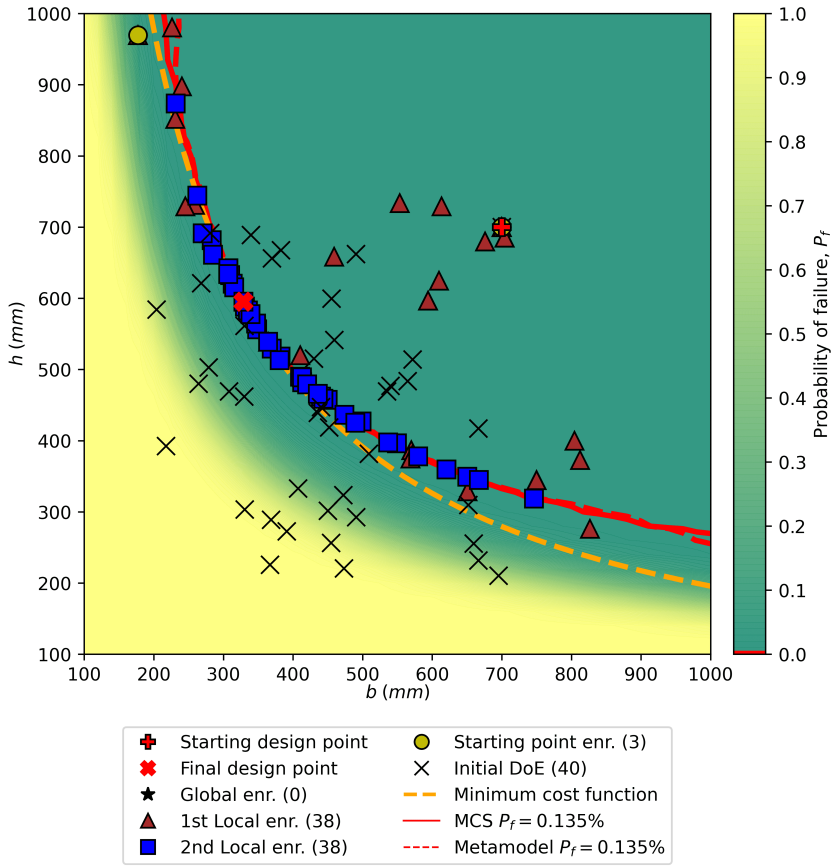

Figure 4.7: RBDO output of Example 2: enrichment points in crude MCS contour.

Table 4.8: Example 2: result comparison with other RBDO methods. Confirmed β values are obtained from a re-run by crude MCS with $N_{mc} = 10^6$, while β is the claimed reliability index value (from the sources). *a* = Lee and Jung [2008], *b* = Dubourg et al. [2011], and *c* = Moustapha and Sudret [2019].

Methods	$\boldsymbol{b}^*(mm)$	$\boldsymbol{h}^*(mm)$	$c(\boldsymbol{b}^*, \boldsymbol{h}^*)(mm^2)$	$\boldsymbol{N}$	β	Confirm. β
Nested FORM[a]	399	513	204,687	9,472	3.380	3.205
Meta-RBDO[b]	358	580	207,640	70	3.320	3.394
SVR+MCS+SQP[c]	332	596	197,872	84	3.000	3.084
Krig.+Subsim+CMAES[c]	336	592	198,912	86	3.000	3.135
Krig.+QMCS+CMAES[c]	331	592	195,952	57	3.000	3.018
This research	328.630	595.150	195,584	119	3.000	3.000

It can be seen from Table 4.8 that the proposed method gives the most accurate result with the lowest cost function despite requiring more number of function calls compared to other more popular Kriging-based RBDO methods.

4.3 EXAMPLE 3: A CANTILEVER SOIL RETAINING WALL UNDER SLIDING MODE OF FAILURE

The following example is taken from Honjo et al. [2009] and is also included in Example 5 of GEOSNet Le and Honjo [2008]. The problem consists of a gravity cantilever soil retaining wall against the sliding mode of failure as displayed in Figure 4.9. The original dimension $\boldsymbol{x}_{init}$={w, B}={0.4 m, 2.0 m} returns P_f =0.347% (β = 2.70). In this research, a "cheaper" design of the retaining wall $\boldsymbol{x}^*$ = {w*, B*} that still gives the same P_f will be investigated by applying the proposed RBDO method. The height of the wall and the slope of the fill are assumed to be the same, therefore only the thickness and the width of

the wall (w & B) are being optimized. The aim here is to show how the RBDO method can also be used to optimize an existing geotechnical engineering design.

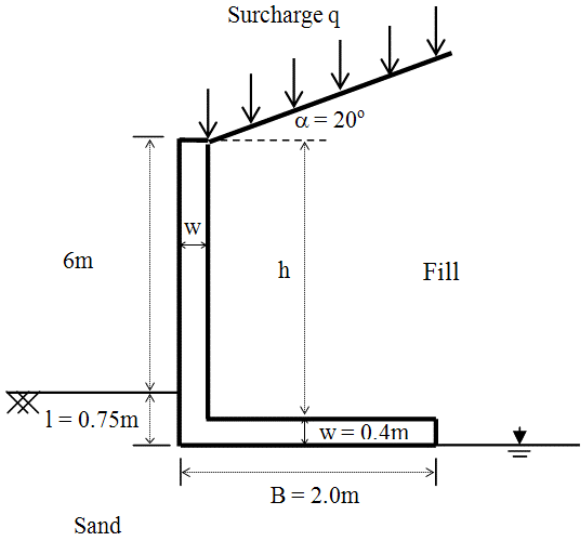

Figure 4.9: Example 3: The original problem definition with $\{w,\ B\}$={0.4, 2.0 m} (Le and Honjo [2008]).

The performance function (Factor of Safety (FS)) of the problem is defined in Equations 4.7 to 4.9.

$$g(\boldsymbol{x}, z) = \frac{R}{S} \tag{4.7}$$

Where:

$$R = \frac{1}{2} l^2 \gamma'_s \, tan^2\left(45° + \frac{\phi'_s}{2}\right) + \left\{(h+B)\, w\, \gamma_c + \left[(B-w)\, h + \frac{1}{2}(B-w)^2 tan\ \alpha\right]\gamma'_f + q\frac{(B-w)}{cos\ \alpha}\right\} tan\ \phi'_{bs} \tag{4.8}$$

$$S = \frac{1}{2}[(B-w)tan\ \alpha + h + w]^2 \gamma'_f tan^2\left(45° - \frac{\phi'_f}{2}\right) cos\ \alpha \tag{4.9}$$

Equation 4.7 basically defines the ratio between resistance R and solicitation (load) S of the soil retaining wall against the sliding mode of failure. Therefore, failure is defined when FS < 1 (thus $\bar{g} = 1$). The environmental parameters are defined in Table 4.9.

Table 4.9: Environmental parameter details of Example 3: a cantilever soil retaining wall.

Parameter	*Description*	*Distribution*	*Mean*(μ)	*COV* (%)
γ'_f (kN/m^3)	Fill unit weight above the wall	Lognormal	20	5
γ'_s (kN/m^3)	Sand unit weight beneath the wall	Lognormal	19	5
γ_c (kN/m^3)	Unit weight of the wall	Lognormal	25	5
$tan\ \phi'_f$ (-)	Tangent of int. frict. angle of the fill	Lognormal	0.781	14
$tan\ \phi'_s$ (-)	Tangent of int. frict. angle of the sand	Lognormal	0.675	13
$tan\ \phi'_{bs}$ (-)	Tan. of int. frict. angle of wall-sand	Lognormal	0.577	12
q (kN/m^2)	Surcharge load	Lognormal	15	30

The cost function to be optimized is the cross-section area of the soil retaining wall. The optimal design corresponds to the lowest cross section area that still gives the same P_f. Therefore, the cost function is defined in Equation 4.10 (all units are in m). For the dimension constraint, a lower boundary of 0.15 m is chosen for w. The constraint functions are defined in Equation 4.11.

$$c(w,\ B) = 6.75 \times w + (B - w) \times w \tag{4.10}$$

subjects to:

$$\mathbf{x^*} = arg\ \min_{\mathbf{x}\in\mathcal{D}} c(\mathbf{x}) \begin{cases} 0.15\ m \leq w \\ 0.00\ m < B \\ \mathbb{P}(g_k(\mathbf{x},\mathbf{z}) \leq 1) \leq 0.347\% \end{cases} \tag{4.11}$$

Moreover, the inputs and initial values for the analysis are presented in Table 4.10. The results are presented in Table 4.11 and Figures 4.10 to 4.12.

Table 4.10: RBDO input summary of Example 3. The initial starting point (x_0, z_0) corresponds to $\{\gamma'_f,\ \gamma'_s,\ \gamma_c,\ tan\ \phi'_f,\ tan\ \phi'_s,\ tan\ \phi'_{bs},\ q\}$).

Parameter	***Input***
n	2
m	3
z	7
Metamodel kernel	Matérn 3/2 without noise component
(x_0, z_0)	$\{1.0, 3.0, 20.0, 19.0, 25.0, 0.781, 0.675, 0.577, 15.0\}^T$
$x_{init}d$ Metamodel N_{mc}	80,000 (Expected δ_{P_f} = 0.059)
$\bar{P}_{f,k}$	0.347% (β = 2.7)
$\bar{\eta}_{glo}$ & $\bar{\eta}_{P_f}$	0.20 & 0.10
$\bar{\eta}_q$	{1, 0.5, 0.25, 0.1} for every 250 iteration
f_{stop}	10^{-6}
σ_0	0.1
$\bar{g}$	1.0
i_{max}	4000
N_{max}	150

Table 4.11: RBDO output summary of Example 3.

