RELIABILITY-BASED STRUCTURAL DESIGN: A CASE OF AIRCRAFT FLOOR GRID LAYOUT OPTIMIZATION

A Thesis
Presented to
The Academic Faculty

by

Qing Chen

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Mechanical Engineering in the
George W. Woodruff School of Mechanical Engineering

Georgia Institute of Technology
May 2010

RELIABILITY-BASED STRUCTURAL DESIGN: A CASE OF AIRCRAFT FLOOR GRID LAYOUT OPTIMIZATION

Approved by:

Dr. Roger Jiao, Advisor
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Seung-Kyum Choi
School of Mechanical Engineering
Georgia Institute of Technology

Dr. Dirk Schaefer
School of Mechanical Engineering
Georgia Institute of Technology

Date Approved: November 22, 2010

ACKNOWLEDGMENTS

First I would like to thank my advisor, Dr. Roger Jiao, from whom I learned the methodology of scientific research; without whom I could not have written this thesis. I would like to thank the other members of my thesis reading committee, Dr. Seung-Kyum Choi and Dr. Dirk Schaefer, for taking time reading my thesis. I would also like to thank Boeing in Charleston, S.C. for providing the opportunity of case study.

I would like to thank students in the System Realization Laboratory. I would like to thank my roommate Andrew McMorrow for sharing the happy hours when we were living in GLC. I would like to thank Lam Duong for studying, doing projects and discussing together with me. I would like to thank Stephanie Thompson and Ben Lee for helping me when I first came to SRL. I would like especially to thank Ms. Lei Zhu for teaching me driving during 2009 winter break when I was in need of help. I would also like to thank Chenjie Wang, Jiten Patel, Andrew Hyder, Xiufeng Yang, Chuanfeng Wang, Mukul Singhee, Thomas Stone and some other students for their friendship. My thanks also go to Kiki Lewis, who has always been by my side every night when I come home.

I would like to thank Georgia Tech for providing financial support. I would like also to thank those undergraduate students in Savannah campus who were involved in and contributed to this research.

Last but not least I would like to thank my parents, my sister and her husband, without their understanding and support I could not have made it this far.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

ρ	= Pearson's correlation coefficient
$K(d)$	= Covariance function
$\mathbf{C};\mathbf{K}$	= Covariance matrix
$\mathbf{k}$	= Covariance vector between sampling points and estimating point
λ_α	= Kriging weight
$f(\mathbf{X})$	= Objective function
$\mathbf{h}(\mathbf{X})$	= Equality constraints
$\mathbf{g}(\mathbf{X})$	= Inequality constraints
∇f	= Gradient of function f()
$\mathbf{H}$	= Hessian matrix of objective function
$L(\mathbf{x},\boldsymbol{\lambda})$	= Lagrangian function
P_f	= Probability of failure
β	= reliability index
$\overline{\mathbf{X}}$	= Mean value point
$\mathbf{X}^*$	= MPP point
$g(\mathbf{X});g(\mathbf{U})$	= Limit state function
$\hat{g}(\mathbf{U})$	= First order approximation of limit state function at MPP point
$C_{XY}(u,v)$	= Copula CDF function between two random variables X and Y
$c_{XY}(u,v)$	= Copula PDF function between two random variables X and Y
θ	= Coefficient of Archimedean copula

SUMMARY

In this thesis, several Reliability-based Design Optimization (RBDO) methods and algorithms for airplane floor grid layout optimization are proposed. A general RBDO process is proposed and validated by an example. Copula as a mathematical method to model random variable correlations is introduced to discover the correlations between random variables and to be applied in producing correlated data samples for Monte Carlo simulations. Based on Hasofer-Lind (HL) method, a correlated HL method is proposed to evaluate a reliability index under correlation. As an alternative method for computing a reliability index, the reliability index is interpreted as an optimization problem and two nonlinear programming algorithms are introduced to evaluate reliability index. To evaluate the reliability index by Monte Carlo simulation in a time efficient way, a kriging-based surrogate model is proposed and compared to the original model in terms of computing time. Since in RBDO optimization models the reliability constraint obtained by MCS does not have an analytical form, a kriging-based response surface is built. Kriging-based response surface models are usually segment functions that do not have a uniform expression over the design space; however, most optimization algorithms require a uniform expression for constraints. To solve this problem, a heuristic gradient-based direct searching algorithm is proposed. These methods and algorithms, together with the RBDO general process, are applied to the layout optimization of aircraft floor grid structural design.

