Spring 2016

The evaluation of sequential optimization and reliability analysis

Guannan Liu

Department:

THE EVALUATION OF SEQUENTIAL OPTIMIZATION AND RELIABILITY ANALYSIS

by

GUANNAN LIU

A THESIS

Presented to the Faculty of the Graduate School of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

In Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

2016

Approved by

Xiaoping Du, Advisor
Xuerong Wen
Serhat Hosder

ABSTRACT

Sequential Optimization and Reliability Assessment (SORA) has been used for more than one decade for reliability-based design (RBD), but comprehensive theoretical studies on its performance have not been conducted. Further investigations on its performance are still needed. The objective of this thesis is to evaluate the performance of SORA for various testing problems. The performance of SORA evaluated in this thesis includes (1) accuracy, (2) efficiency, and (3) convergence behavior or robustness with numerical testing problems. SORA is evaluated with comparison with other major RBD methodologies. The testing problems are in different scales (numbers of design variables, random variables, and reliability constraints), with different distribution types (normal or non-normal distributions), and different nonlinearity of limit-state functions. This evaluation study focuses more on efficiency, which is measured by the number of limit-state function calls. The robustness of SORA is also improved by correcting a sign problem for strength-type random variables that are log-normally distributed. Through the thorough evaluation of SORA, this research helps a better understanding of SORA and other RBD methodologies, offers a better guidance for selecting RBD methodologies, and suggests possible ways for improving RBD.

ACKNOWLEDGMENTS

First, I would like to express my deepest appreciation to my advisor, Professor Xiaoping Du, for his expertise, assistance, and patience along the way finishing this research and thesis. I would also like to thank the committee members, Professors Serhat Hosder and Xuerong Wen, for their assistance, support and suggestions.

Second, I would like to thank my team members, Mr. Zhifu Zhu and Mr. Zhengwei Hu for their helpful suggestions.

Finally, I would like to thank my parents, Zhilong Liu and Xiaohong Zhao, for all their support and encouragement. I would like to thank my wife, Hongwan Li, and my daughter, Lea Liu, for helping and supporting me as I completed the work.

TABLE OF CONTENTS

Page

LIST OF ILLUSTRATIONS

Figure Page

LIST OF TABLES

Table Page

1. INTRODUCTION

The objective of this research is to evaluate the effectiveness of the Sequential Optimization and Reliability Assessment (SORA) method. SORA is a methodology of reliability-based design (RBD). Although it has been used more than one decade for RBD, comprehensive theoretical studies on its performance have not been conducted. Further investigations on its performance are still needed. To achieve the objective, this work uses numerical testing problems to evaluate SORA in the aspects of accuracy, efficiency, and convergence behavior.

This section provides the background, research need, and organization of this thesis.

1.1. BACKGROUND

Engineers always strive to optimize the performance of the product they design. For example, they try to maximize the strength, efficiency, and life, and minimize the cost and energy consumption. To achieve this goal, engineers frequently employ optimization in the design process.

Engineers, however, are always surrounded by uncertainty because it is ubiquitous in every part of an engineering system, and in every step of the design process. Uncertainty could result from modeling errors, physical variations, and environmental changes. Uncertainty has been considered as a significant phenomenon in almost all the real-world systems [1, 2]. Due to the uncertainty, the performance of final products could be away from the designed or expected performance. This may

significantly affect the reliability, robustness, quality, and safety of a product. The effects of uncertainty, fortunately, could be quantified by reliability analysis and could be mitigated by RBD. Reliability methodologies have therefore attracted increasing attention and have been increasingly used in product design.

Reliability is the ability that a product performs its intended function without failures. Reliability is usually quantified by the probability of such ability; in other words, reliability is the probability that a product performs its intended function without failures. As engineering systems become more and more complex, their failures also become increasingly significant, making modeling uncertainty and reliability more critical [3]. Reliability has become a core consideration during the design process for many engineering systems.

There are two major areas of reliability applications. The first is reliability analysis and the second is RDB. The task of the first area is to estimate, evaluate, or calculate the reliability for a given component, system, or process. This can be used to access if the reliability satisfies the reliability requirement. If not, design changes are made, and reliability analysis is performed again. During this process, RDB plays an important role.

RBD is a methodology that ensures the probability of failure be at the acceptable small level with respect to random parameters. It usually minimizes the cost of a product and at the same time maintains the reliability requirement. This is done through optimization. By changing design variables, the cost is reduced in the condition that the

reliability target is met. RBD has been widely used in many engineering and scientific fields.

RBD considers reliability as the probability of constraint satisfaction. A number of reliability analyses are needed during the optimization process. Once a new design point is generated, the reliability of a constraint at the new design point is evaluated. The reliability analysis relies on limit-state functions, which are functions of design variables and random variables and produce responses of constraints. The reliability analysis also calls a number of limit-state functions in evaluating the reliability. As a result, reliability analysis involves an iterative process. Consequently, RBD requires two loops. One loop is the optimization itself, and the other loop is the reliability analysis.

If a RBD problem is solved directly, the reliability analysis loop is embedded in the optimization loop. Then the reliability analysis is the inner loop and the optimization is the outer loop. This method is therefore called the Double-Loop RBD [4]. The computational cost of Double-Loop RBD is usually intensive. For example, if the optimization outer loop requires 50 iterations and the reliability analysis inner loop requires another 50 iterations for each of 10 limit-state functions, the total number of function calls will be $50 \times 50 \times 10 = 25{,}000$. If a limit-state function is a black-box simulation model, such as a finite-element-analysis (FEA) model, the computational time would be prohibitively high [5].

To improve the efficiency of RBD, the Single-Loop RBD method has been devolved. This method avoids the nested structure by converting the reliability analysis into an equivalent deterministic optimization problem. Specifically, the method includes

constraints that are the Karush-Kuhn-Tucker optimality conditions [6] of the reliability analysis. In many cases, the Single-Loop RBD method is more efficient than the Double-Loop RBD method, but the number of design variables are much higher than that of the Double-Loop RBD method [6]. With the increased numbers of design variables and constraint functions, solving the optimization model of the Single-Loop RBD method is more difficult.

SORA is a RBD method that takes advantages of both the Single-Loop RBD and Double-Loop RBD methods. It eliminates the double loop procedure and performs the optimization loop and reliability analysis loop separately and sequentially. If no convergence is reached after a cycle of optimization and reliability analysis, the optimization model is reformulated and then the next cycle is run. This process repeats until convergence. With the sequential cycles of decoupled optimization and reliability analysis, SORA is much more efficient than the Double-Loop RBD method [5]. In many cases, SORA is more robust than the Single-Loop RBD method.

1.2. RESEARCH NEEDS

Many RBD methodologies have been developed and are available for engineers to use. SORA is one of the methodologies. But there is no such a single RBD methodology that would perform well for all problems. Moreover, applications are different with different numbers of random variables, different nonlinearity levels in limit-state functions, different distribution types of random variables, and different degrees of dependencies between random variables. It is therefore necessary to understand the performance of each RBD methodology and its application scope.

SORA is a commonly used RBD methodology and has been applied to engineering applications, such as those in mechanical product development and structural optimization [7-9]. It has also been adopted by the commercial software Hyper Study (a design tool for optimization and reliability), which is widely used in automotive, aerospace, and structural applications. Although SORA is in general more efficient than many double-loop RBD methodologies, its performance, however, is still not well understood. There is therefore a research need to thoroughly evaluate the performance of SORA. The evaluation can then better assist engineers to select the best RBD methodology for their specific applications. It can also help improve the performance of SORA.

1.3. OBJECTIVE OF THIS RESEARCH

This research aims to provide a solution to the research need discussed in Section 1.2. The objective of this research is to evaluate the performance of SORA with a number of testing problems. The problems are selected from journal articles and they are different in terms of scale (numbers of design variables, random variables, and reliability constraints), distribution types (normal and non-normal distributions), and nonlinearity of limit-state functions.

The performance of SORA this research evaluates includes (1) accuracy, (2) efficiency, and (3) convergence behavior or robustness. Since the efficiency is the major concern for a RBD methodology, this evaluation study focuses more on efficiency, and the number of limit-state function calls is used as a measure of efficiency.

1.4. ORGANIZATION OF THIS THESIS

The organization of this thesis is as follows:

In Section 2, reliability analysis is reviewed, including First Order Reliability Method (FORM), the direct MPP search, the inverse MPP search, and Monte Carlo simulation (MCS). Then RBD methodologies are also reviewed. They include the Double-Loop RBD method, Single-Loop RBD method, and SORA.

Section 3 reports the major results of this research. It begins with the evaluation methodologies followed by a number of testing problems. The results of the evaluation, including the accuracy, efficiency, and robustness of SORA, are also provided. Conclusions are made based on the evaluation results and are given in this section.

Section 4 summaries the results with conclusions and possible future work.

2. REVIEW OF RELIABILITY-BASED DESIGN

The major reliability-based design (RBD) methodologies are reviewed in this section. The methodologies include the Double-Loop methods, Single-Loop methods, and Sequential Single-Loop methods. SORA belongs to Sequential Single-Loop methods and will be reviewed in detail.

2.1. OVERVIEW OF RBD

RBD methods are based on optimization. A general optimization model is given by

$$\begin{gathered}\text{Minimize: } \mathrm{f}(\mathbf{d}) \\ \text{Design variable DV} = \{\mathbf{d}\} \\ \text{Subject to: } g_i(\mathbf{d}) \geq 0, \mathrm{i} = 1,2,\dots,m\end{gathered} \tag{2.1}$$

In the above model [2, 10], $\mathbf{d}$ is a vector of design variables. For example, for a gear design, design variables could be the diameter, the width, the material, and the number of teeth of the gear. f is the objective function, such as the cost, life, quality, and efficiency. $g_i(\mathbf{d})$, $(i = 1,2,\dots,m)$, are constraint functions. For example, if $g_i(\mathbf{d}) \leq 0$, the factor of safety of a component is greater than the specified number.

The traditional optimization, however, does not account for any uncertainty. In reality, uncertainty is ubiquitous in almost all engineering applications. Uncertainty may come from random material properties, manufacturing impression, or stochastic operation conditions [11].

Without considering uncertainty, the optimization design obtained from the tradition optimization design may be risky. In other words, the likelihood of satisfying

design requirements in constraint functions will be relatively low. In many cases, the likelihood or probability of a constraint satisfaction is about only 50%. To deal with uncertainty, engineers use RBD, which guarantees that all design requirements are satisfied at required reliability levels [12].

RBD usually minimizes a cost-type function and at the same time satisfies reliability requirements, expressed as design constraints. A typical RBD model is given below.

$$\begin{gathered}\text{Minimize: } f(\mathbf{d}, \mathbf{X}, \mathbf{P}) \\ \text{Design Variable } DV = \{\mathbf{d},\ \boldsymbol{\mu_X}\} \\ \text{Subject to: } \Pr\{g_i(\mathbf{d}, \mathbf{X}, \mathbf{P}) \geq 0\} \geq R_i, i = 1,2, \dots, m\end{gathered} \tag{2.2}$$

where f is a cost-type objective function. For example, it could be the actual cost of a product, the weight of an aircraft wing, the material usage of a component, or the volume of a pressure tank.

$\mathbf{X}$ is a vector of random design variables vector. Their means $\boldsymbol{\mu_X}$ are part of the design variables. $\mathbf{P}$ is a vector of random parameters . $\Pr\{g_i(\mathbf{d}, \mathbf{X}, \mathbf{P}) \geq 0\}$ $(i = 1,2, \dots, m)$ are reliabilities, and R_i $(i = 1,2, \dots, m)$ are required reliabilities. For example, if a constraint is the design margin, which is the strength subtracted by stress, the associated required reliability may be set to 99.999%. This means that the probability of the factor of safety greater than 1 is 99.999%.

The benefits of RBD are multifold. (1) It ensures that the reliability requirement could be met, thereby producing highly reliable products. (2) It reduces the chance of failures and risk, resulting in cutting operation cost and product lifecycle cost. (3) It helps

make important decisions, such as determination of warranty policy and maintenance. Compared to deterministic optimization, however, there are many challenges in solving a RBD model. The major challenge is to reach a good balance between accuracy and efficiency. Solving the RBD model is expensive because reliability analysis is called many times, and reliability analysis also needs to call limit-state function several times. But limit-state functions are sometimes computationally expensive. To this end, many RBD methodologies have been developed. Typical RBD methodologies are reviewed in the rest of this section, including Double-Loop RBD in Subsection 2.3, Single-Loop RBD in Subsection 2.4, and SORA in Subsection 2.6. Before the review of the RBD methodologies, reliability analysis is reviewed in Subsection 2.2 because it is needed by all RBD methodologies.