Parameter	***Output***
x^*	$\{0.150, 2.089\}^T$
Total N	61
Metamodel $\hat{P}_f$ (N_{mc} = 8 × 10^4	0.3445%
MCS P_f (N_{mc} = 10^6)	0.3466%
MCS P_f error to $\bar{P}_{f,k}$	0.028%
Total iteration	9,511
Total function evaluation (metamodel)	7.6088 × 10^8

The offsprings shown in Figure 4.10 are taken from the last 5,000 iterations, and only the last 1/3 of them can be seen. It means that the first 2/3 of the offsprings are located around the enrichment points because the optimization process re-used the parent variables most of the time (Step 8 of Figure 3.3).

It can be seen from Figures 4.10 and 4.11 that the cost function based on the RBDO output (1.30 m^2) is lower than the cost function based on the original design (3.34 m^2, marked as an orange diamond marker on an orange dotted line). It means that the RBDO gives an alternative design that gives the same P_f with 60.98 % less cross-section area! Therefore, the proposed RBDO method can be used in optimizing an existing design by locating the optimal design point that still gives the same reliability index β. Since there is more than one combination of design parameter to achieve a certain reliability target (Figure 4.11), the proposed RBDO is superior in accurately locating the optimal point compared to the traditional "trial-and-error" method.

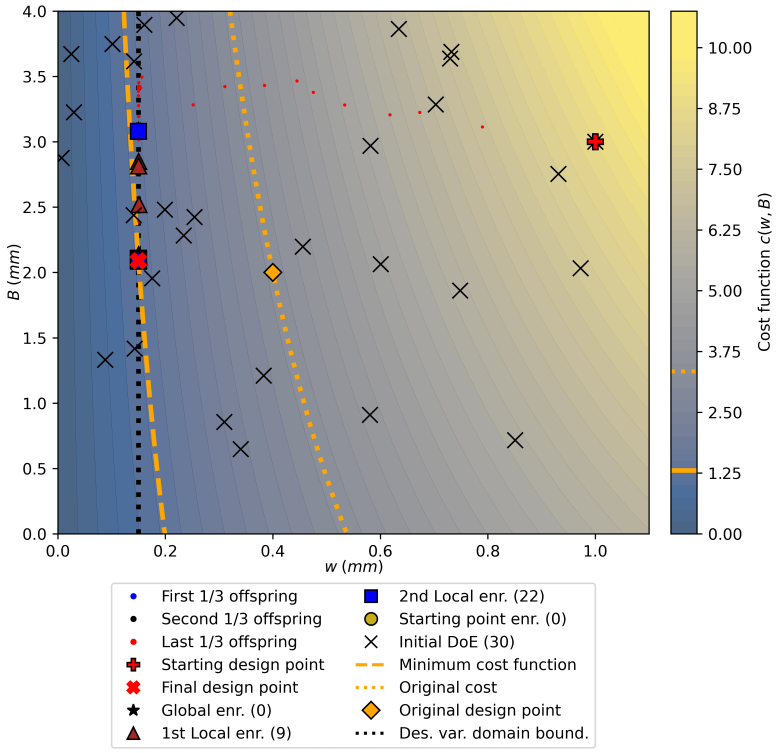

Figure 4.10: RBDO output of Example 3 with iteration history in cost function contours. The obtained $x^* = \{0.150$ m, 2.089 m$\}$ with total number of function calls $N = 61$.

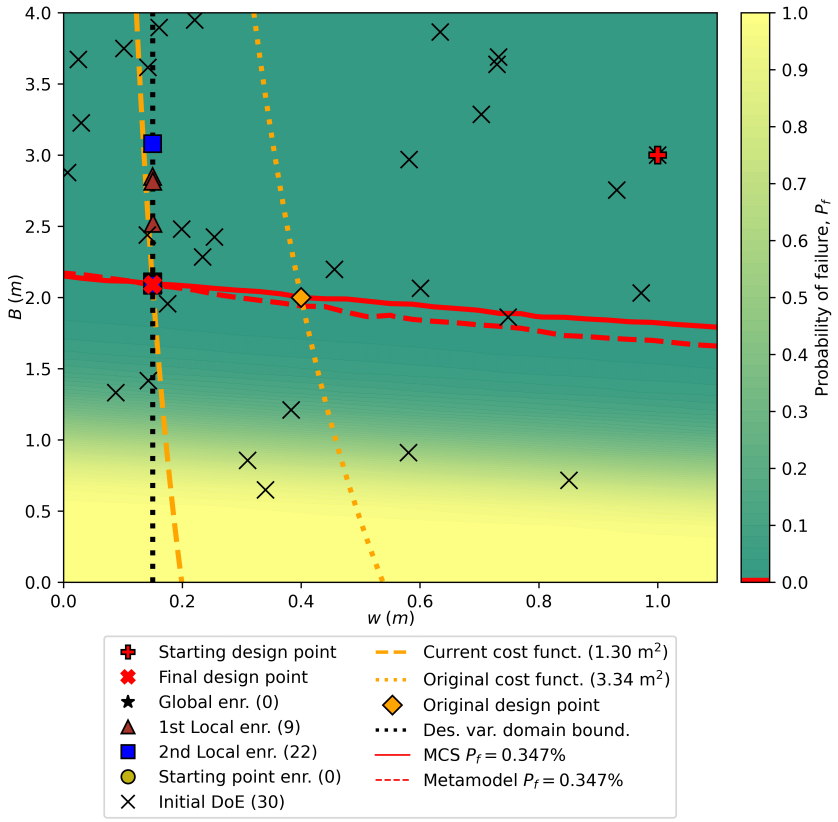

Figure 4.11: RBDO output of Example 3: enrichment points in crude MCS contour.

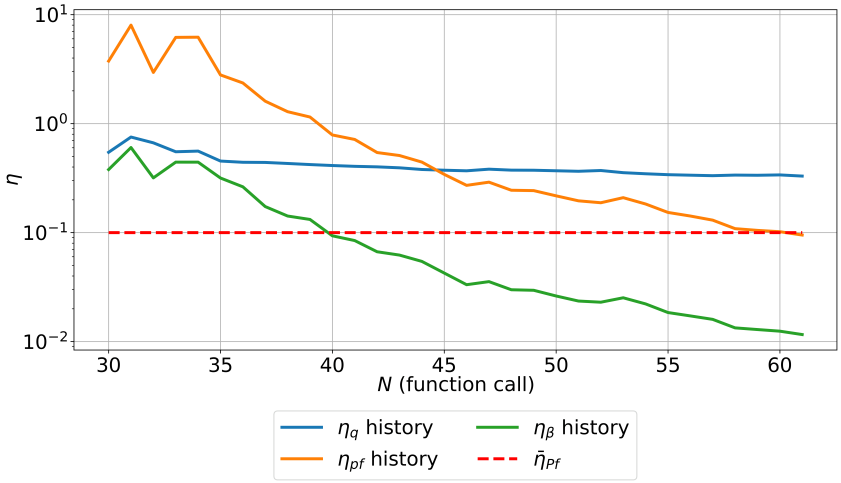

Figure 4.12: RBDO output of Example 3: evolution of $\bar{\eta}_q$, $\bar{\eta}_\beta$ and $\bar{\eta}_{P_f}$ values (y-axis in log. scale). Convergence found at $N = 61$.

4.4 CONCLUSION

Based on the aforementioned examples, some conclusions can be made. Each example obtained the design variable output with high accuracy. This is due to the ability of the method to successfully detect the correct optimal point, which is the point where both the reliability analysis using a metamodel and crude MCS return the targeted probability of failure $\bar{P}_f$. The accuracy is further proven by the very low P_f discrepancy to $\bar{P}_{f,k}$ (or very low "P_f error") when confirmed by crude MCS. The highest P_f error was only 0.40%, which was obtained from Example 1 (Table 4.3). The obtained accuracy is mainly credited to the second local enrichment stage that ensures the optimal point to converge in the correct sub-region and adaptive strategies (discussed in Section 3.5.6) that "force" the method to spend "more time" in exploring the "suspected" optimal sub-region. Moreover, the starting point enrichment helps in ensuring the starting point of each iteration is in a feasible sub-region, while the first local enrichment stage helps the method to "stay" in the correct iteration path. The global enrichment stage is only used in Example 1. Its use in the other examples was skipped because the number of initial enrichment is sufficient enough to fulfill the global enrichment convergence criterion.

Furthermore, efficiency is achieved thanks to the learning functions (discussed in Sections 3.5.3 & 3.5.4) and the adaptive strategies discussed in Section 3.5.6. This efficiency can be best observed in Example 1 (Figure 4.2) where the optimal design point was successfully located by only using "one iteration path" from the initial starting point. Apart from the number of function calls (N), the number of total iterations could also require a considerable amount of computational effort (thus more time is needed) depending on the number of dimensions and N_{mc}. This is especially true after a certain size of DOE is reached, which is one of the disadvantages of using Kriging as a metamodel. For comparison, the five-dimensional Example 1 with $N_{mc} = 10^4$, $N = 23$, and total iteration = 3,450 requires approximately 5 minutes to conclude (using a computer with 16 GB of installed RAM memory and a 2.8 GHz processor). On the other hand, the six-dimensional Example 2 with $N_{mc} = 8 \times 10^4$, $N = 119$, and total iteration = 19,609 requires approximately 12 hours to conclude. Therefore, despite Example 2 only requiring approximately 5 times more function calls than Example 1, it also requires approximately 144 times longer to conclude! This comparison shows that apart from the number of function calls N, it is also important to reduce the required number of total iterations. Therefore, efficiency is not only obtained in terms of the low number of function calls, but also from the low number of total iterations (or "function evaluations") which then correlates to computation time.

To sum up, the proposed RBDO method's effectiveness is demonstrated by solving the various analytical problems accurately and efficiently despite each problem has a different type of performance function.