CHAPTER 1: INTRODUCTION

1.1 Motivation and Significance

In modern structural design, it is always required to assess the uncertainties in material, load, and environment factors of complex structures. For structure models in which randomness is relatively small, it is usually satisfactory to model it by a deterministic method in which the uncertainties are represented by a safety-factor. In the real world, most of the design variables as well as external loads, material properties and environment data usually follow some probabilistic distributions and can be regarded as random variables. The combinations of these random variables determine the safety of the structure. Safety-factor based design is a coarse approximation of the reality. In some cases it leads to design failure, in most cases it results in an over expensive design, i.e., high cost and high weight.

To overcome the drawbacks of safety-factor-based structural design, a set of reliability-based structural design methods has been developed in recent years. Reliability is defined as the ability of a system or component to perform its required functions under stated conditions for a specified period of time. In structural engineering, reliability is usually quantified by the failure rate, given a large set of random samples. Reliability is essentially important for complex systems like warships, space shuttles and aircrafts. The disasters of space shuttle Challenger and Columbia reminds people of the importance of system reliability.

Modern aircraft must undergo severe conditions such as atmospheric pressure differences and temperature differences between the inside and outside of the plane, or heavy structural loads applied upon vehicle components. During aircraft design, the designer needs to consider several factors to satisfy both the static strength requirement, the fatigue requirement and damage tolerance requirement as well. The result

Figure 1-1 Space shuttle Challenger and Columbia tragedy (picture from internet)

of the process is an optimized design with respect to weight, costs and aircraft performance. The purpose of reliability-based structural design for aircraft is to guarantee the reliability of the aircraft structure when the aircraft is operating under uncertainties coming from the material, external load, and environment.

Figure 1-2 Complete structural failure of a 737 in Hawaii in 1988 (picture from internet)

1.2 Problem Description

This thesis mainly solves these problems:

1. Although there are many existing methods for reliability evaluation and optimization, such as First Order Reliability Method (FORM), Second Order Reliability Method (SORM), Monte Carlo Simulation (MCS), Non-Linear Programming (NLP), Copula, Kriging, Artificial Neural Network (ANN), etc. These methods and algorithms are usually focused on details of a specific problem. However in engineering, a general process that incorporates various detailed methods and algorithms is required. Detailed methods and algorithms are usually more "academic", however the general process is more "engineering" concerned. The purpose of the general process is to provide guidance for structure designers from random variable correlation discovery to the finding of the optimal design point.

2. Numerical simulations are usually required for reliability-based design and optimization. To achieve reliable results, it is usually required for the sample set to be big (e.g., 10e6 samples). This is a problem for mathematical models that are computationally intensive since the simulation may take a long time. For structural problems based on Finite Element Analysis (FEA) models, this is especially a problem. The problem is, how to reduce the computational expenses of numerical simulation?

3. Correlation between random variables is not uncommon in reliability-based design optimization problems. Karhunen-Loeve (KL) expansion can be used to represent random variable correlations. Copula as another method to study random variable correlations is now gaining popularity in academia. The problem is, how to use copula in reliability-based structural optimization? What are the advantage and disadvantages of using copula in RBDO?

4. RBDO requires finding an optimal design point in the design space that satisfies the equality and inequality constraints. The challenge is, when the reliability constraint is represented by a response surface model that does not have a uniform form over the design space (or, if the response surface is fragment function), how the optimal design point can be found?

1.3 Research Objective and Scope

Reliability-based structural optimization fulfills the objective function (usually minimizing the structural weight) with the constraint of reliability indices. This reduces the overall cost and weight of the structural while guaranteeing the structural reliability. There are several established structure reliability evaluation methods such as Mean Value First Order Second Moment (MVFOSM) method, Hasofer-Lind (HL) method, Second Order Reliability Method (SORM), etc. However in engineering practice, some practical factors that may impact the reliability evaluation precision and some factors that may improve computation efficiency need to be considered. In the optimization process, some algorithms are proposed and verified.