2.2. RELIABILITY ANALYSIS

The task of reliability analysis is to calculate the reliability. Let all the random input variables be $\mathbf{X} = (X_1, X_2, \ldots X_n)$ and their joint probability density function (PDF) be $f_{\mathbf{X}}(\mathbf{x})$ [2]. The probability of failure is calculated by

$$p_f = \Pr\{g(\mathbf{X}) \leq 0\} \tag{2.3}$$

p_f can be theoretically computed by the following integration

$$p_f = \int_{g(\mathbf{x}) \leq \mathbf{0}} f_{\mathbf{X}}(\mathbf{x})\, \mathbf{dx} \tag{2.4}$$

Then the reliability is given by

$$R = 1 - p_f \tag{2.5}$$

It is difficult to evaluate the above multidimensional probability integral numerically [13]. Approximation methods are therefore always used in RBD. The most popular reliability analysis method is the First Order Reliability Method (FORM), which is reviewed below.

2.2.1. First Order Reliability Approach Method (FORM). Most RDB methods use FORM to calculate reliability, and so does SORA. FORM is briefly reviewed in this subsection. All the random variables herein are assumed independent [14, 15].

FORM at first transforms general random variables **X** to standard normal random variables **U**. This is usually a nonlinear transformation. For example, for a normal distribution, $X \sim N(\mu, \sigma^2)$, the transformation is given by

$$U = \frac{X-\mu}{\sigma} \quad \text{or} \quad X = \mu + \sigma U \tag{2.6}$$

After the transformation, the limit-state function becomes

$$\mathrm{g}(\mathbf{U}) = 0 \tag{2.7}$$

With the first order Taylor expansion, the limit-state function becomes

$$\mathrm{g}(\mathbf{U}) \approx \mathrm{g}(\mathbf{u}^*) + \nabla \mathrm{g}(\mathbf{u}^*)(\mathbf{U} - \mathbf{u}^*)^T \tag{2.8}$$

where $\mathbf{u}^*$ is the expansion point and is called the most probable point (MPP). It is a vector given by

$$\mathbf{u}^* = (u_1^*, u_2^*, \cdots, u_n^*) \tag{2.9}$$

After $\mathbf{u}^*$ is found, the probability of failure is:

$$p_f = \Phi(-\beta) \tag{2.10}$$

where β is reliability index, and $\Phi(\cdot)$ is the standard normal cumulative distribution function. β is given by

$$\beta = \sqrt{\sum_{i=1}^{n} u_i^*} \tag{2.11}$$

2.2.2. Direct Reliability Analysis and MPP Search. To find the $\boldsymbol{p_f}$ directly, the MPP is needed, and this requires the MPP search.

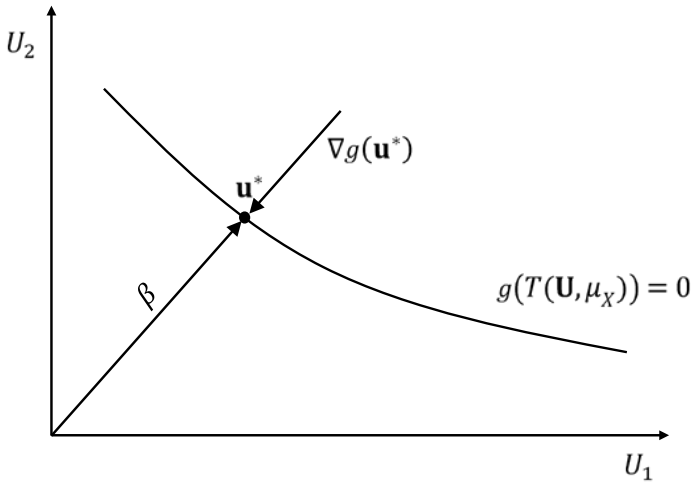

Figure 2.1 Limit-state Function and MPP

Figure 2.1 shows that the MPP is the shortest distance point form the origin to the limited state $g(\mathbf{U}) = 0$. As indicated in Eq. (2.11), the reliability index β is such a distance [16]. The MPP search needs the gradient, which is given by

$$\nabla g(\mathbf{u}^*) = \left(\frac{\partial g(\mathbf{U})}{\partial U_1}, \frac{\partial g(\mathbf{U})}{\partial U_2}, \cdots, \frac{\partial g(\mathbf{U})}{\partial U_n}\right) \tag{2.12}$$

The MPP search algorithm is given below

$$\mathbf{u}^* = -\beta^* \nabla g(\mathbf{u}^*)/|\nabla g(\mathbf{u}^*)| \tag{2.13}$$

Defined $\boldsymbol{\alpha}$ to be

$$\boldsymbol{\alpha}(\mathbf{u}^*) = \nabla g(\mathbf{u}^*)/|\nabla g(\mathbf{u}^*)| \tag{2.14}$$

Thus

$$\mathbf{u}^* = -\beta \boldsymbol{\alpha}(\mathbf{u}^*) \tag{2.15}$$

Based on Eq. (2.13), the MPP search algorithm is derived as follows.

$$\begin{cases} \beta^{k+1} = \beta^k + \nabla g\left(\mathbf{u^k}\right)/\left|\nabla g\left(\mathbf{u^k}\right)\right| \\ \mathbf{u^{k+1}} = -\beta^{\mathrm{k+1}}\boldsymbol{\alpha}(\mathbf{u}) \end{cases} \tag{2.16}$$

After this process converges, Eq. (2.10) is used to calculate p_f.

The stopping criterion of the MPP search is that the difference of $\mathbf{u}$ between two consecutive cycles is small enough. Figure.2.2 give the flowchart of the MPP search.

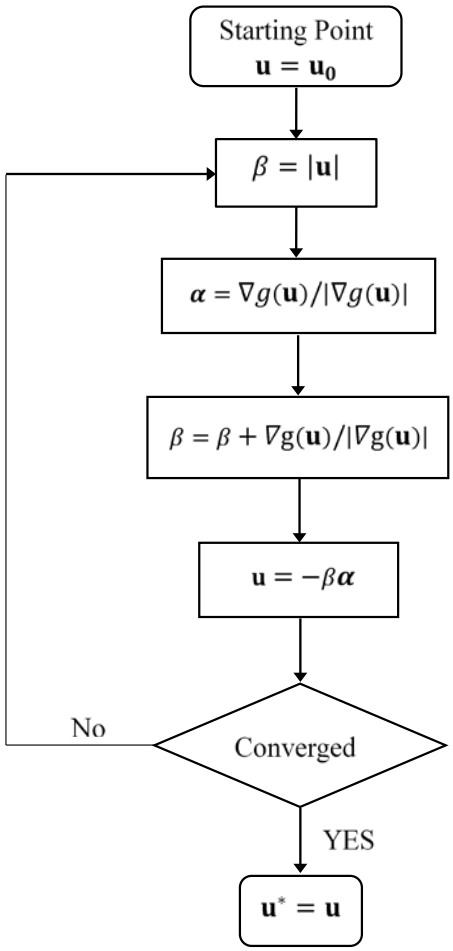

Figure 2.2. Flowchart of the MPP Search

2.2.3. Inverse Reliability Analysis and MPP Search. The MPP search discussed in Subsection 2.2.2 is for the direct reliability analysis. In RBD, inverse reliability analysis is also used.

The inverse MPP method finds the function value corresponding to the required reliability, and this value is called R-percentile. As shown in Fig 2.3 (a), the required probability of constraint function greater than zero is *R*, written as $\Pr(g \geq 0) \geq R$. If we use the percentile of the constraint function as a constraint condition, the constraint condition will be as $\Pr(g \geq g^R) = R$, as showed in Fig 2.3 (b). Therefore, the new

constraint function that satisfies the require reliability becomes $g^R \geq 0$ [5]. g^R is given by

$$g^R = g(\mathbf{u}^*) \tag{2.17}$$

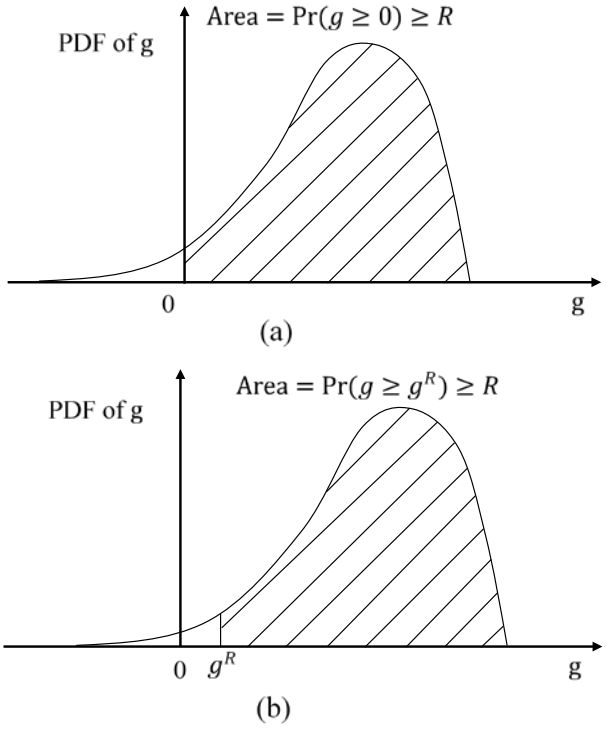

Figure 2.3. (a) PDF of Constraint Function; (b) R-Percentile

where $\mathbf{u}^*$ is the inverse MPP. The inverse MPP search algorithm is given below.

$$\begin{cases} \boldsymbol{\alpha}(\mathbf{u}^{\mathrm{k}}) = \nabla g(\mathbf{u}^{\mathrm{k}}) / \left|\nabla g(\mathbf{u}^{\mathrm{k}})\right| \\ \mathbf{u}^{\mathrm{k+1}} = -\beta^* \boldsymbol{\alpha}(\mathbf{u}^{\mathrm{k}}) \end{cases} \tag{2.18}$$

where

$$\beta = \Phi^{-1}(R) \tag{2.19}$$

The stopping criterion of the inverse MPP search is that the difference of **u** between two consecutive cycles is small enough.

Fig.2.4 gives the flowchart of the inverse MPP search.

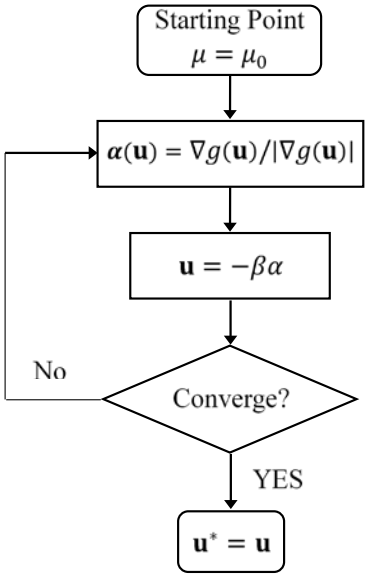

Figure 2.4 Flowchart of the inverse MPP search

2.2.4. Monte Carlo Sampling Method. Monte Carlo Simulation (MCS) is widely used in reliability analysis. This method generates sampling points which are associated with the distributions of random variables. It can deal with all distribution types, a large number of random variables, and highly nonlinear models [17, 18].

MCS evaluates the limit-state function at the samples of input variables. It then counts the number of sample points in the failure region. The probability of failure is then estimated by

$$p_f = \frac{N_f}{N} \tag{2.20}$$

where N_f is number of sampling points in the failure region, and N is the total number of sample points. MCS is accurate if the sample size N is large enough. This method can be used for accuracy comparison due to its high accuracy [19].

Because the simulation process draws random sample points according to the distribution of random variables, most of the sample points will reside near the mean value of the joint distribution. This means that there may be few sample points in the failure region if the reliability is high. Therefore, MCS needs a large sample size to ensure that there are enough sample points in the failure region. Since the sample size is large for high reliability problems, MCS is computationally expensive.

2.3. DOUBLE-LOOP RBD

Double-loop RBD solves the RBD model in Equation (1.2) directly. As a result, there are two nested loops. The first loop is the overall optimization, and it is responsible for seeking for the optimal design variables. The second loop is the reliability analysis, whose task is to calculate the reliability of each constraint functions and then pass the results to the optimization loop.

The flowchart is shown in Fig 2.6. The figure indicates that the inner reliability analysis loop is nested in the outer optimization loop.

The outer loop is the deterministic optimization loop which generates design variables $\mathbf{d}$ and $\boldsymbol{\mu}_{\mathbf{X}}$. It then calls the inner reliability analysis loop to calculate reliabilities of all constraint functions $\Pr\{g_i(\mathbf{d},\ \mathbf{X_{MPP}},\ \mathbf{P_{MPP}}) \leq 0\} \geq R_i, i = 1,2,\dots,m$, for the given

set of **d** and $\boldsymbol{\mu_X}$. If all the constraint functions are satisfied with the required reliabilities, and the change in the objective function is small enough, an optimal solution is found.