5 CASE STUDY

This chapter will discuss the application of the proposed RBDO method to the case study of Starnmeer Polder Dyke reinforcement briefly introduced in Chapter 1. The dyke is located in the province of North Holland. It was originally drained in 1643, covers an area of 580 hectares, and is contained within a 13 km long ring dyke. Recently, water board Hoogheemraadschap Hollands Noorderkwartier (HHNK) who manages the dyke initiated a stability assessment of the dyke. It was found that around half the length of the dyke does not comply with the current safety requirement, therefore, a reinforcement of the dyke is needed. The optimal design dimensions of the dyke reinforcement that gives a reliability target β = 3.00 ($\bar{P}_{f,k}$ = 0.135%) will be investigated here by applying the proposed RBDO method.

5.1 MODEL DEFINITION

In this research, the geometry of the model will be slightly modified in order to avoid numerical instability due to the mesh generation of a very thin geometry component. The modified geometry can be seen in Figure 5.1 below (including the soil types in each soil layer/geometry).

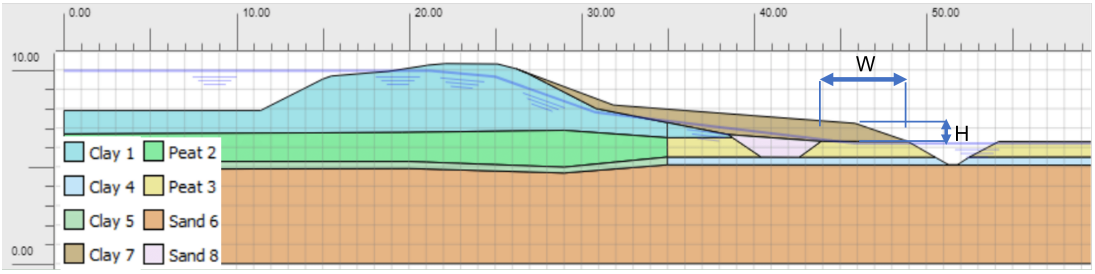

Figure 5.1: The FEM model that will be analyzed by the proposed RBDO method. The number in the legend indicates the number of the soil layers. Dimensions in *m*.

The main difference between Figures 5.1 and 1.1 is that the very thin reinforcement sand layer ("Sand 8") in between "Clay 7" and "Peat 3" layers is removed (where such a thin layer can be seen closely in Figure 1.1). There are 8 different types of soil layers in the original design (Hicks et al. [2019]). However, based on Varkey et al. [2020] who analyzed the same problem to calculate the characteristic values of soil shear strength properties, it was found that soil layers other than layers 1, 2, and 3 have negligible influences on the dyke's FS. Therefore, only the strength of layers 1, 2, 3, and 7 are modeled with stochastic properties in this research. The soil stochastic distributions are presented in Table 5.1, and the deterministic properties are presented in Table 5.2. Moreover, all of the soil will be modeled as the *Mohr-Coulomb* model.

The performance function here would be a FEM model in PLAXIS 2D. The important output here is the FS of the dyke from a combination of W (m), H (m), and soil properties described in Table 5.1. Failure is defined in such a way that FS < 1. Therefore, $\bar{g} = 1$ is chosen as the failure threshold. Moreover, the design variables are $\boldsymbol{x} = \{W,\ H\}^T$, and the environmental variables are $\boldsymbol{z} = \{c'_1,\ c'_2,\ c'_3,\ c'_7,\ \phi'_1,\ \phi'_2,\ \phi'_3,\ \phi'_4\}^T$ (thus $n = 2$ and $z = 8$). The aim here is to find $\boldsymbol{x}^* = \{\boldsymbol{W}^*, \boldsymbol{H}^*\}^T$ that returns a reliability target $\beta = 3$ (or $\bar{P}_{f,k}$ = 0.135%). Therefore, N_{mc} = 80,000 is chosen to satisfy Equation 3.4 ($\delta_{P_f} \leq 0.1$) and $f_{stop} = 10^{-5}$ is chosen. Since the performance function here is a FEM model, a noise term in the kernel will be added to put the noisy FEM responses into account. The rest of the RBDO settings are displayed in Table 5.3.

Table 5.1: Case study: stochastic soil strength parameter distribution (all c′ in kPa). The mean and COV are based on the random variable (not the log of the random variable).

Layer	*Parameter*	*Distribution*	*Mean*(μ)	*COV* (%)
Clay 1	c'_1	Lognormal	4.4	0.773
Peat 2	c'_2	Lognormal	3.2	0.656
Peat 3	c'_3	Lognormal	2.0	0.775
Clay 7	c'_7	Lognormal	6.2	0.773
Clay 1	$\tan \phi'_1$	Lognormal	0.580	0.081
Peat 2	$\tan \phi'_2$	Lognormal	0.398	0.058
Peat 3	$\tan \phi'_3$	Lognormal	0.358	0.145
Clay 7	$\tan \phi'_7$	Lognormal	0.531	0.081

Table 5.2: Case study: deterministic soil parameters. γ and γ' are saturated and unsaturated unit weight. E is modulus of elasticity. ν is Poisson's ratio. c' and $\tan \phi'$ of layers 1, 2, 3, and 7 are presented in Table 5.1

Layer	γ (kN/m^3)	γ' (kN/m^3)	E (kPa)	ν (-)	c' (kPa)	$\tan \phi'$ (-)
Clay 1	13.9	6.9	5000	0.33	-	-
Peat 2	11.0	1.6	2500	0.33	-	-
Peat 3	11.0	1.4	2500	0.33	-	-
Clay 4	15.0	15.0	5000	0.33	4.5	0.559
Clay 5	15.0	15.0	5000	0.33	5.4	0.601
Sand 6	20.0	18.0	45000	0.30	0.0	0.637
Clay 7	17.0	17.0	5000	0.33	-	-
Sand 8	20.0	18.0	45000	0.30	0.0	0.637

Table 5.3: RBDO input summary of the case study.

Parameter	*Input*
n	2
m	3
z	8
Metamodel kernel	Matérn 3/2 with a white noise component
$(\boldsymbol{x_0})$	$\{5.04, 0.94\}^T$
$(\boldsymbol{z_0})$	The mean values displayed in Table 5.1
Metamodel N_{mc}	80,000 (Expected δ_{P_f} = 0.096)
$\bar{P}_{f,k}$	0.135% (β = 3.00)
$\bar{\eta}_{glo}$ & $\bar{\eta}_{P_f}$	0.20 & 0.10
$\bar{\eta}_q$	$\{1.00, 0.75, 0.50, 0.10\}$ for every 300 iteration
$\bar{\eta}_q$	$\{1.20, 0.75, 0.50, 0.10\}$ (after "new minimum" Kriging variance)
f_{stop}	10^{-5}
σ_0	0.5
$\bar{g}$	1.0
i_{max}	4000
N_{max}	350

The cost function and constraints of the problem are formulated in Equations 5.1 and 5.2. There are a few possibilities for defining the cost function of the problem. One could define it as the area (or volume) of the whole Clay 7, or an area of a representative part. In this case, the latter is chosen whereas the cost function is the triangular area of the slope at the toe of the dyke (the red triangle in Figure 5.2). Moreover, the chosen cost function is more straightforward and utilizes the original design variables W & H that are also used in Hicks et al. [2019] and represent the cross-section area of the whole Clay 7 layer. If the cost function were defined as the whole cross-section area of the Clay 7 layer, one must define design parameters (could be more than 2) that will be used in the formulation of the cross-section area. Furthermore, lower values of W & H lead to a lower red triangle area (and Clay 7 volume in general), thus less clay is needed for the reinforcement design. A lower W value also leads to a narrower reinforcement design, which may be advantageous if the dyke is located in a limited space (e.g. near a housing border). Therefore, an optimal design would be a combination of W & H

that returns the lowest possible cross-section area of the red triangle, yet still fulfilling $P_f = \hat{P}_f$. The lower limit of H is set to 0.05 m to avoid numerical instability in the model geometry (which is kept to be as close as possible to the original reinforcement design).

$$c(W,\ H) = \frac{(W - 1.97)}{2} \times H \tag{5.1}$$

subjects to:

$$\mathbf{x}^* = arg\ \min_{\mathbf{x}\in\mathcal{D}} c(\mathbf{x}) \begin{cases} 1.97\ m < W < 5.10\ m \\ 0.05\ m \leq H \leq 1.00\ m \\ \mathbb{P}(g_k(\mathbf{x}, \mathbf{z}) \leq 1) \leq 0.135\% \end{cases} \tag{5.2}$$

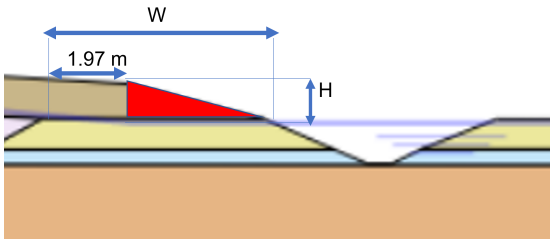

Figure 5.2: The cost function to be minimized: the area of the red triangle.

The initial preferred starting point $\boldsymbol{x}_0 = \{5.04\text{ m}, 0.94\text{ m}\}^T$ is chosen as this is the original design dimensions in Hicks et al. [2019]. The soil properties are the mean value taken from Table 5.1 (only here the soil friction angle is defined as ϕ' instead of $tan\ \phi'$). The initial DOE for metamodel initialization will be spread in such a way that most of the points are slightly less than $\boldsymbol{x}_0 =$ since these sub-regions are the most likely to be passed by the iteration path of the optimization process because they return lower cost function values. Therefore, there is no point in spreading the initial DOE at the sub-regions where $\boldsymbol{x}$ is bigger than $\{5.04\text{ m}, 0.94\text{ m}\}$ since it will not be explored by the optimization iteration path. Moreover, the sub-regions where the initial DOE was sampled will be used in determining the value of η_{glo}, Therefore, the global enrichment stage could be minimized (more efficient).