1. Copula for Modeling Random Variable Correlations

Copula [Nelsen,R.B., 1999] has been introduced to model random variable correlations. In engineering problems, it's not unusual that some random variables have a certain relationship. For example, when one load is big, another one also tends to be big. Copula can be applied to model such relationships. The concept of copula has been introduced by some examples in this thesis. The application of copula in correlated random variable generation has been discussed. With several different types of Archimedean families of copula as examples, this thesis explains why copulas with the same correlation coefficient produce different simulation results. Comparison has been made between Karhunen-Loeve (KL) expansion and copula via a ten-bar-truss example.

2. Kriging for Surrogate Model

Reliability index β is a measure of structural reliability; it is a function of random variable distribution and correlation. Details about reliability index β can be found in [Choi,S.K., 2006]. Most methods for computing reliability index β are iterative numerical methods which are usually computationally intensive. For Monte Carlo Simulation (MCS) of a large number of samples, the computation time usually becomes unaffordable. To deal with this problem, a kriging-based surrogate model is studied to replace the original computationally expensive model. The kriging-based surrogate model can achieve high computational speed while still keeping high precision when sample points are chosen appropriately. A comparison has been made for computation speed between different methods, including those based on original models and those based on surrogate model.

3. Heuristic Gradient Based Searching Algorithm

Reliability-based structural design eventually gives an optimization problem with a constraint being structural reliability index requirement or threshold. The reliability constraints obtained by MCS are usually discrete numerical points without an analytical form. For an optimization problem having all inequality constraints with analytical form, Karush-Kuhn-Tucker (KKT) condition can be applied to compute the optimal point. However if the constraints have no analytical form, KKT condition cannot be applied. To solve this problem, a response surface model is proposed. However for response surface models obtained by many interpolative methods, the response surfaces are usually segment functions without a uniform expression, which poses a problem for most optimization algorithms. To solve this problem, a heuristic gradient based numerical searching method is proposed. On each searching step, the constraints will be checked for activation. If a constraint is activated during a step, gradient projection method will be applied to find the constraint optimum of that step. This process continues until it converges on the optimal point. This thesis proposed this method and validated it by a Reliability-based Design Optimization (RBDO) problem.

4. Correlated HL Method

In structural reliability studies, the original HL method does not consider the correlations between random variables. However, many engineering problems observe the existence of correlations between random variables. This thesis studied the correlations between random variables and proposed a correlated HL method. The correlated HL method is a modified HL method that considers the correlation between two random variables. A simulation has been made to validate this method.

5. Evaluating Reliability Index by Non-Linear-Programming (NLP)

Reliability index β can be computed by MVFOSM, HL and SORM methods. Some methods are imprecise, and some methods are non-intuitive and it is difficult to interpret the physical meaning. When the process of computing β is viewed as an optimization problem, nonlinear optimization method can be applied to solve β with precision and with intuitive geometric meanings. A ten-bar-truss example has been studied to compute β by FORM, HL, SORM and nonlinear optimization method. This example shows that the nonlinear optimization method can achieve high precision.

6. Modeling a RBDO Problem as Bilevel Optimization Problem

In many practical optimization problems it's not unusual to find multi-objective problems with different priorities for different objectives. One of such examples is the famous Stackelberg problem. For optimization problems with two optimization objectives, one of the objectives can be selected as leader objective and another can be selected as follower objective. The leader objective has higher priority over the follower objective in the optimization process. In this thesis, the airplane floor grid reliability-based optimization problem will be studied. The problem will be modeled as a bilevel optimization problem. The reliability requirements can be modeled as inequality constraints that do not have analytical form. To solve the bilevel optimization problem, the response surface method based on regression can be used to replace the reliability constraints.

1.4 Outline of This Thesis

Organization of this thesis is illustrated in Figure 1-3:

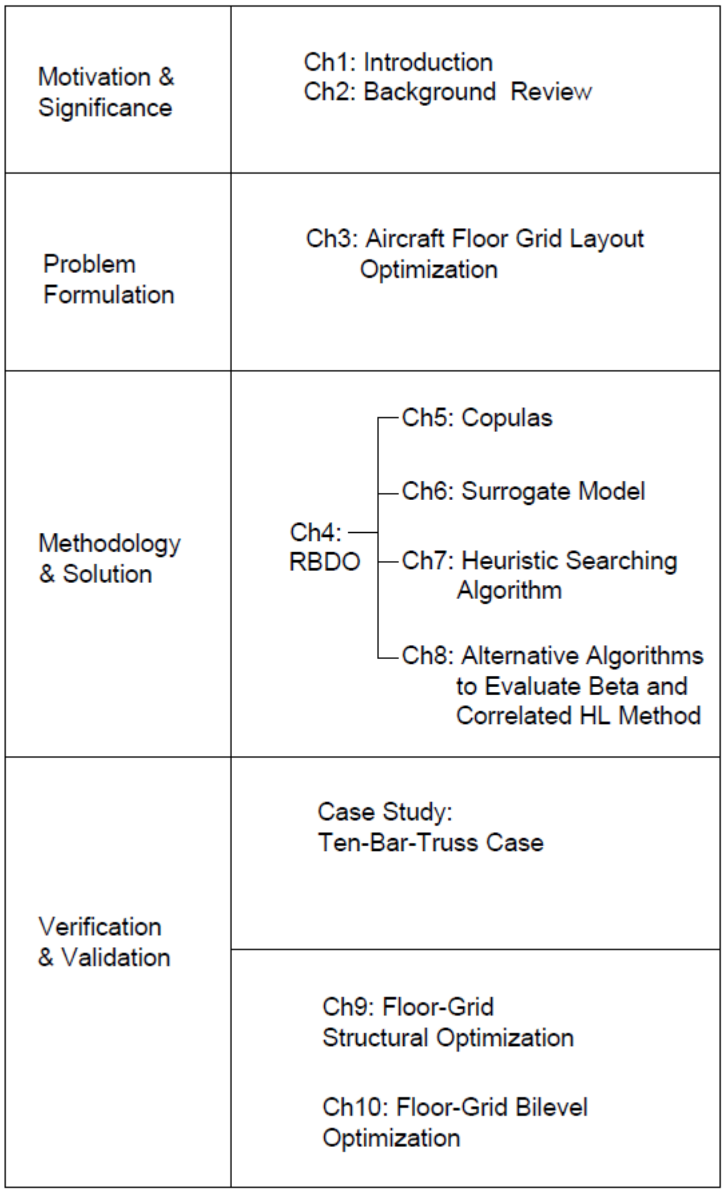
Motivation & Significance
Ch1: Introduction
Ch2: Background Review
Problem Formulation
Ch3: Aircraft Floor Grid Layout Optimization
Methodology & Solution
Ch4: RBDO
Ch5: Copulas
Ch6: Surrogate Model
Ch7: Heuristic Searching Algorithm
Ch8: Alternative Algorithms to Evaluate Beta and Correlated HL Method
Verification & Validation
Case Study: Ten-Bar-Truss Case
Ch9: Floor-Grid Structural Optimization
Ch10: Floor-Grid Bilevel Optimization

Figure 1-3: Outline of this thesis

In Chapter 2, the background knowledge for this thesis including probability and reliability-based design method, commonly used solutions for computationally intensive models, concept of random variable dependence and correlation are briefly reviewed. In Chapter 3, the aircraft floor grid layout optimization problem is formulated and an optimization model is given. Chapter 4 is a review of RBDO related algorithms and methods including analytical method and simulation based method as well as nonlinear optimization methods. A ten bar truss model and another example are given in this chapter to exemplify commonly used RBDO methods. This ten-bar-truss case also provides an example for further studies in the coming chapters. In Chapter 5, popular methods for modeling random variable correlations are reviewed and Copula as an alternative method for studying correlation is introduced. A method for applying copula in MCS is also introduced. In Chapter 6, a kriging-based surrogate model is proposed to reduce the MCS simulation intensity as well as to provide a response surface for optimization. Chapter 7 proposes a heuristic searching algorithm for optimization problems which have inequality constraints that do not have analytical form. In Chapter 8, some other algorithms including an alternative NLP based method to evaluate reliability index and a correlated HL method is proposed. In Chapter 9, an aircraft floor grid layout optimization problem is studied as an example of the general process of the reliability-based structural design and optimization, in which copula, Monte Carlo Simulation, heuristic gradient-based search algorithm and response surface model are applied. In Chapter10, the aircraft floor grid optimization problem is modeled as a bilevel optimization problem.