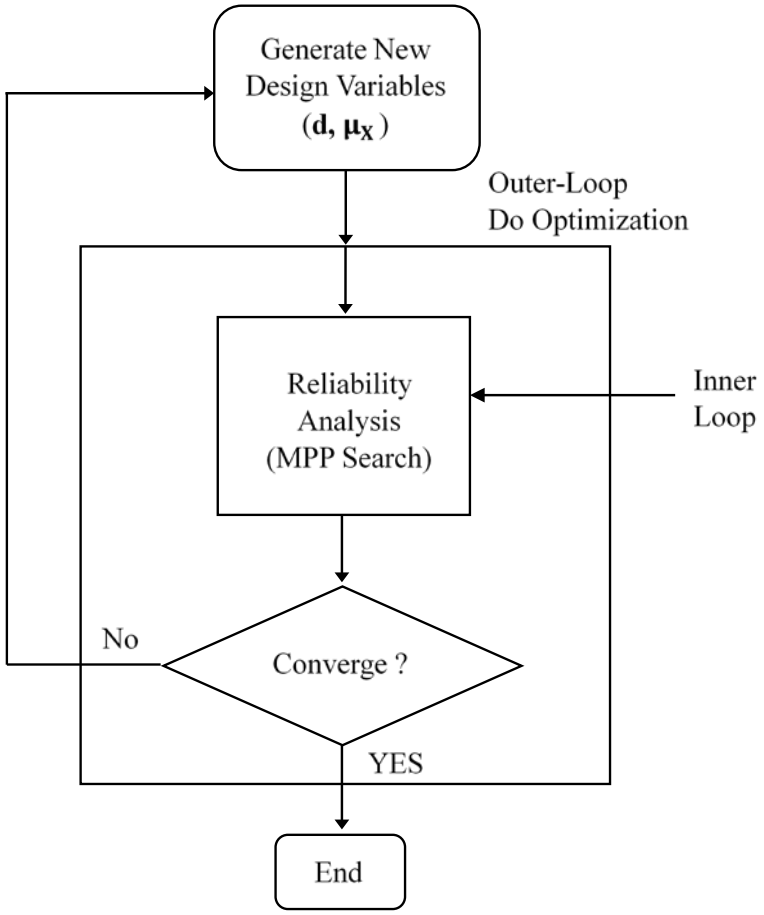

Figure 2.5. Double-loop Flowchart

The double-loop procedure requires a large number of function calls. This leads to intensive computations, which are not practical for industrial applications. To improve the computational efficiency, many methods have been proposed, including the Single-Loop method discussed below.

2.4. SINGLE-LOOP RBD

The flowchart of the Single-Loop RBD method is shown in Fig 2.7, which indicates that there is only one optimization loop. This single-loop structure avoids the high computational cost caused by the nested double-loop structure. The RBD model is given below

$$\begin{gathered}\text{Minimize: } f\left(\mathbf{d}, \boldsymbol{\mu}_{\mathbf{X}}, \boldsymbol{\mu}_{\mathbf{P}}\right) \\ \text{DV}=\left\{\mathbf{d}, \boldsymbol{\mu}_{\mathbf{X}}, \mathbf{u}_i^*\right\}, i=1,2, \ldots, m \\ \text{Subject to: MPP search conditions,} \\ \text{where } \mathbf{u}_i^* \text{is the MPP of constraint function.}\end{gathered} \tag{2.21}$$

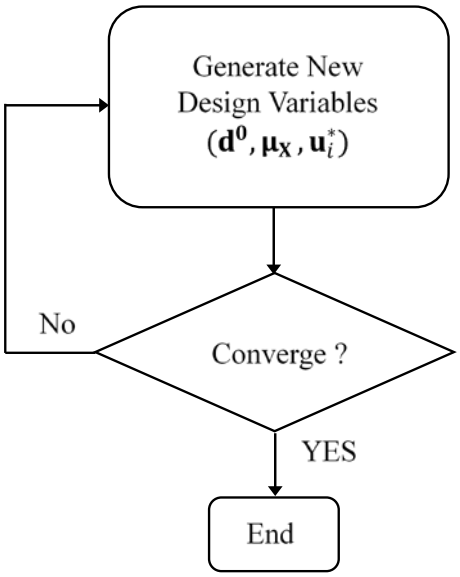

Figure 2.6. Single-loop flowchart

Since there is no reliability loop, The RBD problem is converted into an equivalent deterministic optimization problem by enforcing the Karush-Kuhn-Tucker optimality conditions of the MPP search [6]. Under these conditions, all $\mathbf{u}_\mathbf{i}^*$ vectors are set

as design variables. Then there is no need to perform reliability analysis anymore. A nonlinear optimization algorithm can be sued to solve the Single-Loop RBD problem.

This method may reduce the cost of computations without the nested structure. But for the same RBD problem, the Single-Loop method has more design variables than the Double-Loop method by consider $\mathbf{u}_\mathbf{i}^*$ as design variables. If there is a large number of constraint functions and a large number of random variables, the design space will be extremely large. This may affect the efficiency of the optimization. In addition, this method may not be robust for some RBD problems, because of the equality constraints of the MPP search.

2.5. SORA

The flowchart of SORA is given in Fig 2.8. The reliability analysis loop is completely decoupled from the optimization loop. The RBD model is the same as Eq. (2.2).

SORA performs RBD by sequential cycles of deterministic optimization and reliability analysis. Each cycle starts form deterministic optimization. Then the optimal point is passed to reliability analysis, which then performs the inverse MPP search. The MPPs are used to reformulate the constraint functions for the next cycle. The reformulated constraint functions help improve reliability.

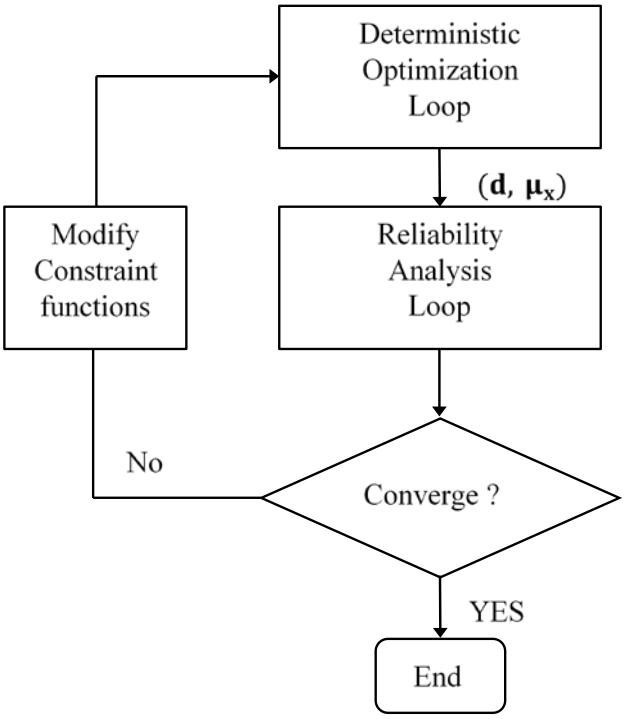

Figure 2.7. SORA flowchart

In the first cycle, the means of the random design variables are used for deterministic optimization before reliability analysis. Then the inverse MPP method is used in the reliability analysis based on the deterministic optimization optimal point. In the next cycle, the MPPs $(\mathbf{x}_i^*, \mathbf{p}_i^*)$ are used to formulate a new deterministic optimization problem [5, 20], where constraint functions will be move quickly to the feasible region based on the MPP information.

For the k^{th} cycle, constraint i is formulated as

$$g_i(\mathbf{d}, \mathbf{u}_i^*) \geq 0 \tag{2.22}$$

where $\mathbf{u}_i^*$ is the MPP of $\mathbf{X}$ and $\mathbf{P}$ in the $\mathbf{U}-$ Space. $\mathbf{u}_i^*$ has to be transformed into $\mathbf{x}_i^*$ and $\mathbf{p}_i^*$. Assume the transformation for $\mathbf{x}_i^*$ is $\mathrm{T}(\mathbf{u}_i^*)$, then the constraint is given by

$$g_i(\mathbf{d}, \mathrm{T}(\mathbf{u}_i^*, \boldsymbol{\mu}_\mathbf{X}), \mathbf{p}_i^*) \geq 0, i = 1,2, \dots, m \tag{2.23}$$

The optimization model in cycle $(k+1)^{th}$ is then reformulated as

$$\begin{gathered} \text{Minimize: } f(\mathbf{d}, \boldsymbol{\mu}_\mathbf{X}, \boldsymbol{\mu}_\mathbf{P}) \\ \text{DV}=\{\mathbf{d}, \boldsymbol{\mu}_\mathbf{X}\} \\ \text{Subject to: } g_i(\mathbf{d}, \mathrm{T}(\mathbf{u}_i^*, \boldsymbol{\mu}_\mathbf{X}), \mathbf{p}_i^*) \geq 0, i = 1,2, \dots, m \end{gathered} \tag{2.24}$$

The stopping criteria of SORA are as follows: 1). The difference of the objective function is small enough. 2). All the constraints are satisfied.

The following measures are used in order to increase the efficiency. 1). If the inverse MPPs of probabilistic constraints in two consecutive cycles are extremely close, use the inverse MPP obtained from the last cycle as the initial guess of the inverse MPP in the following cycle. It can decrease the computational effort for the MPP search. 2). The starting point of the optimization of one cycle is considered as the optimum point of the previous cycle. 3). After one cycle of optimization ends, if the results do not change or slightly change, the MPP in the current cycle will be the same or at least very similar to that in the last cycle. Therefore, there is no need to search for the MPP for the probabilistic constraint in the current reliability assessment.

In sum, SORA does not calculate reliability directly. The reliability is evaluated only at a particular level (R-percentile), and searching for the inverse MPP is more efficient. An efficient and robust inverse MPP search algorithm is also used. The most important contribution for high efficiency is the sequential cycles of optimization and reliability analysis [5].

3. EVALUATION OF SORA

The objective of this research is to evaluate the effectiveness of the Sequential Optimization and Reliability (SORA) method. This section discusses the methodologies that are used to evaluate SORA and also reports the evaluation results from testing problems. Conclusions are also given based on the evaluation results.

3.1. METHODOLOGIES FOR THE EVALUATION OF SORA

The major approach is to use selected testing problems to evaluate the performance of SORA. The evaluation criteria are listed below.

3.1.1. Efficiency. The number of total limit-state function calls is used as a metric for the efficiency. The number includes those for both deterministic optimization and reliability analysis. In real engineering applications, a limit-state function may be a Computer Aided Engineering (CAE) model, such as a finite element analysis (FEA) model, which is computationally expensive. It may take minutes, hours, or even days to run the model. An efficient RBD method minimizes the number of limit-state function calls. Using the number of function calls is better than that of the computational time because the latter is largely depends on the computer that is used for the RBD problem. Efficiency is the most important criterion considered in the evaluation.

3.1.2. Robustness. Robustness herein is defined as the ability that SORA could successfully identify an optimal solution for a RBD problem. Such ability is evaluated by observing if SORA could converge to an optimal solution. If not, the cause of divergence is recorded and investigated.

3.1.3. Accuracy. The accuracy is for the reliability analysis. After an optimal point is found, all reliability constraints are satisfied. Since the reliability is calculated by FORM, an error is unavoidable even though FORM has good accuracy. Mote Carlo simulation (MCS) with a large sample size is used as a benchmark for the accuracy assessment. The accurate reliabilities associated with all active reliability constraints

are computed by MCS and are then compared with the required reliabilities. The differences are considered as errors. The accuracy comparison is only for active constraints. The deterministic optimization method used in this study is active-set. The MPP search method used in this study is the direct MPP search and inversed MPP search methods [16, 21].

3.2. TESTING PROBLEMS AND RESULTS

Testing problems are carefully selected. Several representative testing problems and the associated evaluation results are reported in this subsection. Three methods are compared, including the Double-Loop Method with direct reliability analysis (DL-Direct), Double-Loop Method with inverse reliability analysis (DL-Inverse), and SORA.

3.2.1. Testing Problem 1. A cantilever beam is subjected to two independent random forces $\boldsymbol{P_x}$ and $\boldsymbol{P_y}$ as showed in Fig 3.1 [22, 23].