5.2 RESULT

The results of the RBDO process are presented in Table 5.4 and Figures 5.3 to 5.6. The obtained optimal values for the design parameter are $\boldsymbol{x}^* = \{2.586\text{ m},\ 0.050\text{ m}\}^T$.

Table 5.4: RBDO output summary of the case study.

Parameter	*Output*
$\boldsymbol{x}^*$ (m)	$\{2.586, 0.050\}^T$
Total N	350
Metamodel $\hat{P}_f$ $(N_{mc} = 8 \times 10^5)$	0.1350% (β =3.003)
AK-MCS P_f $(N_{mc} = 5 \times 10^5)$	0.1336%
AK-MCS P_f error to $\bar{P}_{f,k}$	1.048%
Total iteration	44,566
Total function evaluation (metamodel)	3.56×10^9

The iteration points shown in Figure 5.3 are limited to the last 2000 data (to save computation memory and make the result easier to read). Therefore, it can be seen that the last 2000 iteration brought the design parameters closer to the converged area faster (because the metamodel was already good enough to identify possible optimal sub-regions after a certain number of DOE). The last 2000 data roughly depicts the iteration paths of the last 4-5 enrichment processes (where in this case, after each addition of 1 DOE, 400-500 iterations are usually needed).

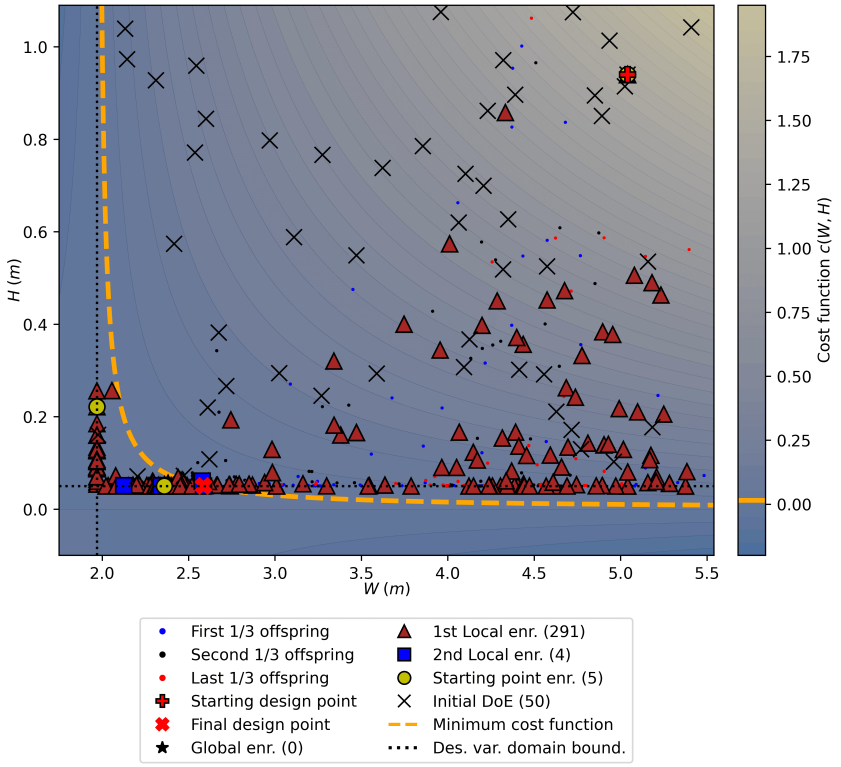

Figure 5.3: Case study's RBDO output: iteration history in cost function contour.

The number of N_{max} is reached here, therefore the RBDO process is concluded by the steps defined in Section 3.5.7 with 44,566 total iterations (which means 44,566 $\times$ 80,000 $\approx 3.56 \times 10^9$ function calls are needed had a metamodel not been used!). The optimal point is finally chosen from the shortlisted design variable candidates (from the best points in the second local enrichment population) and the last optimization process where $\bar{\eta}_q = \infty$. The RBDO process could be continued by increasing the allowed number of N_{max} to get a more accurate output, however, each iteration would take significantly longer time due to the increase in the DOE (note that the more DOE the metamodel has, the bigger the size of matrix $\boldsymbol{K}$ in Equation 3.8, therefore the increase is exponential).

Figures 5.3 and 5.4 show that there are four second local enrichment points. It means that the RBDO converged to those points, but the RBDO did not conclude because some of them were "false positive". Note that the number of the second local enrichment points shown in both figures could be reduced if a lower value of f_{stop} was chosen (e.g. 10^{-8} or lower). The number of second local enrichments is relatively low compared to the first local enrichment stage, this is due to the convergence criteria defined in Section 3.5.5 (especially Equation 3.57) which further limits the "validity" of the converged point. Moreover, Figure 5.3 or 5.4 shows that the first local enrichments were made along the iteration path from the initial starting point to the optimal sub-region.

Figure 5.4 shows the $\hat{P}_f$ contour based on the optimized metamodel (from the RBDO). The optimal design point obtained from the RBDO lies in the intersection of the $\hat{P}_f$ and minimum cost function contour line, where many first and second local enrichment points are also located. This is because the preferred starting points were updated in close proximity to this sub-region at a certain time during the RBDO process (shown by a few starting point enrichment points that were visible by yellow circles that are closed to the optimal sub-region).

Figure 5.3 (or 5.4) also shows that almost all of the first and second local enrichments were made along the lower boundary limit of H. It means that at the earlier phase of the RBDO process, the metamodel $\hat{P}_f$ contour line lies further than the feasible design domain. This can be explained in Figure 5.5 which shows the location of the $\hat{P}_f = \bar{P}_{f,k}$ contour line when N =200 (roughly at the middle of the RBDO process). Compared to Figure 5.4, the $\hat{P}_f = \bar{P}_{f,k}$ contour line lies closer to the feasible design domain boundary. It means that the $\hat{P}_f = \bar{P}_{f,k}$ contour line was moved as the metamodel got "smarter" due to the enrichments performed.

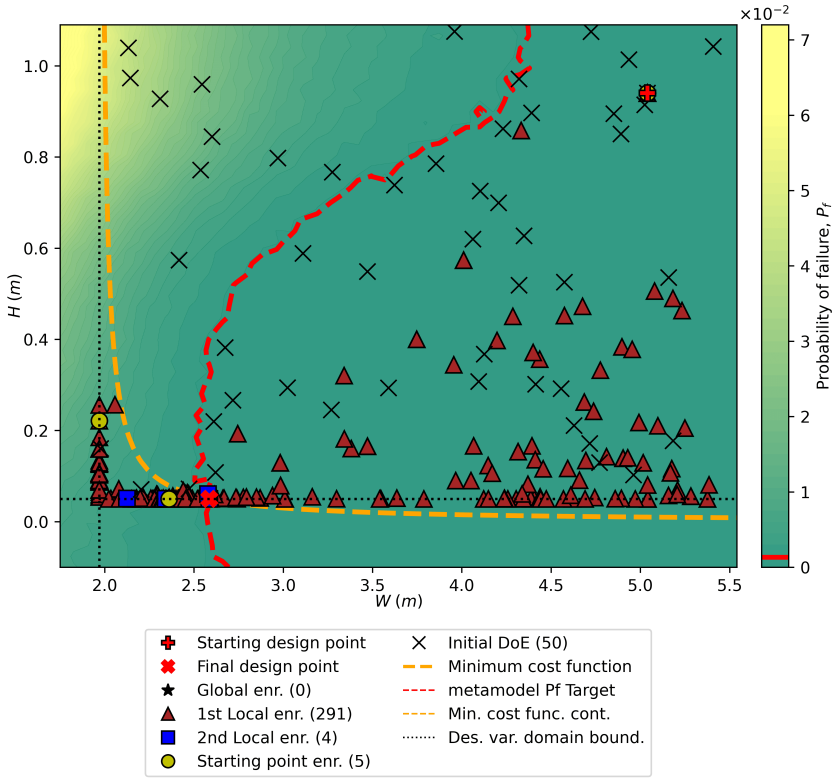

Figure 5.4: Case study's RBDO output: enrichment points in the metamodel output's contour.

Note that the chosen $\bar{\eta}_{glo}$ and $\bar{\eta}_q$ values are slightly more relaxed compared to the chosen value in the examples presented in Sections 4.1 & 4.2. The relaxed criteria are chosen to reduce the required number of function calls during the global and first local enrichment stage. Due to the noise component, the Kriging variance will increase when there are some outliers in the model responses. Therefore, to avoid further increase in the Kriging variance, after a certain N, training data with FS < 0.7 will be ignored (not added to the DOE) and the iteration will be restarted from the latest preferred starting point. When N was still low (approximately when $N < 150$), outliers (FS < 0.7) were still allowed to detect "unsafe" sub-regions. An overly strict limit in identifying outliers may prevent the metamodel to detect unsafe sub-regions, therefore leading to a higher chance of *not* having the $\hat{P}_f = \bar{P}_{f,k}$ contour line even when the number of DOE already reached N_{max} (or still having $\hat{P}_f = \bar{P}_{f,k}$ similar to what is shown in Figure 5.5 even when $N = N_{max}$). Moreover, a strict $\bar{\eta}_q$ during the early stage of iteration could also create "endless loops" of the local enrichment stage. Endless loops of local enrichment happens when the following situation occurs:

1. At the early stage of iteration, it is found that $\eta_q > \bar{\eta}_q$.
2. One first local enrichment stage is made by adding point (x_e, z_e) as a new DOE (where it has the lowest U value).
3. When the iteration continues (from the last enrichment point), the η_q value is still lower than $\bar{\eta}_q$ (because the $\bar{\eta}_q$ value is strict enough to allow so).
4. One first local enrichment will be performed again (despite the iteration having just started at $i = 0$).
5. The same point as before will be selected as the new DOE.
6. There are now two exact points in the DOE.
7. When the iteration continues, step 3 is repeated.