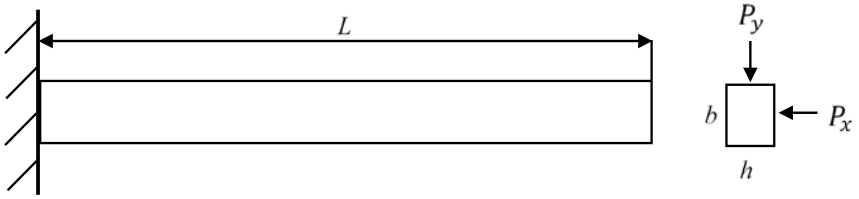

Figure 3.1. Cantilever Beam

There are two failure modes. The first failure mode is the excessive stress, and the limit-state function is given by

$$g_1(\mathbf{X}, \mathbf{P}) = S - \frac{6L}{bh}\left(\frac{P_x}{b} + \frac{P_y}{h}\right) \tag{3.1}$$

where S is the random yield strength, $L = 100$ in is the length of the beam.

The second failure mode is the excessive deflection, and the limit-state function is given by

$$g_2(\mathbf{X},\mathbf{P}) = D - \frac{4L^3}{E}\sqrt{\left(\frac{P_x}{b^3h}\right)^2 + \left(\frac{P_y}{h^3b}\right)^2} \tag{3.2}$$

where E is Young's modulus, $D = 2.5\ in$ is the allowed displacement value, $L = 100\ in$ is the length. The complete RBD model is given by:

$$\text{Minimize: } f(\boldsymbol{\mu}_{\mathbf{X}}) = \mu_b\mu_h L$$

Subject to:

$$\Pr\{g_1(\mathbf{X},\mathbf{P})\} = \Pr\left\{\frac{6L}{bh}\left(\frac{P_x}{b} + \frac{P_y}{h}\right) - S \le 0\right\} \le p_{f1} \tag{3.3}$$

$$\Pr\{g_2(\mathbf{X},\mathbf{P})\} = \Pr\left\{\frac{4L^3}{E}\sqrt{\left(\frac{P_x}{b^3h}\right)^2 + \left(\frac{P_y}{h^3b}\right)^2} - D \le 0\right\} \le p_{f2}$$

The design variables are b and h **.** They are given in Table 3.1. There are five cases for this problem. The normal distributions and log-normal distributions are involved. All the random variables for all the cases are given in following Tables. Both of the required reliabilities of the two constraints are: 0.9987 for Case 1, and 0.9999683 for Case 2 through 4.

Table 3.1 Bounds of Design Variables for Cases 1 through 5

Design variables	Lower bound	Upper bound
b	0.1 in	10 in
h	0.1 in	10 in

Table 3.2 Distributions of Random Variables for Cases 1 and 2

Design variables	Mean	Standard Deviation	Distribution Type
b	μ_b in	0.01 in	Normal
h	μ_h in	0.01 in	Normal
P_x	500 lb	100 lb	Normal
P_y	1000 lb	100 lb	Normal
E	29×10^6 psi	1.45×10^6 psi	Normal
S	4000 psi	2000 psi	Normal

Table 3.3 Distributions of Random Variables for Case 3

Design variables	Mean	Standard Deviation	Distribution Type
b	μ_b in	0.01 in	Normal
h	μ_h in	0.01 in	Normal
P_x	500 lb	100 lb	Log-normal
P_y	1000 lb	100 lb	Log-normal
E	29×10^6 psi	1.45×10^6 psi	Normal
S	4000 psi	2000 psi	Normal

Table 3.4 Distributions of Random Variables for Case 4

Design variables	Mean	Standard Deviation	Distribution Type
b	μ_b in	0.01 in	Normal
h	μ_h in	0.01 in	Normal
P_x	800 lb	100 lb	Log-normal
P_y	1000 lb	100 lb	Log-normal
E	29×10^6 psi	1.45×10^6 psi	Normal
S	15000 psi	2000 psi	Normal

Table 3.5 Distributions of Random Variables for Case 5

Design variables	Mean	Standard Deviation	Distribution Type
b	μ_b in	0.01 in	Normal
h	μ_h in	0.01 in	Normal
P_x	500 lb	100 lb	Log-normal
P_y	1000 lb	100 lb	Log-normal
E	29×10^6 psi	1.45×10^6 psi	Normal
S	40000 psi	2000 psi	Normal

The results of Case 1 are discussed in detail as follows: the optimal results from the three methods are given in Table 3.6, which shows that the three methods produce almost identical solutions. The slight differences are only due to numerical errors. As discussed previously, the efficiency is measured by the number of function calls, including both objective and constraint functions. The number of function calls are also listed in Table 3.6. SORA calls all functions 199 times, including 156 for deterministic optimization and 43 for reliability analysis. The double-loop RBD method with direct reliability (DL-Direct) and double-loop RBD method with inverse reliability analysis call functions 6447 and 301 times, respectively. SORA is therefore the most efficient method and DL-Inverse is more efficient than DL-Direct.

Table 3.6 Results for Case 1

Method	DL-Direct	DL-Inverse	SORA
b	2.4487	2.4508	2.4374
h	3.8878	3.8841	3.9057
Objective	9.5201	9.5192	9.5200
Function Call	6447	301	199 (156+43)
Error (constraint 1)	0.13 %	0.1302 %	0.1291 %

Table 3.7 SORA Convergence History for Case 1

Cycle	Design Variables	g_{mpp}	Objective function
1	(2.047, 3.746)	(-14379.40, -1.30)	7.668
2	(2.491, 3.811)	(152.913, -0.277)	9.495
3	(2.437, 3.905)	(-1.51×10^{-8}, 0.235)	9.520

The convergence history of SORA is shown in Table 3.7. The convergence history of objective function is shown in Fig 3.2, and the convergence history of the two constraints are also shown in Fig 3.3 and Fig 3.4. After three cycles, SORA converged. The first cycle involves deterministic optimization with mean values of all the random

variables, and it was therefore the conventional optimization. The limit-state function values (g_{MPP} in the table) from reliability analysis are all negative, meaning that the reliability requirements are not satisfied. As shown in the third cycle, g_{MPP} of the first constraint is almost zero, and g_{MPP} of the second constraint is positive. This means that, the first constrain is active and that the actual reliability is exactly at the required level. The reliability of the second constraint exceeds the required value because the g_{MPP} value is positive.

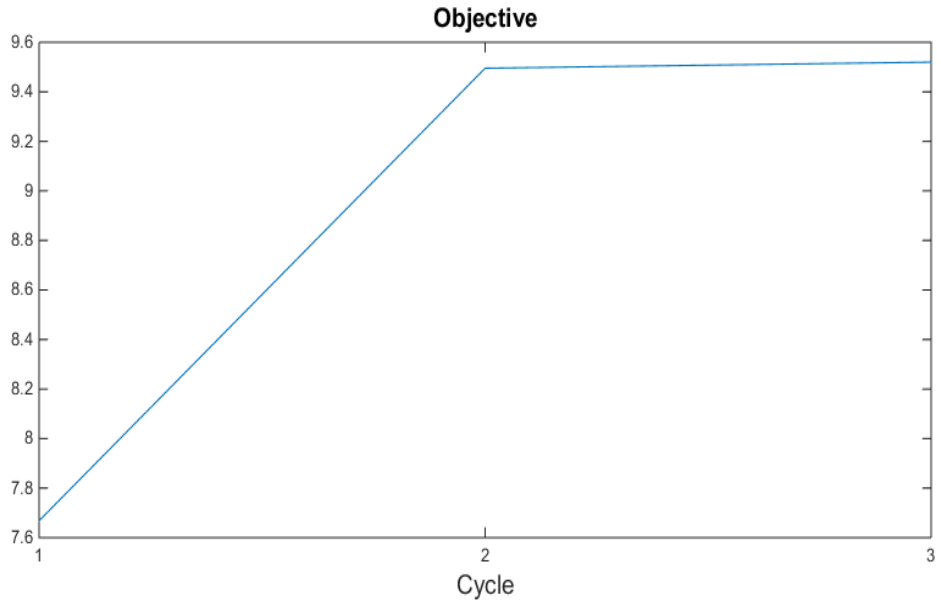

Figure 3.2 Convergence history of objective function

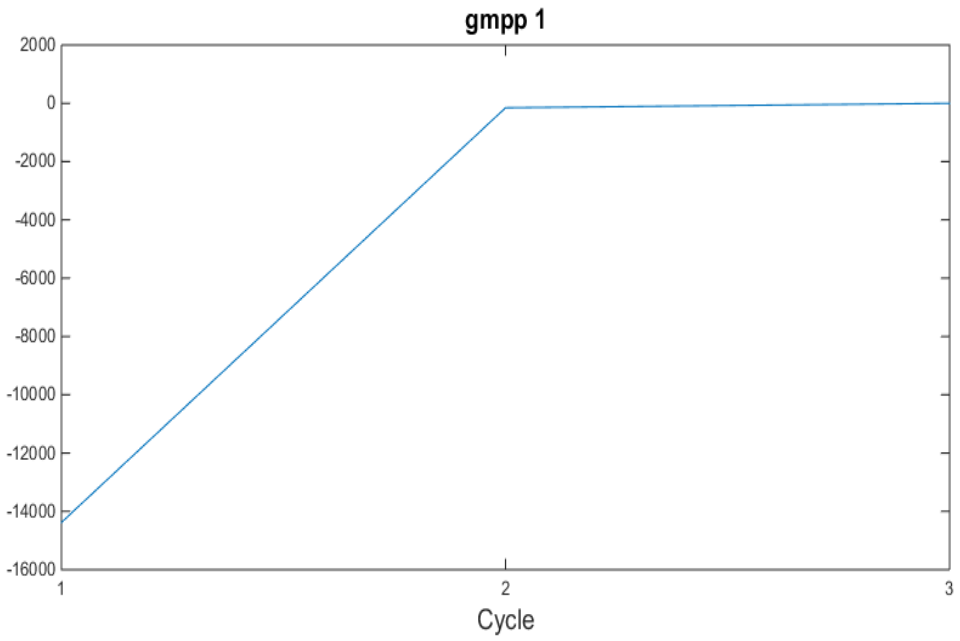

Figure 3.3 Convergence history of g_{1MPP}

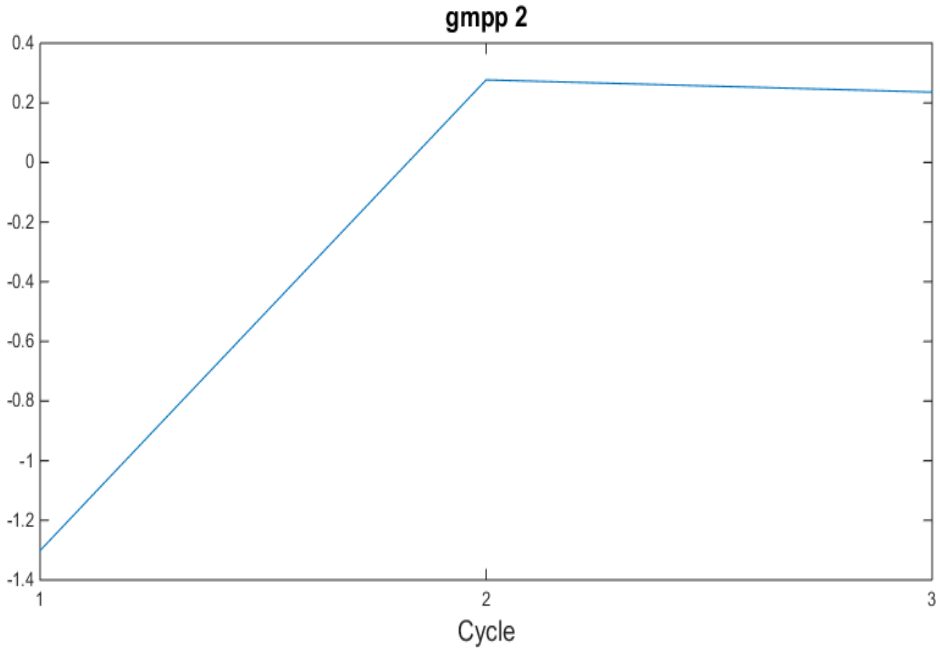

Figure 3.4 Convergence history of g_{2MPP}

MCS was also performed to check the accuracy. The sample size was taken as 10^7. With this large sample size, the MCS solution is considered accurate. Then the three optimal points from the three methods were plugged into MCS to calculate the reliability. Note that, only the reliability of the active constraint was calculated, and the relative error of the calculated reliability with respect to the required reliability is reported, as shown in Table 3.6. The accuracy of the three methods are good, and their accuracy is almost identical because all of them use FORM to calculate reliability.

The difference between Cases 1 and 2 is that the latter requires higher reliabilities for the two constraints. The results of the three methods are given in Table 3.6 and the convergence history of SORA is given in Table 3.7. The results show that the three method successfully found optimal solutions. SORA is still the most efficient method because it needs the least number of function calls. The accuracy of the three methods is also good.

Table 3.8 Results for Case 2

Method	DL-Directive	DL-Inverse	SORA
b	2.5786	2.5723	2.5608
h	3.9400	3.9496	3.9671
Objective	10.1598	10.1596	10.1589
Function Call	1629	305	212 (165+47)
Error(MCS)	0.32 %	0.32 %	0.32 %

Table 3.9 SORA Convergence History for Case 2

Cycle	Design Variables	g_{mpp}	Objective function
1	(2.047, 3.746)	(-19172.54, -1.79)	7.6681
2	(2.654, 3.805)	(-334.040, 0.369)	10.1003
3	(2.560, 3.967)	(-8×10^{-7}, 0.3136)	10.1589

Observations from Cases 1 and 2 are summarized below.

1. SORA converged with three cycles. For Case 2, the required reliability is much higher, and SORA could still perform well.
2. The number of function calls indicates the efficiency. The inverse reliability method is more efficient than the direct reliability method. SORA is significantly better than the other two methods in terms of efficiency.
3. The accuracy of the three methods were verified by MCS method. They have similar accuracy.

All the input information of Case 3 is the same as that of Case 2, except different distributions of P_x and P_y. In Case 3, of P_x and P_y follow log-normal distributions. The new distributions are given in Table 3.3. The results (Tables 3.8 and 3.9) indicate that SORA is still the best method with respect to efficiency even non-normal distributions are involved. It is noted that SORA converged with four cycles.

Table 3.10 Results for Case 3

Method	DL-Directive	DL-Inverse	SORA
b	2.8873	2.9212	2.8870
h	3.6497	3.6072	3.6507
Objective	10.5378	10.5374	10.5396
Function Call	2322	317	321 (237+84)
Error(MCS)	0.32 %	0.32 %	0.32 %

Table 3.11 SORA Convergence History for Case 3

Cycle	Design Variables	g_{mpp}	Objective function
1	(2.047, 3.746)	(-24295.9, -2.460)	7.6681
2	(3.089, 3.374)	(-734.30, 0.45.3)	10.4255
3	(2.828, 3.724)	(-33.52, -0.4094)	10.5345

Table 3.11 SORA Convergence History for Case 3 (cont.)

4	(2.887, 3.6507)	(0, 0.4469)	10.5396

In Case 4, the distributions of P_x and S are changed. This makes it difficult to meet the reliability requirement. For this case, no feasible solution exists. The new distributions are given in Table 3.4. DL-direct and DL-inverse methods stopped prematurely and no feasible solutions were reported. SORA kept running cycle by cycle, and this is an indication of divergence. Even though this is not a robustness problem for SORA, it is desirable to terminate the SORA cycles due to no feasible solution. Thus SORA software could be improve for this situation.

Table 3.12 Results for Case 4

Method	DL-Directive	DL-Inverse	SORA
b	4.0049	4.2918	4.2919
h	5.8614	6.2918	6.2919
Objective	23.4741	27.0032	27.0047
Function Call	6747	341	736 (630+106)
Error(MCS)	48.97 %	2.03 %	2.03 %

Table 3.13 SORA Convergence History for Case 4

Cycle	Design Variables	g_{mpp}	Objective function
1	(3.7133, 4.6416)	(-9330.7, 1.635)	17.2355
2	(4.2558, 6.2558)	(-1238.97, 2.104)	26.6237
3	(4.2919, 6.2919)	(-1055.98, 2.116)	27.0047
4	(4.2919, 6.2919)	(-1055.98, 2.116)	27.0047
⋮	⋮	⋮	⋮
10	(4.2919, 6.2919)	(-1055.98, 2.116)	27.0047

The distributions of some random variables changed again in Case 5. The new distributions are given in Table 3.5. All the three methods failed to converge to the

optimal solutions. For SORA, this is a robustness issue. As will be discussed in Sec.4, this problem could be fixed by accommodating log-normal distributed strengths correctly.

Table 3.14 Results for Case 5

Method	DL-Directive	DL-Inverse	SORA
b	2.5816	2.0470	0.1000
h	3.9202	3.7491	0.1000
Objective	10.1205	7.6744	0.0100
Function Call	1878	153	314(264+50)
Error(MCS)	0.32%	-50.24%	100 %

Table 3.15 SORA Converge History for Case 5

Cycle	Design Variables	g_{mpp}	Objective function
1	(2.047, 3.746)	(-49.9064, -0.0031)	7.6681
2	(0.1, 0.1)	(-899960049, -1544039)	0.01
⋮	⋮	⋮	⋮
10	(0.1, 0.1)	(-899960049, -1544039)	0.01

The observations and findings from this examples are summarized below.

1. All three methods work well for random variables that are normally distributed.
2. SORA is more efficient than the other two methods.
3. The accuracy of the three methods are almost the same because they all use FORM.

4. When there is no feasible solution because the reliability requirement is too high, SORA does not converge. Divergence should be reported to the user.
5. SORA fails to converge when strength-type random variables are log-normal distributed.

3.2.2. Testing Problem 2. A welded beam is shown in Fig 3.1. There are four independent random variables and five probabilistic constraints. The objective function is the welding cost. And the failure modes are the excessive shear stress, bending stress, buckling, and excessive displacement [24-26].

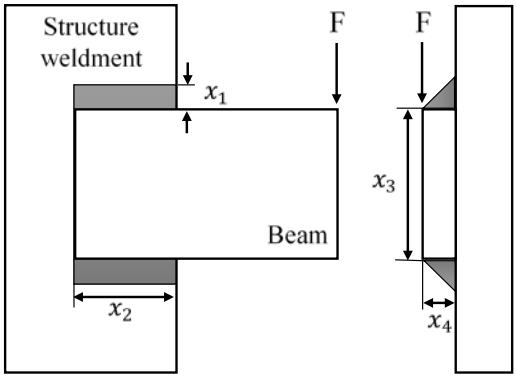

Figure 3.5. A Welded Beam Design Example

The complete RBD model is given by:

$$\text{Minimize: } f(\mathbf{d}, \boldsymbol{\mu}_{\mathbf{X}}) = c_1\mu_{\text{x1}}^2\mu_{\text{x2}} + c_2\mu_{\text{x3}}\mu_{\text{x4}}(p_2 + d_2)$$

Subject to:

$$\Pr\{g_1(\mathbf{d}, \mathbf{X})\} = \Pr\{\tau(\mathbf{X}, \mathbf{P})/p_6 - 1\} \le p_{f1}$$
$$\Pr\{g_2(\mathbf{d}, \mathbf{X})\} = \Pr\{\sigma(\mathbf{X}, \mathbf{P})/p_7 - 1\} \le p_{f2}$$
$$\Pr\{g_3(\mathbf{d}, \mathbf{X})\} = \Pr\{X_1/X_4 - 1\} \le p_{f3}$$
$$\Pr\{g_4(\mathbf{d}, \mathbf{X})\} = \Pr\{\delta(\mathbf{X}, \mathbf{P})/p_5 - 1\} \le p_{f4}$$
$$\Pr\{g_5(\mathbf{d}, \mathbf{X})\} = \Pr\{1 - P_c(\mathbf{X}, \mathbf{P})/p_1\} \le p_{f5}$$

$$
\begin{gathered}
H(\mathbf{X}) = \sqrt{X_2^2 + (X_1 + X_3)^2}/2 \\
J(\boldsymbol{X}) = \sqrt{2}X_1X_2\{X_2^2/12 + (X_1 + X_3)^2/4\} \\
\tau(\mathbf{X},\mathbf{P}) = \sqrt{\{t(\mathbf{X},\mathbf{P})^2 + 2X_2t(\mathbf{X},\mathbf{P})^2tt(\mathbf{X},\mathbf{P})/2H(\mathbf{X}) + tt(\mathbf{X},\mathbf{P})^2\}} \\
t(\mathbf{X},\mathbf{P}) = p_1/\sqrt{2}X_1X_2 \\
tt(\mathbf{X},\mathbf{P}) = M(\mathbf{X},\mathbf{P})H(\boldsymbol{X})/J(\boldsymbol{X}) \\
M(\mathbf{X},\mathbf{P}) = p_1(p_2 + X_2/2) \\
\sigma(\mathbf{X},\mathbf{P}) = 6p_1p_2/X_3^2X_4 \\
\delta(\mathbf{X},\mathbf{P}) = 4p_1p_2^3/p_1X_3^2X_4 \\
P_c(\mathbf{X},\mathbf{P}) = \left(4.013X_3X_4^2\sqrt{p_3p_4}\right)\left(1 - X_3\sqrt{p_3/p_4}/4p_2\right)/6p_2^2
\end{gathered}
\tag{3.4}
$$

The random design variables are X_1, X_2, X_3, and X_4**.** They are given in Table 3.14 and Table 3.15. The system parameters are $p_1, p_2, p_3, p_4, p_5, p_6, p_7, c_1$, and c_2. They are given in Table 3.16. All of the required reliabilities of the constraints are 0.9987.

Table 3.16 Bounds of Design Variables for Example 2

Design Variables	Lower bound	Upper bound
μ_{X1}	3.175 in	50.8 in
μ_{X2}	0 in	280 in
μ_{X3}	0 in	254 in
μ_{X4}	0 in	50.8 in

Table 3.17 Distributions of Random Variables for Example 2

Random Design Variables	Mean	Standard Deviation	Distribution Type
X_1	μ_{X1} in	0.1693 in	Normal
X_2	μ_{X2} in	0.1693 in	Normal
X_3	μ_{X3} in	0.0107 in	Normal
X_4	μ_{X4} in	0.0107 in	Normal

Table 3.18 Design Parameters for Example 2

Other Parameters	Value
p_1	2.6688×10^4 N
p_2	3.556×10^2 mm
p_3	2.0685×10^5 MPa
p_4	8.274×10^4 MPa
p_5	6.35 mm
p_6	9.377×10^1 MPa

Table 3.18 Design Parameters for Example 2 (cont.)

p_7	2.0685×10^2 MPa
c_1	6.74135×10^{-5} mm^3
c_2	2.93585×10^{-6} mm^3

The optimal results from the three methods are given in Table 3.17, which shows that the DL-Direct method failed to converge to a true optimal solutions because constraint 2 was not satisfied. The DL-Inverse and SORA methods produced almost identical solutions. The number of function calls are also listed in Table 3.17. SORA called all functions 754 times, including 696 for deterministic optimization and 58 for reliability analysis. The DL-Direct and DL-Inverse RBD method called functions 4263 and 2052 times, respectively. SORA is therefore the most efficient method and DL-Inverse.

Table 3.19 Results for Testing Example 2

Method	DL-Direct	DL-Inverse	SORA
X_1	8.5029	8.2269	8.2269
X_2	280.0	280.0	280.0
X_3	164.0070	177.0380	177.0380
X_4	9.0966	8.8206	8.8206
Objective	4.1486	4.1915	4.1915
Function Call	4263	2052	754 (696+58)
Error (constraint 1)	0.9 %	0.05 %	0.05 %
Error (constraint 2)	100.00 %	0.03 %	0.03 %
Error (constraint 4)	2.33 %	2.33 %	2.33 %
Error (constraint 5)	2.33 %	2.33 %	2.33 %

The convergence history of SORA is shown in Table 3.20. SORA converged after three cycles. The first cycle involves deterministic optimization with mean values of all the random variables, and it is therefore the conventional optimization. The reliability

requirements are not satisfied if the limit-state function values (g_{MPP} in the table) from reliability analysis are negative. Instead, the reliability of third limit-state function are satisfied due to its positive value. As shown in the third cycle, g_{MPP} of the all the constraints is almost zero. This means that, all constrains are active and that the actual reliability is exactly at the required level. The reliability of the fourth and fifth constraint exceeds the required value because the g_{MPP} value is positive.

Table 3.20 SORA Convergence History for Testing Example 2

Cycle	Design Variables	g_{mpp}	Objective function
1	(8.300726, 252.4269, 178.1419, 8.674441)	(-0.10879 -0.0043564 -0.02537 0.92515 1.3866)	3.9309
2	(8.226855, 280, 177.038, 8.820585)	(-5.3877×10^{-10}, -1.8421×10^{-10}, $1.1446e\times10^{-13}$, 0.92501 1.4981)	4.1915
3	(8.226855, 280, 177.038, 8.820585)	($-5.3877e\times10^{-10}$, -1.8421×10^{-10}, 1.1446×10^{-13}, 0.92501 1.4981)	4.1915

The observations and findings from this examples are summarized below.