The endless loop specified above could be solved by increasing the $\bar{\eta}_q$ threshold during the early stage of the optimization iteration. To further reduce unnecessary enrichment at the starting point,

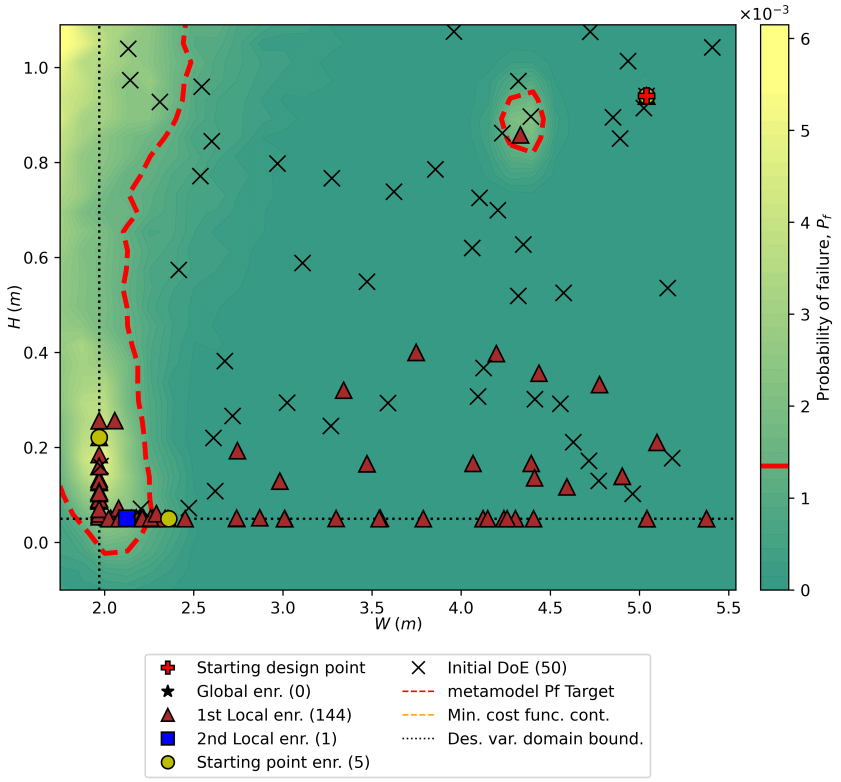

Figure 5.5: The location of the $\hat{P}_f = \bar{P}_{f,k}$ contour line at the middle of the RBDO process (when N =200).

$\bar{\eta}_q = 1.2$ will be assigned when $i \leq 20$ and $x_0 = \{5.04, 0.94\}$. The $\bar{\eta}_q = 1.2$ will also be assigned when the Kriging variance reached a "new minimum" due to the increasing number of DOE. This "new minimum" stage can be detected when the addition of more DOE at the starting point $x_0 = \{5.04, 0.94\}$ does not reduce the corresponding η_q, therefore continuing enriching at the same point would just "waste" function calls. The chosen initial $\bar{\eta}_q$ values will also "force" the metamodel to conclude to the optimal sub-region candidates faster at the expense of a less accurate metamodel along the iteration path. Sacrificing an accurate metamodel during the iteration path will be compensated by the second local enrichment stage, where it will play a big role in ensuring the convergence falls in the correct sub-region later on. Forcing the metamodel to conclude faster will also help to conclude the RBDO faster since fewer global and first local enrichments are expected, this strategy is particularly useful for $N_{mc} > 10{,}000$ and when $N > 100$ (these values are relative, depending on the performance of the computer used).

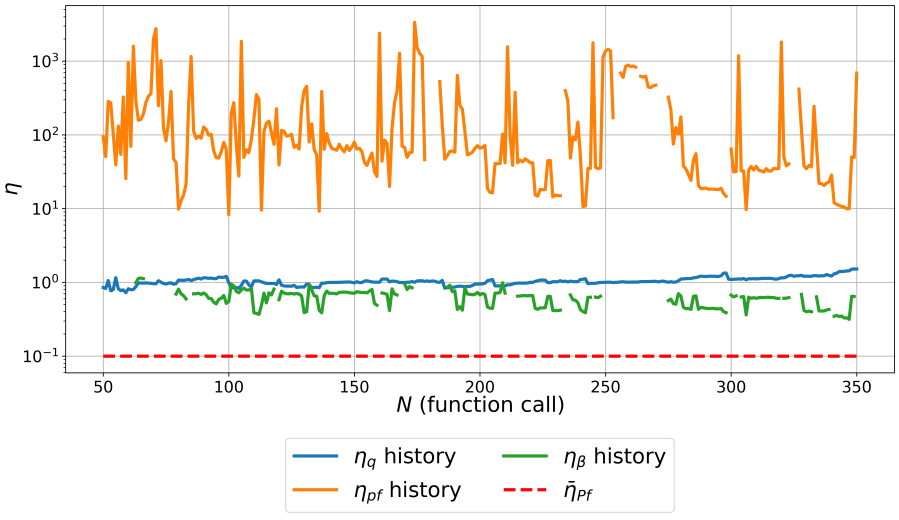

Figure 5.6: Case study RBDO output: convergence history (y-axis in log. scale).

The convergence history (η) is shown in Figure 5.6. It was found that the η_{P_f} was still far from $\bar{\eta}_{P_f} = 0.1$ despite the general trend of η_{P_f} slightly decreasing (almost negligible). This is due to the noise variance component in the metamodel's kernel that allows for noise or "tolerance" on the prediction trend at the expense of high Kriging variance ("new minimum"), as shown in Figure 3.2. Figure 5.6 also shows the η_q trend to slightly increased throughout the RBDO process. This is due to the new threshold of $\bar{\eta}_q = 1.20$ at the early stages of iteration after the Kriging "new minimum" variance was detected. The final design of the dyke reinforcement, with $\boldsymbol{x}^* = \{\boldsymbol{W}^*, \boldsymbol{H}^*\} = \{2.586 \text{ m}, 0.050 \text{ m}\}^T$, is displayed in Figure 5.7. Furthermore, the typical mode of failure of the dyke is shown in Figure 5.8.

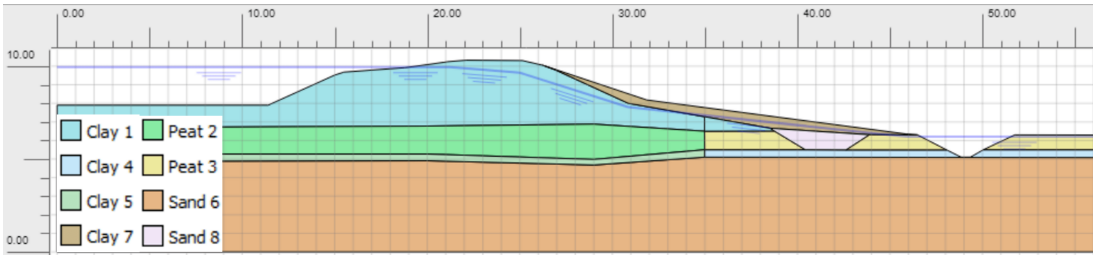

Figure 5.7: The final design of the dyke with $\{W, H\} = \{2.586 \text{ m}, 0.050 \text{ m}\}$. Dimensions in m.

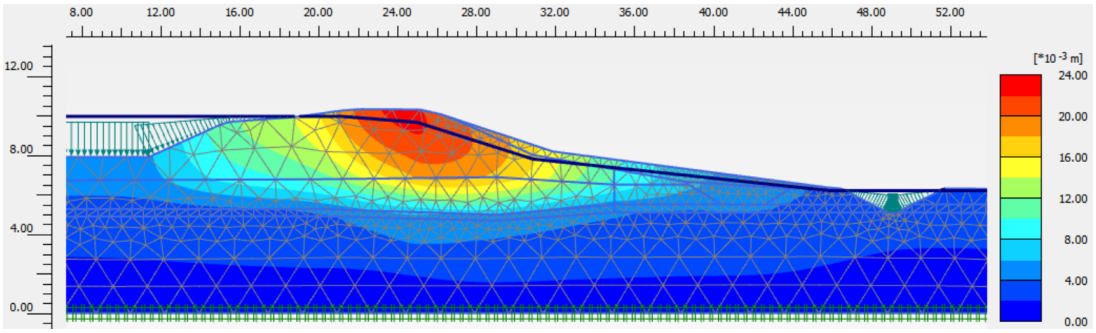

Figure 5.8: The mode of failure of the dyke with $\{W, H\} = \{2.586 \text{ m}, 0.050 \text{ m}\}$. Dimensions in m.

Finally, due to the limited resource and time in confirming the obtained result by using crude MCS, AK-MCS was used to confirm the obtained $\hat{P}_f$. For confirmation, an AK-MCS analysis (limited to 100 function calls) with $\boldsymbol{x}^* = \{2.586 \text{ m}, 0.050 \text{ m}\}^T$ returns $P_f = 0.1336\%$. Therefore, there is a 1.048% in error from the RBDO output. The optimal design $\boldsymbol{x}^*$ also reduces the use of clay in the clay 7 layer up to approximately 30%!

5.3 CONCLUSION

The proposed RBDO method can predict the optimal design variables of a complex and expensive FEM geotechnical engineering model within $N_{max} = 350$ function calls. The optimal design obtained from the method, $\{W, H\} = \{2.586 \text{ m}, 0.050 \text{ m}\}$, reduces the need for clay in the reinforcement design. Also, it decreases the required reinforcement width from $W = 5.040$ m to only $W = 2.586$ m, which is very beneficial when space availability is limited.