1. SORA worked well for this example, which has more reliability constraint functions.
2. DL-Inverse and SORA produced the same accuracy because both of them use FORM.
3. SORA is more efficient than the other two methods.

4. DL-Direct method did not find a feasible solution.

3.2.3. Testing Problem 3. A two dimensional mathematical RBDO problem is defined by

$$\text{Minimize: } f(\boldsymbol{\mu}_{\mathbf{X}}) = -\frac{(\mu_{X1} + \mu_{X2} - 10)^2}{30} - \frac{(\mu_{X1} - \mu_{X2} + 10)^2}{120}$$

Subject to:

$$\Pr\{g_1(\mathbf{X})\} = \Pr\left\{\frac{X_1^2 X_2}{20} - 1 \le 0\right\} \le p_{f1}$$

$$\begin{aligned}\Pr\{g_2(\mathbf{X})\} = \Pr\{&1 - (0.9063X_1 + 0.4226X_2 - 6)^2 \\ &- (0.9063X_1 + 0.4226X_2 - 6)^3 \\ &+ 0.6(0.9063X_1 + 0.4226X_2 - 6)^4 \\ &+ (-0.4226X_1 + 0.9063X_2) \le 0\} \le p_{f2}\end{aligned} \tag{3.5}$$

$$\Pr\{g_3(\mathbf{X})\} = \Pr\left\{\frac{80}{X_1^2 + 8X_2 + 5} - 1 \le 0\right\} \le p_{f3}$$

The design variables are X_1 and X_2 . They are given in Table 3.21. The required reliabilities of the three constraints are all 0.9772, or $p_{fi} = 0.0228, i = 1,2,3$ [27].

Table 3.21 Bounds of Design Variables

Design variables	Lower bound	Upper bound
X_1	0	10
X_2	0	10

Table 3.22 Distributions of Random Variables

Design variables	Mean	Standard Deviation	Distribution Type
X_1	μ_{X1}	0.5	Normal
X_2	μ_{X1}	0.5	Normal

The optimal results from the three methods are given in Table 3.23, which shows that the three methods produce the same solutions. Thus they have the same accuracy.

From this example, SORA is obvious efficient than DL-Direct and DL-Inverse because SORA only called all functions 243 times. DL-Direct called all functions 2166 times and DL-Inverse called all function 785 times. All the three methods are accuracy with the same small errors.

Table 3.23 Results for Example 3

Method	DL-Direct	DL-Inverse	SORA
X_1	4.6717	4.6717	4.6717
X_2	1.5684	1.5684	1.5684
Objective	-1.902	-1.902	-1.902
Function Call	2166	785	243 (180+63)
Error (constraint 1)	0.93 %	0.93 %	0.93 %

The convergence history of SORA is shown in Table 3.24. SORA converged only after 2 cycles. When SORA converged, the first and second constraints are active because their g_{MPP} value are almost zero. The third constraint exceeds the required reliability because the g_{MPP} value is positive.

Table 3.24 SORA Convergence History for Example 3

Cycle	Design Variables	g_{MPP}	Objective function
1	(5.1969, 0.7405)	(-0.8114, -1.0688, 0.7378)	-2.2917
2	(4.6717, 1.5684)	(6.9863×10^{-6}, 4.2335×10^{-6}, 0.7032)	-1.902

3.2.4. Testing Problem 4. This testing problem has ten random design variables and eight probabilistic constraints [26, 28]. The complete RBD model is given by

$$\text{Minimize: } f(\boldsymbol{\mu}_{\mathbf{X}}) = \mu_{X1}^2 + \mu_{X2}^2 + \mu_{X1}\mu_{X2} - 14\mu_{X1} - 16\mu_{X2} + (\mu_{X3} - 10)^2 + 4(\mu_{X4} - 5)^2 + (\mu_{X5} - 3)^2 + 2(\mu_{X6} - 1)^2 + 5\mu_{X7}^2 + 7(\mu_{X8} - 11)^2 + 2(\mu_{X9} - 10) + (\mu_{X10} - 7)^2 + 45$$

Subject to:

$$\Pr\{g_1(\mathbf{X})\} = \Pr\left\{1 - \frac{4X_1 + 5X_2 - 3X_7 + 9X_8}{105} \le 0\right\} \le p_{f1}$$

$$\Pr\{g_2(\mathbf{X})\} = \Pr\{-10X_1 + 8X_2 + 17X_7 - 2X_8 \le 0\} \le p_{f2}$$

$$\Pr\{g_3(\mathbf{X})\} = \Pr\left\{1 - \frac{-8X_1 + 2X_2 - 5X_9 - 2X_{10}}{12} \le 0\right\} \le p_{f3}$$

$$\Pr\{g_4(\mathbf{X})\} = \Pr\left\{1 - \frac{3(X_1 - 2)^2 + 4(X_2 - 3)^2 - 2X_3^2 - 7X_4}{120} \le 0\right\} \le p_{f4} \quad (3.6)$$

$$\Pr\{g_5(\mathbf{X})\} = \Pr\left\{1 - \frac{5X_1^2 + 8X_2 + (X_3 - 6)^2 - 2X_4}{40} \le 0\right\} \le p_{f5}$$

$$\Pr\{g_6(\mathbf{X})\} = \Pr\left\{1 - \frac{0.5(X_1 - 8)^2 + 2(X_2 - 4)^2 + 3X_5^2 - X_6}{120} \le 0\right\} \le p_{f6}$$

$$\Pr\{g_7(\mathbf{X})\} = \Pr\{-X_1 - 2(X_2 - 2)^2 + 2X_1X_2 - 14X_5 - 6X_6 \le 0\} \le p_{f7}$$

$$\Pr\{g_8(\mathbf{d}, \mathbf{X})\} = \Pr\{3X_1 - 6X_2 - 12(X_9 - 8)^2 + 7X_{10} \le 0\} \le p_{f8}$$

The random design variables are $X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_8, X_9$, and X_{10}**.** They all positive numbers and follow normal distributions $X_i \sim N(\mu_{Xi}, 0.02^2)$. All of the required reliabilities of the constraints are 0.9987, or $p_{fi} = 0.0013, i = 1,2, \ldots,8$.

The results of Example 4 are discussed in detail as follows: the optimal results from the three methods are given in Table 3.25, which shows that the three methods produce almost identical solutions. As discussed previously, the computational cost is extremely high by using traditional RBD methods to solve multidimensional and high nonlinear problems, such as this example. The advantages of SORA are more obvious. The Dl-Direct calls all functions 157,840 time, and DL-Inverse calls all functions 11,550. DL-Inverse therefore is more efficient than DL-Direct. But SORA only calls all function 3,066 times, including 2,601for deterministic optimization and 465 for

reliability analysis. Obviously, SORA is the most efficient method among the three methods.

Table 3.25 Results for Example 4

Method	DL-Direct	DL-Inverse	SORA
μ_{X1}	1.7491	1.7483	1.7482
μ_{X2}	2.6414	2.6406	2.6405
μ_{X3}	8.7345	8.7344	8.7344
μ_{X4}	5.0630	5.0630	5.0629
μ_{X5}	1.0186	1.0186	1.0186
μ_{X6}	1.4240	1.4246	1.4246
μ_{X7}	1	1	1
μ_{X8}	9.6790	9.6797	9.6797
μ_{X9}	6.2461	6.2465	6.2465
μ_{X10}	7.1642	7.161	7.1610
Objective	8.5439	8.5458	8.5458
Function Call	157840	11550	3066 (2601+465)
Error (constraint 1)	0.0041 %	0.002 %	0.002 %
Error (constraint 8)	0.0006 %	0.0005 %	0.0005 %

The convergence history of SORA is shown in Table 3.26. After three cycles, SORA converged. The negative g_{MPP} values are indicate the reliability requirements of corresponding limit-state function are not satisfied. When g_{MPP} of a constraint function is close to zero, the corresponding constraint is active. The positive g_{MPP} values indicate that the reliability requirements of corresponding limit-state functions are exceed the required value.

Table 3.26 SORA Convergence History for Example 4

Cycle	Design Variables	g_{mpp}	Objective function
1	(1.613, 2.3776, 8.7616, 5.0756, 1.0156, 1.4078, 1.0022, 9.963, 6.1591, 7.1562)	(-0.0065, -1.2815, 9.5827, -0.0181, 0.2336, 0.7665, -0.9467, -2.7553)	7.1858
2	(1.7483, 2.6406, 8.7344, 5.063, 1.0186, 1.4246, 1, 9.6797, 6.2464, 7.1616)	(1.7426× 10^{-12}, -3.3704× 10^{-11}, 10.4049, -9.2919× 10^{-12}, 0.1255, 0.7870, 5.0597× 10^{-7}, -1.9455× 10^{-6})	8.5458
3	(1.7482, 2.6406, 8.7344, 5.062, 1.0186, 1.4246, 1, 9.6797, 6.2465, 7.161)	(8.2345× 10^{-13}, -8.3013× 10^{-11}, 10.4044, -5.8518× 10^{-9}, 0.1255, 0.787, 4.9527× 10^{-7}, -2.2241× 10^{-6})	8.5458

3.3. IMPPROVE CONVERGENCE ROBUSTNESS

This subsection discusses how to improve the convergence robustness of SORA. As has been shown in Case 5 of Example 1, SORA could not converge when the strength-type random variables follow log-normal distributions. It is found that the divergence is caused by the sign of the limit-state function at the origin in the U-space during the inverse MPP search.

Recall that the probability of failure is defined by $p_f = \Pr(g \leq 0)$. This means that the failure region is away from the original and the distance between the two is the reliability index $\beta = |\mathbf{u}^*|$, where $\mathbf{u}^*$ is the MPP. This also implies that the origin **O** is in the safe region; in other words, $g(\mathbf{0}) > 0$. If this condition is not satisfied, the performance of the inverse MPP search is unpredictable and the process may diverge.

Let X be a strength-type of random variable with its mean μ_X and standard deviation σ_X. The natural logarithm of X is normally distributed; namely, for $Y = \mathrm{Ln}(X)$, $Y \sim N(\mu_Y, \sigma_Y^2)$. At the original of the U-Space, $U = 0$. The transformation between U and X is given by

$$F_X(X) = \Phi(U) \tag{3.7}$$

When $U = 0$,

$$F_X(X) = \Phi(0) = 0.5 \tag{3.8}$$

The transformed X is then

$$X = X_m \tag{3.9}$$

where X_m is the median of X.

It is known that

$$X_m = e^{\mu_Y} \tag{3.10}$$

$$\mu_X = e^{\mu_Y + \sigma_Y^2} \tag{3.11}$$

This gives

$$X_m \leq \mu_X \tag{3.12}$$

As a result, the transformed X is less than the mean of the strength. If σ_Y is large, according to Eqs. (3.10) and (3.11), X will be much smaller than the average strength, and this will lead to a failure. Then the limit-state function at the origin in the U-Space will be negative, or $g \leq 0$. This violates the condition $g > 0$.

This problem can be fixed by setting U to correspond to the mean μ_X. The new transformation is given by

$$\Phi(U) = F_X(\mu_X) \tag{3.13}$$

Then

$$U = \Phi^{-1}(F_X(\mu_X)) \tag{3.14}$$

Since $\mu_X \geq X_m$, $F_X(\mu_X) > 0.5$. This leads to $U > 0$. At U defined in Eq. (3.14), g is positive. Then the convergence problem of SORA is fixed.

With the change in the inverse MPP search, SORA could converge when the strength-type random variables are log-normally distributed. This is demonstrated by the result in Tables 3.27 and 3.28 with the change for Case 5 of Example 1.

Table 3.2 shows that SORA converged to a feasible optimal point as the other two methods did. SORA is also the most efficient method for this case.

Table 3.27 New Results for Case 5

Method	DL-Directive	DL-Inverse	SORA
b	2.5816	2.5759	2.5629
h	3.9202	3.9289	3.9485
Objective	10.1205	10.1204	10.1195
Function Call	1878	305	212 (165+47)
Error(MCS)	00.32 %	00.32 %	00.32 %

Table 3.28 New SORA Convergence History for Case 5

Cycle	Design Variables	g_{mpp}	Objective function
1	(2.047, 3.746)	(-19081.18, -1.7778)	7.6681
2	(2.6519, 3.8023)	(-215.118, 0.37172)	10.0834
3	(2.5629, 3.9485)	(-1.73×10^{-8}, 0.311)	10.1195

Table 3.27 indicates that SORA converged in three cycles. The g_{mpp} values also indicate that the optimal solution is feasible and that the reliability requirements are satisfied.

4. CONCLUSIONS

The objective of this research is to evaluate the Sequential Optimization and Reliability Analysis (SORA). SORA is a methodology for reliability-based design (RBD). RBD usually minimizes a cost-type objective function and also maintains the reliability at the required level. It can therefore reduce the product cost with increased reliability. Due to this advantage, RBD has increasingly used in engineering applications. Compared to deterministic optimization, however, RBD is much more computationally expensive. Thus it is critical for engineers to select an appropriate RBD approach for their specific problems. This needs better understanding of all common RBD methodologies, including SORA. The objective of this research is motivated by such a need. Through the thorough evaluation of SORA, this research offers better understanding of SORA, a better guidance for selecting RBD methodologies, and possible ways for improving RBD.

4.1. SUMMARY OF THE EVALUATION STUDY

RBD is evaluated in this study with respect to efficiency, accuracy, and robustness. The efficiency is measured by the number of limit-state function evaluations, including the function evaluations used by both optimization and reliability loops. The accuracy is evaluated using the Monte Carlo simulation (MCS) solutions as a benchmark. The MCS solutions are regarded as the accurate solutions given a large sample size. The reliabilities of the active constraints at the optimal points produced by SORA are calculated by MCS, and such reliabilities are compared to the required reliabilities. The

differences between the two types of reliabilities are considered as the errors. The robustness is measured by the ability of convergence.

4.2. FINDINGS AND CONCLUSIONS

SORA is more efficiency than Double-Loop RBD with direct reliability analysis and Double-Loop RBD with inverse reliability analysis. The efficiency is measured by the number of function calls, including both objective and constraint functions. This is demonstrated by the results of four testing problems. For example, for testing example 1, SORA called all functions 199 times, including 156 for deterministic optimization and 43 for reliability analysis. The double-loop RBD method with direct reliability (DL-Direct) and double-loop RBD method with inverse reliability analysis called functions 6447 and 301 times, respectively. For testing problem 4, SORA called all functions 3066 times, including 2601 for deterministic optimization and 465 for reliability analysis. The DL-Direct RBD method and DL-Inverse RBD method called functions 157,840 and 11,550 times, respectively.

SORA has the same accuracy as the Double-Loop RBD with direct reliability analysis and Double-Loop RBD with inverse reliability analysis. The season is that all the three methods use the same reliability analysis method, which is the First Order Reliability Method (FORM). AS a result, the accuracy of SORA for reliability depends on the accuracy of FORM. In general, the accuracy is satisfactory. When a limit-state function in the U-space is highly nonlinear, the accuracy will decrease.

The robustness is measured by the ability of convergence. The results show that SORA is robust. SORA does not calculate reliability directly. Instead, SORA employs

the inverse MPP search algorithm, which is more robust. However, SORA may not converge when strength-type random variables are log-normal distributed. This robustness issue is fixed by accommodating log-normal distributed strengths correctly. After the modification, SORA converges when log-normally distributed strength-type variables are involved. It can be concluded that the robustness of SORA depends on the robustness of the deterministic optimization and the MPP search. If both could converge to an optimal solution and an MPP, respectively, then SORA would converge to a feasible optimal solution.

4.3. FUTURE WORK

As evaluated by this study, SORA is efficient for reliability-based design. It can still be further evaluated and further improved in the future work. Possible future research directions are listed below.

4.3.1. Perform Further Evaluations. Large-scale problems could be used for the evaluation. For examples, the number of random variables and number of reliability constraints could be higher than what has been used in this study. Real CAE simulation models can also be used for the evaluation. For instant, a limit-state function may involve the stress in a mechanical component and finite element analysis is used to calculate the stress. This study used normal and log-normal distributions. More distributions could also be included for the evaluation.

4.3.2. Use More Efficient and Robust MPP Search Algorithms. The efficiency and robustness of SORA can be further improved. As discussed previously, SORA consists of both optimization and reliability analysis (the MPP search). As a result,

improving the efficiency and robustness of the MPP search will improve the efficiency and robustness of SORA. One way is to use more efficient and robust MPP algorithms that are developed recently and are available to use. The other way is to develop more efficient and robust MPP algorithms. In some cases, improving the robustness may increase the number of function calls, thereby decreasing the efficiency. It is therefore important to find a good balance between robustness and efficiency.

4.3.3. Use More Efficient and Robust Optimization Algorithms. The other way to improve the efficiency and robustness is to use more efficient and robust optimization algorithms. There are numerous optimization algorithms available. In this evaluation study, the active-set optimization algorithm within the Matlab function fmincon was used. It was better than other algorithms available in fmincon for the testing problems. This indicates that the importance of optimization algorithms.

4.3.4. Use the Second Order Reliability Method (SORM). Since SORA uses FORM for the reliability analysis, its accuracy for reliability is the same of FORM. FORM linearizes a limit-state function at the MPP. Due to the linearization, an error is unavoidable for a nonlinear limit-state function. Although the accuracy of SORA in general is acceptable, for important applications, higher accuracy is required. It is well known that SORM is in general more accurate than FORM. For this reason, SORM might be used. But the challenge is that the reliability estimated by SORM is not directly linked to the MPP as FORM does. The future research direction will be to find an equivalent MPP that corresponds to the reliability obtained by SORM. Then the equivalent MPP can be used to reformulate a reliability constraint function.

APPENDIX

Matlab Code for Strenth-Type variables with log-normal distribution

```
function
[reliability,gmpp,funEvaluation,sign,g0,xmpp,umpp,exitflag]=...

relia_eval_opt(x,ncr,nd,nx,np,nr,d,model,numran,disttype,distpara,...
    randx,betaopt,umppold,eps)
% -------------------------------------------------------------------
% Find percentile value during optimization
% exitfalg = -1, the limit-state function has negtive sign at the
origin at U space
% exitfalg = 1, the limit-state function has positive (or zero) sign at
the origin at U space
%
reliability=[];
funEvaluation=[];
xmpp=zeros(ncr,nr);
if nx~=0
   distpara(1:nx,1)=x(nd+1:nd+nx)';
end

% Calculate limit-state function at origin
u0=zeros(1,nr);
% For lognormal, added on 03/11/2016
% -----------------------------------------------
for i = 1:nr
    if disttype(i) == 5
        b=(log((distpara(i,2)/distpara(i,1))^2+1))^0.5;
        a=log(distpara(i,1))-0.5*b^2;
        cdf_logn=logncdf(distpara(i,1),a,b);
        u0(i) = norminv(cdf_logn);
    end
end
% -----------------------------------------------
index=0;
g0=gatu(model,nx,nr,u0,disttype,distpara,d,numran,randx,index);
for i=1:ncr
   if g0(i)>=0
      sign(i)=1;
   else
      sign(i)=-1;
   end
end

step(1:length(u0))=0.001;
if abs(max(umppold))==0

dgdu0=dire(model,nx,nr,ncr,u0,g0,disttype,distpara,numran,randx,d,index
,step);
   for i=1:ncr
      dgdu0(i,:)=sign(i)*dgdu0(i,:);
   end
else
   dgdu0=zeros(ncr,nr);
end
```

```
for i=1:ncr
   clear dgdui umppoldi u minusgrad umppi xmppi;
   for j=1:numran(i)
      dgdui(j)=dgdu0(i,randx(i,j));
      umppoldi(j)=umppold(i,j);
   end

   if sign(i)==-1
      exitflag(i)=-1;
      gmpp(i)=g0(i);
      xmppi=zeros(1,numran(i));
      umppi=zeros(1,numran(i));
   end
   if sign(i)==1
      if norm(umppoldi)~=0 | norm(dgdui)==0
         u=umppoldi;
      else
         minusgrad=-dgdui/norm(dgdui);
         u=betaopt(i)*minusgrad;
      end
      index=i;

[umppi,xmppi,gmpp(i)]=mppbeta(model,u,nx,nr,ncr,disttype,distpara,d,num
ran,betaopt(i),sign(i),randx,index,eps);
      exitflag(i)=1;
   end
   xmpp(i,1:numran(i))=xmppi(1:numran(i));
   umpp(i,1:numran(i))=umppi(1:numran(i));
end
```

Matlab Code of Main Testing Example 1

```
function [z0,nc,ncr,ncd,nd,nx,np,nr,model,zl,zu,numran,disttype,...
    distpara,randx,d,betaopt,eps] ...
    = exp2_in(z)
ncr = 2;              % Number of reliability constraints
nc = 2;               % Number of constraints
ncd = 0;              % Number of deterministic constraints
nd = 2;               % Number of deterministic design variables
nx = 0;               % Number of random design variables
np = 4;               % Number of random parameters
nr = 4;               % Number of random variables

model = 'exp2';
distpara= [ 500,100,0,0; % Distribution parameters
           1000,100,0,0;
           29e6,1.45e6,0,0;
           4e4,2e3,0,0];
zl= [0.1,0.1];
zu= [4,6];
w = 2;
t = 4;
d = [w,t];
z0 = [w,t];          % Initial design variables

randx =zeros(2,4);
randx(1,1:3) = [1,2,4];
randx(2,1:3) = [1,2,3];
numran = [3,3,0];
eps=2.5;

case_exp =1;
switch case_exp
    case 1                              % Success
        disttype(1:4) = 1;       % All nornal
        betaopt(1:nc) = 3;      % beta for required reliability
        distpara= [ 500,100,0,0; % Distribution parameters
           1000,100,0,0;
           29e6,1.45e6,0,0;
           4e4,2e3,0,0];
     case 2                             % Success
        disttype(1:4) = 1;       % All nornal
        betaopt(1:nc) = 4;      % beta for required reliability
        distpara= [ 500,100,0,0; % Distribution parameters
           1000,100,0,0;
           29e6,1.45e6,0,0;
           4e4,2e3,0,0];
     case 3                             % Success
        disttype(1:2) = 5;        % Lognormal
        disttype(3:4) = 1;
        betaopt(1:nc) = 4;   % beta for required reliability
        distpara= [ 500,100,0,0; % Distribution parameters
           1000,100,0,0;
           29e6,1.45e6,0,0;
           4e4,2e3,0,0];
     case 4                             % Failure, no feasible solution
```

```
        disttype(1:2) = 1;        % Lognormal
        disttype(3:4) = 1;
        betaopt(1:nc) = 4;   % beta for required reliability
        distpara= [ 800,100,0,0; % Distribution parameters
           1000,100,0,0;
           29e6,1.45e6,0,0;
           1.5e4,2e3,0,0];

     case 5                           % Failure, SORA problem
        disttype(1:2) = 1;        % Lognormal
        disttype(3:4) = 5;
        betaopt(1:nc) = 4;   % beta for required reliability
        distpara= [ 500,100,0,0; % Distribution parameters
           1000,100,0,0;
           29e6,1.45e6,0,0;
           4e4,2e3,0,0];
 end


function g = exp2(d,x,p)
% Onjective and constraint functions
% g < 0 -> failure
global funcall
funcall = funcall+1;

[m,~] = size(d);
[n,~] = size(p);
% 2 deterministic design variables
w = d(1:m,1);
t = d(1:m,2);
% 4 random parameters
X = p(1:n,1);
Y = p(1:n,2);
E = p(1:n,3);
R = p(1:n,4);

d0 = 2.5;
L = 100;
% 3 reliability constraints
g(1,:) = R - (600*Y./w./t.^2 + 600*X./w.^2./t);
g(2,:) = d0 - 4*L^3./E./w./t.*((Y./t.^2).^2+(X./w.^2).^2).^0.5;
g(3,:) = w.*t;
```