Moreover, due to the high Kriging variance caused by using a noise component in the metamodel's kernel, the convergence criteria $\bar{\eta}_{P_f}$ is not achieved. Therefore, N_{max} is reached and $\boldsymbol{x}^*$ is obtained through the steps defined in Section 3.5.7. The obtained result was confirmed through AK-MCS with a 1.048% error in the P_f compared to the targeted $\bar{P}_{f,k}$.

6 CONCLUSION

6.1 CONCLUSION

The main objective of this research is to accurately and efficiently optimize a FEM-based dyke reinforcement design that fulfills the reliability target by using an RBDO method consisting of an active-learning metamodel combined with an optimization scheme. Therefore, the discussions and findings presented in Chapters 3 to 5 will be concluded to answer the research questions formulated in Chapter 1.

How to efficiently and accurately perform a design optimization of a dyke reinforcement problem with an expensive FEM-based performance function while also fulfilling the reliability target?

To answer the main research question, a novel RBDO method is proposed in this research. The method is developed by coupling active-learning Kriging metamodeling reliability analysis (AK-MCS) with (1+1)-CMA-ES optimization process, and further combining them with certain adaptive strategies to obtain the optimal design point accurately and efficiently. The proposed method returns an optimal set of design variable $\boldsymbol{x}^*$ within a certain number of function calls N_{max} (limited to 350 in this research) instead of performing function calls in the order of 10^9.

How to combine a metamodel and an optimization process into an RBDO method?

As defined in Chapter 3, the proposed RBDO method builds the Kriging metamodel through four enrichment stages to obtain the optimal design point. These stages are the global, first local, second local, and starting point enrichments. The global enrichment aims to reduce the uncertainty in the whole domain of the augmented space to give a "decent enough" metamodel to start the RBDO process. The first local enrichment helps the optimization process to detect the correct iteration path when searching for the optimal design point. The second local enrichment is responsible for ensuring the optimization process converges to the correct optimal design point. Moreover, the starting point enrichment ensures the optimization process starts from a feasible starting point, which is crucial for the (1+1)-CMA-ES optimization process.

How to achieve the accuracy of the RBDO method?

The accuracy of the RBDO method is mainly credited to the second local enrichment stage where it confirms the accuracy of a converged design point and avoids the incorrect optimal point (or "false positive"). This is further supported by the adaptive strategies defined in Sections 3.5.4, 3.5.5, and 3.5.6 that "force" the optimization iterations to "spend more time" around a suspected optimal sub-region, thus leading to a higher chance of locating the optimal design point. These strategies also lower the probability of getting an "outlier" in the DOE which could decrease the metamodel's accuracy. Moreover, when N_{max} is reached, strategies described in Section 3.5.7 ensure that the optimal point is chosen based on the most updated metamodel, thus the most probable optimal design point can be chosen.

The role of the second local enrichment and the aforementioned strategies are successfully demonstrated in the discussed analytical and FEM problems. In Example 1, the obtained optimal design point only gives a 0.003% cost function value error compared to the analytical solution, which is more accurate compared to other existing well-known RBDO methods (Table 4.4). Moreover, Example 2 shows that the proposed RBDO method obtained an optimal design point that returns a reliability index β closest to the target compared to other existing RBDO methods (Table 4.8). Example 3 shows that the method could predict a better alternative design point that only has a 0.028% error to the targeted P_f.

How to achieve the efficiency of the RBDO method?

The efficiency of the RBDO is achieved by performing metamodel enrichments at the most strategic sub-regions (where the optimal design point is "suspected" to lie in one of these sub-regions), therefore avoiding unnecessary enrichments at less important sub-regions. Apart from the learning functions defined in Sections 3.5.3 and 3.5.4 that choose the best enrichment point candidate from a Monte Carlo population, efficiency is further enhanced by strategies described in Sections 3.5.4, 3.5.5, and 3.5.6. These strategies avoid wasting enrichments (both the first local enrichment or the starting point enrichment) at "less important" design variables, and also avoid unnecessary iterations at sub-regions that are deemed "obviously not important". When dealing with a noisy FEM model, these strategies also adapt the first local enrichment's convergence criterion when the metamodel's Kriging variance reaches a "new minimum" due to the noisy response. This further avoids unnecessary enrichment at the starting point (both the original and the updated one). The efficiency can be best observed in Example 1, where the optimal design point can be located by only using one "iteration path" and 23 function calls. Moreover, Examples 2 and 3 obtain the optimal design point within 119 and 61 function calls respectively, which are less than the corresponding N_{max} values.

What is the optimal design for the dyke reinforcement based on the proposed RBDO method?

The proposed RBDO method obtains an optimal design point of $\boldsymbol{x}^* = \{2.586 \text{ m}, 0.05 \text{ m}\}$ for the dyke reinforcement. The optimal design is obtained after $N_{max} = 350$ function calls are reached. After a confirmation with AK-MCS reliability analysis (limited to 100 function calls), the error in the P_f is only 1.048%.

Furthermore, the accuracy and efficiency are also achieved to a certain extent depending on the balance of the RBDO setting, e.g.: convergence criteria, locations of the initial DOE, size of the MCS population, and the metamodel setting. Despite the ability in predicting an optimal design, the proposed method becomes inefficient after a certain number of function calls due to the increase in the correlation matrix size of the training data. Therefore, the robustness of the method is highly dependent on the ability of the user's computer.

6.2 RECOMMENDATION

Despite the promises that the proposed RBDO method has, some things could be considered to further improve the method. The effectiveness of the method for problems with more than ten dimensions has not been verified yet. To achieve further efficiency during the optimization iterations, the use of a more efficient reliability analysis method (e.g. subset simulation, importance sampling, directional sampling, etc.) could have been investigated. Furthermore, the correlation between the convergence criteria and the type of performance function could have been further investigated to achieve an optimal or "acceptable" result without having excessive function calls.

BIBLIOGRAPHY

Agarwal, H. and Renaud, J. (2004). Reliability-based design optimization using response surfaces in application to multidisciplinary systems. *Eng Opt*, 36(3):291 – 311.

Aoues, Y. and Chateauneuf, A. (2010). Benchmark study of numerical methods for reliability-based design optimization. *Struct Multidisc Optim*, 41:277–294.

Arnold, D. and Hansen, N. (2012). A (1+1)-cma-es for constrained optimization. *Genetic and evolutionary computation conference*, pages 297 – 304.

Au, S. (2005). Reliability-based design sensitivity by efficient simulation. *Comput Struct*, 83(14):1048 – 1061.

Bichon, B. J., Eldred, M. S., Swiler, L. P., Mahadevan, S., and McFarland, J. M. (2008). Efficient global reliability analysis for nonlinear implicit performance functions. *AIAA*, 46(10):2459 – 2468.

Blatman, G. and Sudret, B. (2009). An adaptive algorithm to build up sparse polynomial chaos expansions for stochastic finite element analysis. *Probabilistic Engineering Mechanics*, 25:183–197.

Bourinet, J.-M. (2018). Reliability analysis and optimal design under uncertainty - focus on adaptive surrogate-based approaches. *HAL open science, Université Clermont Auvergne*, page 124.

Brinkgreve, R. B. J., Kumarswamy, S., Swolfs, W. M., and Fonseca, F. (2020). *PLAXIS 2020 (User Manual)*. Bentley, Delft.

Chen, X., Hasselman, K., and Neil, D. (1997). Reliability-based structural design optimization for practical applications. *38th Structures, Structural Dynamics, and Materials Conference*, pages 2724 – 2732.

Chen, Z., Peng, S., Li, X., Qiu, H., Xiong, H., Gao, L., and Li, P. (2015). An important boundary sampling method for reliability-based design optimization using kriging model. *Struct Multidisc Optim*, 52(1):55 – 70.

Claudio, M. R. and Alí, M. J. (2002). Fast monte carlo reliability evaluation using support vector machine. *Reliab Eng Sys Saf*, 76(3):237 – 243.

Du, X. and Chen, W. (2004). Sequential optimization and reliability assessment method for efficient probabilistic design. *J Mech Des (ASME)*, 126(2):225 – 233.

Dubourg, V., Sudret, B., and Bourinet, J.-M. (2011). Reliability-based design optimization using kriging surrogates and subset simulation. *Struc Multidisc Optim*, 44:673–690.

Echard, B., Gayton, N., and Lemaire, M. (2011). Ak-mcs: An active learning reliability method combining kriging and monte carlo simulation. *Structural Safety*, 33:145–154.

Enevoldsen, I. and Sørensen, J. (1994). Reliability-based optimization in structural engineering. *Struct Saf*, 15(3):169–196.

Forrester, A., Sobester, A., and Keane, A. (2008). *Engineering Design Via Surrogate Modelling: A Practical Guide*. John Wiley & Sons, Chichester.

Foschi, R., Li, H., and Zhang, J. (2002). Reliability and performance-based design: a computational approach and applications. *Struct Saf*, 24(2-4):205 – 218.

Hicks, M. A., Varkey, D., van den Eijnden, A. P., de Gast, T., and Vardon, P. J. (2019). On characteristic values and the reliability-based assessment of dykes. *Georisk: Assessment and Management of Risk for Engineered Systems and Geohazards*, 13(4):313 – 319.

Honjo, Y., Suzuki, M., Hara, T., and Zhang, F. (2009). *Geotechnical Risk and Safety*. CRC Press/Balkema, Leiden, The Netherlands.

Jia, G. and Taflanidis, A. (2013). Non-parametric stochastic subset optimization for optimal-reliability design problems. *Comput Struct*, 126:86 – 99.

Jones, D., Schonlau, M., and Welch, W. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492.

Kentrop, D. (2021). A metamodelling approach to reliability updating with dike construction survival. *MSc Thesis, Delft University of Technology*.

Kuschel, N. and Rackwitz, R. (1997). Two basic problems in reliability-based structural optimization. *Math Method Oper Res*, 46(3):309–333.

Le, T. C. K. and Honjo, Y. (2008). Geotechnical safety network: Example 5. *https://www.geoengineer.org/geosnet/*.

Lee, P. (1997). *Bayesian statistics: an introduction, 2nd edn.* Wiley publishing, London.

Lee, T. and Jung, J. (2008). A sampling technique enhancing accuracy and efficiency of metamodel-based rbdo: Constraint boundary sampling. *Comput Struct*, 86(13-14):1463–1476.

Lehký, D., Slowik, O., and Novák, D. (2018). Reliability-based design: Artificial neural networks and double-loop reliability based optimization approaches. *Advances in Engineering Software*, 117:123–135.

Li, X., Qiu, H., Chen, Z., Gao, L., and Shao, X. (2016). A local kriging approximation method using mpp for reliability-based design optimization. *Comput Struct*, 162:102 – 115.

Liang, J., Mourelatos, Z., and Nikolaidis, E. (2007). A single-loop approach for system reliability-based design optimization. *J Mech Des*, 129(12):1215 – 1224.

Moustapha, M. and Sudret, B. (2019). Surrogate-assisted reliability-based design optimization: a survey and a unified modular framework. *Structural and Multidisciplinary Optimization*, 60:2157 – 2176.

Moustapha, M., Sudret, B., Bourinet, J.-M., and Guillaume, B. (2016). Quantile-based optimization under uncertainties using adaptive kriging surrogate models. *Struct Multidisc Optim*, 54(6):1403 – 1421.

Papadrakakis, M., Lagaros, N., and Plevris, V. (2005). Design optimization of steel structures considering uncertainties. *Eng Struct*, 27(9):1408 – 1418.

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

Ranjan, P., Bingham, D., and Michailidis, G. (2008). Sequential experiment design for contour estimation from complex computer codes. *Technometrics*, 50(4):527–541.

Rasmussen, C. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA.

Rayyan, M. (2021). Reliability benchmarking of eurocode 7 design examples: Cie5050-09: Additional thesis project 2021. *Delft University of Technology: Additional MSc Thesis*.

Royset, J., Kiureghian, A. D., and Polak, E. (2001). Reliability-based optimal structural design by the decoupling approach. *Reliab Eng Syst Saf*, 73(3):213 – 221.

Santner, T. J., Williams, W. B. J., Notz, W., and Williamns, B. J. (2003). *The Design and Analysis of Computer Experiments*. Springer, Vol 1. New York.

Sudret, B. and Kiureghian, A. D. (2002). Comparison of finite element reliability methods. *Probabilistic Engineering Mechanics*, 17:337–348.

Sun, Z., Wang, J., Li, R., and Tong, C. (2017). Lif: A new kriging based learning function and its application to structural reliability analysis. *Reliability Engineering and System Safety*, 157:152 – 165.

Taflanidis, A. and Beck, J. (2008). Stochastic subset optimization for optimal reliability problems. *Prob Eng Mech*, 23:324 – 338.

Teixeira, R., Nogal, M., and O'Connor, A. (2021). Adaptive approaches in metamodel-based reliability analysis: A review. *Structural Safety*, 89:1–15.

Tu, J. and Choi, K. (1999). A new study on reliability-based design optimization. *J Mech Des (ASME)*, 121(4):557–564.

van den Eijnden, A. P., Schweckendiek, T., and Hicks, M. A. (2021). Metamodelling for geotechnical reliability analysis with noisy and incomplete models. *Georisk: Assessment and Management of Risk for Engineered Systems and Geohazards*, pages 1–18.

van der Werf, J. (2021). Reliability analysis of quay walls using metamodelling. *MSc Thesis, Delft University of Technology*.

Varkey, D., Hicks, M. A., van den Eijnden, A. P., and Vardon, P. J. (2020). On characteristic values for calculating factors of safety for dyke stability. *Geotechnique Letters*, 10:353–359.

Zhang, X., Lu, Z., and Cheng, K. (2021). Reliability index function approximation based on adaptive double-loop kriging for reliability-based design optimization. *Reliability Engineering and System Safety*, 216:1–13.

Zhaoyan, L., Zhenzhou, L., and Wang, P. (2015). A new learning function for kriging and its applications to solve reliability problems in engineering. *Comput Math Appl*, 70(5):1182 – 1197.

A (1+1)-CMA-ES ALGORITHM

The following is the algorithm of the (1+1)-CMA-ES procedure (Algorithm 1) implemented in the research. This algorithm is a more detailed description of the optimization process depicted in Figure 3.3. Note that since the aim is to minimize the cost function in this research, therefore $f(\boldsymbol{y})$ is deemed as inferior to $f(\boldsymbol{d}^{(i-5)})$ if $f(\boldsymbol{y}) > f(\boldsymbol{d}^{(i-5)})$ (Line 43 of Algorithm 1).

Algorithm 1 : (1+1)-CMA-ES optimization procedure

1: **procedure** (1+1)-CMA-ES (n, m, z, i, $\boldsymbol{x}$, σ, $\boldsymbol{A}$, $\boldsymbol{v}$, $\boldsymbol{P}_{succ}$, $\boldsymbol{s}$, $f(\boldsymbol{d}^{i-5})$)
2: Check the initial or existing iteration parameters.
3: **if** $i = 0$ **then** ▷ If it is the start of iteration, use initial parameters.
4: $\boldsymbol{x} = \boldsymbol{x}_0$ ▷ Parental var., shape = (n, 1)
5: $\sigma = \sigma_0$ ▷ e.g.: 0.5
6: $\boldsymbol{A} = I$ ▷ (n, n)
7: $\boldsymbol{v} = \boldsymbol{0}$ ▷ (n, m)
8: $\boldsymbol{P}_{succ} = 0$
9: $\boldsymbol{s} = \boldsymbol{0}$ ▷ (n, 1)
10: $f(\boldsymbol{d}^{(i-5)}) = 0$ ▷ Last fifth ancestor.
11: g_j : Constraint funct. ▷ g_j is here defined as unfeasible when $g_j > 0$.
12: **end if**
13:
14: Check $g_j(\boldsymbol{x})$ for j= 1 to m.
15: **if** $g_j(\boldsymbol{x}) > 0$ **then** ▷ If the starting point is not feasible.
16: Perform starting point enrichment.
17: $\boldsymbol{y} = \boldsymbol{x}_0$
18: **else**
19: Calculate d, c, c_p, P_{target}, c_{cov}^{+}, c_c, β, Eq. 3.25 to 3.32.
20: Generate random standard normal values $\boldsymbol{z}$. ▷ (n, 1)
21: Calculate c_{cov}^{-} by using Equation 3.30.
22: **if** $c_{cov}^{-} < 0$ **then**
23: Re-generate $\boldsymbol{z}$.
24: **else**
25: Generate one offspring $\boldsymbol{y}$, Eq. 3.14.
26: **end if**
27: Calculate the cost function difference f_Δ, Eq. 3.23.
28: Calculate $\boldsymbol{g}_j(\boldsymbol{y})$ for $j = 1$ to m. ▷ (m, 1)
29: $\boldsymbol{w} = \boldsymbol{0}$ ▷ (n, m)
30: **for** $j = 1$ to m **do**
31: **if** $\boldsymbol{g}_j > 0$ **then**
32: Update $\boldsymbol{v}_j$ that gives $g_j(\boldsymbol{y}) > 0$, Eq. 3.15. ▷ (n, 1)
33: Compute w_j, Eq. 3.17. ▷ (n, 1)
34: **end if**
35: **end for**
36: $\boldsymbol{v} \leftarrow \boldsymbol{v}$. ($\boldsymbol{v}$ is updated). ▷ (n, m)
37: $\boldsymbol{w} \leftarrow \boldsymbol{w}$. ▷ (n, m)
38: **if** Any $\boldsymbol{g}_j(\boldsymbol{y}) > 0$ **then**
39: $\boldsymbol{y}$ is unfeasible.
40: Update transformation matrix $\boldsymbol{A}$, Eq. 3.16. ▷ (n, n)
41: The iteration is stopped here, $\boldsymbol{x}$ will be re-used as the parent.

```
        else
            y is feasible.
            Evaluate f(y).
            Update P_succ, Eq. 3.18.
            Update σ, Eq. 3.19.
            if f(y) ≤ f(x) then                ▷ This research aims to minim. the cost funct.
                x ← y.
                Update s, Eq. 3.20.            ▷ (n, 1)
                Update A, Eq. 3.21             ▷ (n, n)
                The iteration is stopped.
            else
                y ← x.                         ▷ x will be re-used in next iteration.
                if f(y) is inferior to f(d^(i−5)) then
                    Update A, Eq. 3.22.        ▷ (n, n)
                end if
                Iteration is stopped.
            end if
        end if
    end if
    return y                                   ▷ y will be used as x in the next iteration.
end procedure
```

B RBDO METHOD ALGORITHM

The following is the algorithm of the proposed RBDO procedure (Algorithm 2). This algorithm is a more detailed description of the RBDO process displayed in Figure 3.5. For further questions, please send an email to rayyan8818@gmail.com or muhammad.rayyan@ptpp.co.id.