Matlab Code of Main Testing Example 2

```
function [z0,nc,ncr,ncd,nd,nx,np,nr,model,zl,zu,numran,disttype,...
    distpara,randx,d,betaopt,eps] = rexp_1_in(z)

ncr= 5;     %number of reliability constraints
ncd= 0;     %number of deterministic constraints
nc = 5;     %number of constraints
nd = 0;     %number of deterministic design variables
nx = 4;     %number of random design variables
np = 0;     %number of random parameters
nr=nx+np;   %number of random variables

model='rexp_1';

randx=zeros(5,4);
numran(1)=4;
randx(1,1:4)=[1,2,3,4];
numran(2)=2;
randx(2,1:2)=[3,4];
numran(3)=2;
randx(3,1:2)=[1,4];
numran(4)=2;
randx(4,1:2)=[3,4];
numran(5)=2;
randx(5,1:2)=[3,4];
numran(6)=4;
randx(6,1:4)=[1,2,3,4];

zl=[3.175,0,0,0];
zu=[50.8,280,280,50.8];

z0=[5,200,210,6];

disttype(1:4)=1;
distpara=[0,0.1693,0,0;
    0,0.1693,0,0;
    0,0.0107,0,0;
    0,0.0107,0,0];
betaopt(1:nc) = 3.5;
d=0;
eps=2.5;

function g = rexp_1(d,x,p)
global funcall
funcall = funcall+1;
[m,~]=size(x);
[n,~]=size(p);

z1=2.6688e4;
z2=3.556e2;
z3=2.0685e5;
z4=8.274e4;
z5=6.35;
z6=9.377e1;
```

```
z7=2.0685e2;
c1=6.74135e-5;
c2=2.93585e-6;
% 4 random design variables
x1=x(1:m,1);
x2=x(1:m,2);
x3=x(1:m,3);
x4=x(1:m,4);

t_x=z1./(sqrt(2)*x1.*x2);
m_x=z1*z2+z1*x2/2;
r_x=sqrt(x2.^2+(x1+x3).^2)/2;
j_x=sqrt(2)*x1.*x2.*(x2.^2/12+((x1+x3).^2)/4);
sigma_x=6*z1*z2./(x3.^2.*x4);
theta_x=4*z1*z2^3./(z3*x3.^3.*x4);
p_x=4.013*x3.*x4.^3*sqrt(z3*z4)/(6*z2^2).*(1-x3/4/z2*sqrt(z3/z4));

tt_x=m_x.*r_x./j_x;
tao_x=sqrt(t_x.^2+2*t_x.^2.*tt_x.*x2/2./r_x+tt_x.^2);
% 5 reliability constraints
g1=1-tao_x./z6;
g2=1-sigma_x./z7;
g3=1-x1./x4;
g4=1-theta_x./z5;
g5=p_x./z1-1;
% objective function
f=c1*mean(x1).^2.*mean(x2)+c2*mean(x3).*mean(x4).*(z2+mean(x2));

g(1,:)=g1;
g(2,:)=g2;
g(3,:)=g3;
g(4,:)=g4;
g(5,:)=g5;
g(6,:)=f;
```

Matlab Code of Main Testing Example 3

```
function [z0,nc,ncr,ncd,nd,nx,np,nr,model,zl,zu,numran,disttype,...
    distpara,randx,d,betaopt,eps]=rexp_5_in(z)

% enriched performance measure approach for reliabiity-based design
optimization
ncr=3;              %number of reliability constraints
ncd=0;              %number of deterministic constraints
nc=ncr+ncd;         %number of constraints
nd=0;               %number of deterministic design variables
nx=2;               %number of random design variables
np=0;               %number of random parameters
nr=2;               %number of random variables

model='rexp_5';

randx=zeros(3,2);
numran(1)=2;
randx(1,1:2)=[1,2];
numran(2)=2;
randx(2,1:2)=[1,2];
numran(3)=2;
randx(3,1:2)=[1,2];
numran(4)=2;
randx(4,1:2)=[1,2];

disttype(1:2)=1;

zl=[0,0];
zu=[10,10];
distpara=[0,0.3,0,0;
          0,0.3,0,0];
d=0;
eps=0.5;
z0=[5,5];
betaopt(1:nc)=2;

function g=rexp_5(d,x,p)
global funcall
funcall = funcall+1;
[m,~] = size(x);
x1=x(1:m,1);
x2=x(1:m,2);
% 3 reliability constraint
Y=0.9063*x1+0.4226*x2;
Z=-0.4226*x1+0.9063*x2;
```

```
g1=x1.^2.*x2/20-1;
g2=1-(Y-6).^2-(Y-6).^3+0.6*(Y-6).^4+Z;
g3=80/(x1.^2+8*x2+5)-1;
% objective function
f=-(x1+x2-10).^2/30-(x1-x2+10).^2/120;
g(1,:)=g1;
g(2,:)=g2;
g(3,:)=g3;
g(4,:)=f;
```

Matlab Code of Main Testing Example 4

```
function [z0,nc,ncr,ncd,nd,nx,np,nr,model,zl,zu,numran,disttype,...
    distpara,randx,d,betaopt,eps]=rexp_12_in(z)

% enriched performance measure approach for reliabiity-based design
optimization
ncr=8;              %number of reliability constraints
ncd=0;              %number of deterministic constraints
nc=8;               %number of constraints
nd=0;               %number of deterministic design variables
nx=10;              %number of random design variables
np=0;               %number of random parameters
nr=10;              %number of random variables

model='rexp_12';

randx=zeros(8,10);
numran(1)=4;
randx(1,1:4)=[1,2,7,8];
numran(2)=4;
randx(2,1:4)=[1,2,7,8];
numran(3)=4;
randx(3,1:4)=[1,2,9,10];
numran(4)=4;
randx(4,1:4)=[1,2,3,4];
numran(5)=4;
randx(5,1:4)=[1,2,3,4];
numran(6)=4;
randx(6,1:4)=[1,2,5,6];
numran(7)=4;
randx(7,1:4)=[1,2,5,6];
numran(8)=4;
randx(8,1:4)=[1,2,9,10];
numran(9)=10;
randx(9,1:10)=[1,2,3,4,5,6,7,8,9,10];
disttype(1:10)=1;

zl=[1,1,1,1,1,1,1,1,1,1];
zu=[10,10,10,10,10,10,10,10,10,10];
z0=[5,5,5,5,5,5,5,5,5,5];
distpara=[0,0.02,0,0;
          0,0.02,0,0;
          0,0.02,0,0;
          0,0.02,0,0;
          0,0.02,0,0;
          0,0.02,0,0;
          0,0.02,0,0;
          0,0.02,0,0;
          0,0.02,0,0;
          0,0.02,0,0;];
d=0;
eps=0.1;

betaopt(1:nc)=3;
```

```
function g=rexp_12(d,x,p)
global funcall
funcall = funcall+1;

[m,~] = size(x);

x1=x(1:m,1);
x2=x(1:m,2);
x3=x(1:m,3);
x4=x(1:m,4);
x5=x(1:m,5);
x6=x(1:m,6);
x7=x(1:m,7);
x8=x(1:m,8);
x9=x(1:m,9);
x10=x(1:m,10);
% 8 reliability constraints
g1=1-(4*x1+5*x2-3*x7+9*x8)/105;
g2=-10*x1+8*x2+17*x7-2*x8;
g3=1-(-80*x1+2*x2+5*x9-2*x10)/12;
g4=1-(3*(x1-2).^2+4*(x2-3).^2+2*x3.^2-7*x4)/120;
g5=1-(5*x1.^2+8*x2+(x3-6).^2-2*x4)/40;
g6=1-(0.5*(x1-8).^2+2*(x2-4).^2+3*x5.^2-x6)/120;
g7=-(x1+2*(x2-2).^2-2*x1.*x2+14*x5-6*x6);
g8=-(-3*x1+6*x2+12*(x9-8).^2-7*x10);
% objective function
f=x1.^2+x2.^2+x1.*x2-14*x1-16*x2+(x3-10).^2+...
4*(x4-5).^2+(x5-3).^2+2*(x6-1).^2+5*x7.^2+...
7*(x8-11).^2+2*(x9-10)+(x10-7).^2+45;
g(1,:)=g1;
g(2,:)=g2;
g(3,:)=g3;
g(4,:)=g4;
g(5,:)=g5;
g(6,:)=g6;
g(7,:)=g7;
g(8,:)=g8;
g(9,:)=f;
```

BIBLIOGRAPHY

1. Ang, A.H.-S. and W.H. Tang, ***Probability concepts in engineering planning and design*.** 1984.

2. Mahadevan, S. and A. Haldar***, Probability, reliability and statistical method in engineering design*.** 2000: John Wiley & Sons.

3. Ayyub, B. and M.M. Gupta, ***Uncertainty analysis in engineering and sciences: fuzzy logic, statistics, and neural network approach*.** Vol. 11. 2012: Springer Science & Business Media.

4. Tu, J., K.K. Choi, and Y.H. Park, ***A new study on reliability-based design optimization.*** Journal of mechanical design, 1999. 121(4): p. 557-564.

5. Du, X. and W. Chen, ***Sequential optimization and reliability assessment method for efficient probabilistic design.*** Journal of Mechanical Design, 2004. 126(2): p. 225-233**.**

6. Liang, J., Z.P. Mourelatos, and J. Tu. ***A single-loop method for reliability-based design optimization*.** *ASME 2004 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. 2004. American Society of Mechanical Engineers.

7. Chen, W., R. Jin, and A. Sudjianto, ***Analytical variance-based global sensitivity analysis in simulation-based design under uncertainty.*** Journal of mechanical design, 2005. 127(5): p. 875-886.

8. Cheng, G., L. Xu, and L. Jiang, ***A sequential approximate programming strategy for reliability-based structural optimization.*** Computers & structures, 2006. 84(21): p. 1353-1367.

9. Saitou, K., et al., ***A survey of structural optimization in mechanical product development.*** Journal of Computing and Information Science in Engineering, 2005. 5(3): p. 214-226.

10. Rao, S.S., ***Reliability-based design*.** 1992: McGraw-Hill Companies.

11. Youn, B.D., K.K. Choi, and Y.H. Park, ***Hybrid analysis method for reliability-based design optimization.*** Journal of Mechanical Design, 2003. 125(2): p. 221-232.

12. Arora, J., ***Introduction to optimum design*.** 2004: Academic Press.

13. Du, X., J. Guo, and H. Beeram, ***Sequential optimization and reliability assessment for multidisciplinary systems design.*** Structural and Multidisciplinary Optimization, 2008. 35(2): p. 117-130.

14. Bucher, C. and U. Bourgund, ***A fast and efficient response surface approach for structural reliability problems.*** **Structural safety,** 1990. 7(1): p. 57-66.

15. Dolinski, K., ***First-order second-moment approximation in reliability of structural systems: critical review and alternative approach.*** Structural Safety, 1983. 1(3): p. 211-231.

16. Du, X. and W. Chen, ***A most probable point-based method for efficient uncertainty analysis.*** Journal of Design and Manufacturing Automation, 2001. 4(1): p. 47-66.

17. Du, X., A. Sudjianto, and W. Chen, ***An integrated framework for optimization under uncertainty using inverse reliability strategy.*** Journal of Mechanical Design, 2004. 126(4): p. 562-570.

18. Hastings, W.K., ***Monte Carlo sampling methods using Markov chains and their applications.*** Biometrika, 1970. 57(1): p. 97-109.

19. Davidson, J.M., ***A Distributed Surrogate Methodology for Inverse Most Probable Point Searches in Reliability Based Design Optimization*****.** 2015, Wright State University.

20. Du, X., ***Saddlepoint approximation for sequential optimization and reliability analysis.*** Journal of Mechanical Design, 2008. 130(1): p. 011011.

21. Du, X., W. Chen, and Y. Wang, ***Most Probable Point-Based Methods,*** *Extreme Statistics in Nanoscale Memory Design*. 2010, Springer. p. 179-202.

22. Wu, Y.-T., et al. ***Safety-factor based approach for probability-based design optimization*****.** in *AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference and Exhibit, 42 nd, Seattle, WA*. 2001.

23. Mourelatos, Z.P. and J. Zhou, ***A design optimization method using evidence theory.*** Journal of mechanical design, 2006. 128(4): p. 901-908.

24. Cho, T.M. and B.C. Lee, ***Reliability-based design optimization using convex linearization and sequential optimization and reliability assessment method.*** Structural safety, 2011. 33(1): p. 42-50.

25. Lee, J.J. and B.C. Lee, ***Efficient evaluation of probabilistic constraints using an envelope function.*** Engineering Optimization, 2005. 37(2): p. 185-200.

26. Hyeon Ju, B. and B. Chai Lee, ***Reliability-based design optimization using a moment method and a kriging metamodel.*** Engineering Optimization, 2008. 40(5): p. 421-438.

27**.** Lee, I., et al., ***Inverse analysis method using MPP-based dimension reduction for reliability-based design optimization of nonlinear and multi-dimensional systems.*** Computer Methods in Applied Mechanics and Engineering, 2008. 198(1): p. 14-27.

28**.** Chen, Z., et al.**,** ***An adaptive decoupling approach for reliability-based design optimization.*** Computers & Structures, 2013. 117: p. 58-66.

VITA

Guannan Liu was born on March 20, 1990 in Chongqing, China. He received both his primary and secondary education in Nanping, Chongqing. He studied at China University of Petroleum (Huadong) and Missouri University of Science and Technology where he received his Bachelor of Science degree in Geology and Geosciences in May of 2014. While in school he joined the Science Club and participated in geosciences projects in Huadong. During May to August 2014, he worked as a Project Management Intern in a Petroleum company, China National Petroleum Corporation (CNPC-BGP), Houston, TX. He then started his study at the Missouri University of Science and Technology for his Master's degree in Mechanical Engineering. In May 2016, he received his Master's degree in Mechanical Engineering.