Algorithm 2 : An RBDO Procedure with Four Stages of Enrichment

procedure RBDO (n, z, f, $\bar{P}_{f,k}$, $\boldsymbol{x}_0$, $\boldsymbol{z}_0$)
  Initialization:
  n = No. design variable.
  z = No. environmental variable.
  $ndim = n + z$.
  f = Objective function (cost function).
  $\bar{P}_{f,k} = P_f$ target.
  $(\boldsymbol{x}_0, \boldsymbol{z}_0)$ = The preferred starting point. ▷ Shape = $(1, ndim)$
  Generate n_{init} number of initial DOE. ▷ $(n_{init}, ndim)$
  $N = n_{init}$. ▷ No. of function calls.
  Construct metamodel $\hat{G}_k$ based on n_{init} DOE.
  Generate N_{mc} MCS samples of $\boldsymbol{z}$. ▷ (N_{mc}, z)
  Commencing global enrichment stage:
  Global convergence = **False**
  while *Global convergence* = **False do**
    Generate n_c candidates of design variables $\boldsymbol{d}$. ▷ (n_c, n)
    Combine $\boldsymbol{d}$ with $\boldsymbol{z}$ into $\mathfrak{C}_q$, Eq. 3.41 to 3.42. ▷ $(N_{mc}, ndim)$
    For every $\boldsymbol{d}$ in $\mathfrak{C}_q$, compute $\mathcal{G}_k(\boldsymbol{d}, \boldsymbol{z})$, Eq. 3.39. ▷ $(N_{mc}, 1)$
    Compute the quantile $q_{\alpha k}$ of each $\mathfrak{C}_q$, Eq. 3.40. ▷ $(n_c, 1)$
    Identify the input $(\mathbf{x}_\alpha, \mathbf{z}_\alpha)$ of $q_{\alpha k}$, Eq. 3.43. ▷ $(1, ndim)$
    Compute $\mathcal{U}$ from each $(\mathbf{x}_\alpha, \mathbf{z}_\alpha)$ in $\mathfrak{C}_q$, Eq. 3.44. ▷ $(n_c, 1)$
    Evaluate η_{glo}, Eq. 3.46.
    if $\eta_{glo} < \bar{\eta}_{glo}$ **then**
      Global convergence = **True**.
    else
      Global convergence = **False**.
      Add the input point $(\mathbf{x}_\alpha, \mathbf{z}_\alpha)$ that gives the lowest $\mathcal{U}$ to the DOE.
      Improve the metamodel based on the new DOE.
      $N = N + 1$.
    end if
  end while
  Commencing local enrichment stage:
  $i = 0$
  $u_{rep} = 0$ ▷ No. of repeat in the first local enr.
  $\boldsymbol{d}_{i=0} = \boldsymbol{x}_0$
  Local convergence = **False**
  Reliability convergence = **False**
  Optimization convergence = **False**
  while *Local convergence* = **False** and *Reliability convergence* = **False** and *Optimization convergence* = **False do**
    Combine $\boldsymbol{d}$ with $\boldsymbol{z}$ into $\mathfrak{C}_q$, Eq. 3.41 to 3.42. ▷ $(N_{mc}, ndim)$
    Compute metamodel prediction $\hat{\boldsymbol{G}}_k(\boldsymbol{d}, \boldsymbol{z})$, Eq. 3.39. ▷ $(N_{mc}, 1)$
    Compute $\mathcal{U}$ for each $(\boldsymbol{d}, \boldsymbol{z})$ in $\mathfrak{C}_q$, Eq. 3.54 & 3.55. ▷ $(N_{mc}, 1)$

Identify the input $(\mathbf{x}_k, \mathbf{z}_k)$ that gives the lowest $\mathcal{U}$.
Compute metamodel prediction of $\hat{P}_f$ from $\hat{\mathbf{G}}_k(\boldsymbol{d}, \boldsymbol{z})$, Eq. 3.2.
Compute P_f^+ & P_f^- from $\hat{\mathbf{G}}_k(\boldsymbol{d}, \boldsymbol{z})$, Eq. 3.56.
if $i = 0$ and $\boldsymbol{d} = \boldsymbol{x}_0$ and $\hat{P}_f > \bar{P}_{f,k}$ **then**
    Switch to a new or default pref. starting point.
end if
Compute η_{P_f}, Eq. 3.58.
Compute the quantile $q_{\alpha k}$ of $\mathfrak{C}_q$, Eq. 3.40.
Compute $\hat{q}_\alpha^-$ & $\hat{q}_\alpha^+$, Eq. 3.47.
Compute η_q, Eq. 3.49 or 3.50.
if $\eta_q > \bar{\eta}_q$ **then**
    Local convergence = **False**.
    if $\boldsymbol{d}_i = \boldsymbol{d}_{i-1}$ **then**
        u_{rep} += 1 ▷ Count the repetition of enrichment at one point.
    end if
    if $u_{rep} \leq 5$ and $\hat{\mathbf{G}}_k(\mathbf{x}_k, \mathbf{z}_k) > 0.5$ **then** ▷ If 0.5 is the threshold in detecting an outlier.
        Add the point $(\mathbf{x}_k, \mathbf{z}_k)$ to the DOE.
        Improve the metamodel based on the new DOE.
        $N = N + 1$
    else
        $\boldsymbol{d}_{i+1} = \boldsymbol{x}_0$ ▷ Re-start iteration from $\boldsymbol{x}_0$ to avoid wasting function calls.
    end if
    Reset (1+1)-CMA-ES iteration parameters.
    Re-start the optimization process from the last iteration path.
    $i = 0$
    if $\eta_{P_f} \leq 20$ **then** ▷ If 20 is chosen as the threshold in adding a new pref. start. point.
        Add the point $(\mathbf{x}_k, \mathbf{z}_k)$ as a new pref. starting point candidate.
    end if
else
    Local convergence = **True**.
    $u_{rep} = 0$
    Perform (1+1)-CMA-ES optimization procedure. ▷ Alg. 1.
    $i = i + 1$
    $\boldsymbol{d} = \boldsymbol{y}$ = Alg. 1 output.
    if (1+1)-CMA-ES returns $\boldsymbol{d} = \boldsymbol{x}_0$ (Line 16 of Alg. 1) **then**
        Commencing starting point enrichment stage:
        Compute metamodel prediction $\hat{\mathbf{G}}_0(\boldsymbol{x}_0, \boldsymbol{z})$, Eq. 3.39. ▷ $(N_{mc}, 1)$
        Compute $\mathcal{U}$ from $\hat{\mathbf{G}}_0$, Eq. 3.54 & 3.55. ▷ $(N_{mc}, 1)$
        Identify the input $(\mathbf{x}_0, \mathbf{z})$ that gives the lowest $\mathcal{U}$.
        Add the point $(\mathbf{x}_0, \mathbf{z})$ that gives the lowest $\mathcal{U}$ value to the DOE.
        Improve the metamodel based on the new DOE.
        $N = N + 1$
        Reset (1+1)-CMA-ES iteration parameters.
        Re-start the optimization process from the preferred starting point.
        $i = 0$
    else
        Evaluate $f_\Delta(\boldsymbol{d}_i, \boldsymbol{d}_{i-1})$, Eq. 3.23.
        if $f_\Delta < f_{stop}$ and $\delta_{P_f} < 5\%$ **then**
            Optimization convergence = **True**
            if $\eta_{P_f} < \bar{\eta}_{P_f}$ **then**
                if $P_f \leq \bar{P}_{f,k}$ **then**
                    Reliability convergence = **True**
                else
                    Reliability convergence = **False**
                end if
            else

Commencing second local enrichment stage:
Compute metamodel predict. $\hat{\boldsymbol{G}}_2(\boldsymbol{d}, \boldsymbol{z})$, Eq. 3.39. ▷ $(N_{mc}, 1)$
Compute $\mathcal{U}$ from $\hat{\boldsymbol{G}}_2(\boldsymbol{d}, \boldsymbol{z})$, Eq. 3.54 & 3.55. ▷ $(N_{mc}, 1)$
Identify the input $(\boldsymbol{d}_{2nd}, \boldsymbol{z}_{2nd})$ that gives the lowest $\mathcal{U}$.
Add $(\boldsymbol{d}_{2nd}, \boldsymbol{z}_{2nd})$ to the DOE.
Improve the metamodel based on the new DOE.
$N = N + 1$
Reset (1+1)-CMA-ES iteration parameters.
Re-start the optimization process from the last DOE point.
$i = 0$
Add $(\boldsymbol{d}_{2nd}, \boldsymbol{z}_{2nd})$ as a new pref. starting point candidate.
Compute δ_{P_f}, Eq. 3.57.
if $\delta_{P_f} \leq 5\%$ **then**
Put $\boldsymbol{d}$ into $\boldsymbol{d}_{\textbf{shortlist}}$. ▷ Best candidates of $\boldsymbol{d}$.
end if
end if
else
Optimization convergence = **False**
end if
end if
end if
if $N = N_{max}$ **then**
Local convergence = **True**
Reliability convergence = **True**
Optimization convergence = **True**
$\eta_q = \infty$
Re-start the iteration until a converged point $\boldsymbol{d}_\infty$ is found.
Include $\boldsymbol{d}_\infty$ to the shortlist.
$n_{shortlist}$ = Total number of best $\boldsymbol{d}$ candidates in $\boldsymbol{d}_{shortlist}$.
Evaluate $\hat{P}_f$ of each $\boldsymbol{d}$ in $\boldsymbol{d}_{\textbf{shortlist}}$ with the latest $\hat{G}_k$. ▷ $(n_{shortlist}, 1)$
Evaluate δ_{P_f} of each $\boldsymbol{d}$ in $\boldsymbol{d}_{\textbf{shortlist}}$ with the latest $\hat{G}_k$. ▷ $(n_{shortlist}, 1)$
The optimal design $\boldsymbol{d}^* = \boldsymbol{d}$ that gives the lowest δ_{P_f}.
n_{best} = Total number of $\boldsymbol{d}$ that give the lowest δ_{P_f}.
if $n_{best} > 1$ **then**
Compute $f(\boldsymbol{d})$ for each $\boldsymbol{d}$ that gives the lowest δ_{P_f}.
$\boldsymbol{d}^* = \boldsymbol{d}$ that gives the lowest $f(\boldsymbol{d})$.
end if
end if
end while
return $\boldsymbol{d}^*$ ▷ The optimal design.
end procedure

Muhammad Rayyan