CHAPTER 2: BACKGROUND REVIEW

2.1 Uncertainty in Structural Design

Uncertainty is an inevitable phenomenon in structural design. Uncertainty in structural design could come from various sources including the unevenness of material properties, tolerance of structural member dimensions, stochastic external loads, etc. The structural designer should look for a safe structural design that could survive various uncertainties. Traditional safety factors were used to cover the influence of uncertainties. However, structural design based on safety factors is a coarse process which may cause heavy expenses and waste of material. Recently, probabilistic methods have been used which can give a safer design at the cost of certain computation expenses. However these methods would require the statistical parameters that could be computationally expensive to obtain.

Probabilistic methods for structural analysis include the study of performance functions, or limit state functions. A limit state function is the difference between the structural resistance capacity (e.g., yield strength, allowable displacement, permissible vibration level, etc.) and the response of the system (e.g., stress, actual displacement, actual vibration level, etc.). Reliability analysis methods can be classified into two categories: analytical method and simulation method. Analytical methods are easy to use and computationally inexpensive. Simulation methods include Monte Carlo Simulation (MCS) and Latin Hypercube Sampling (LHS). Simulation methods are computationally intensive and time consuming but they can be used for any structural model or limit state function. Simulation methods can also be used for multiple limit state functions.

Shown in Figure 2-1 is the taxonomy of reliability evaluation methods.

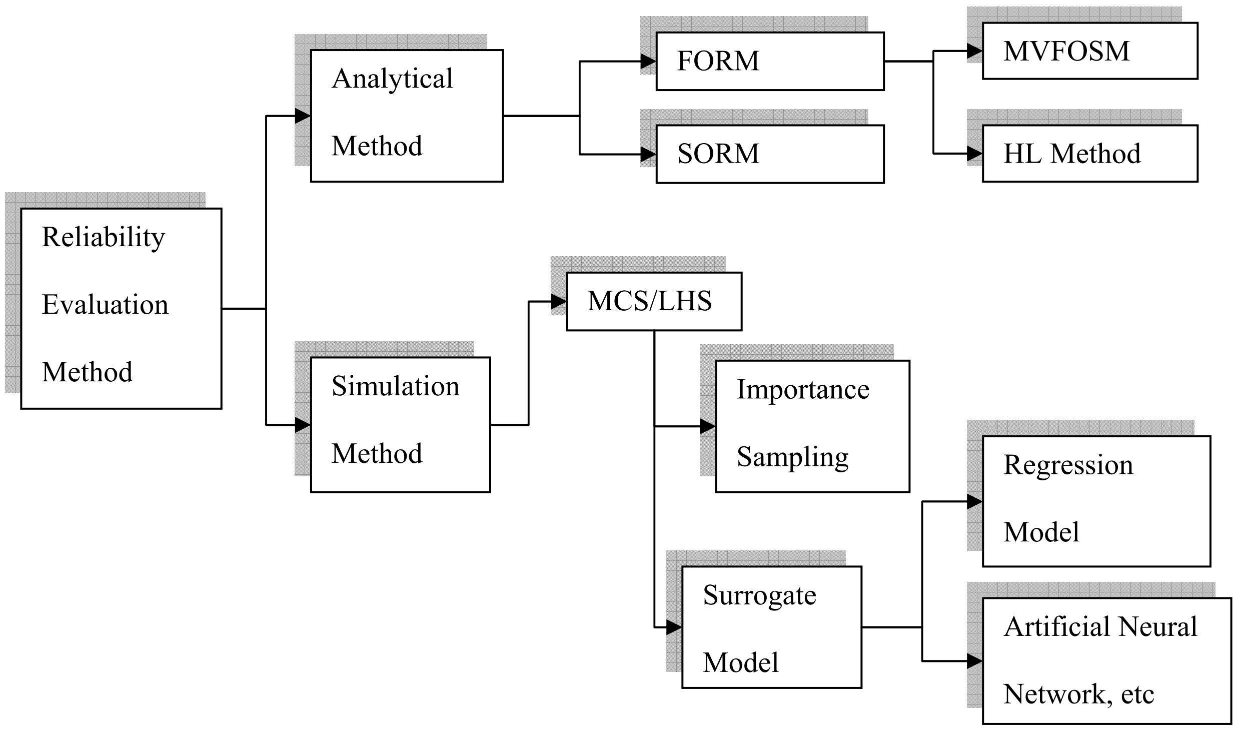

Figure 2-1: Taxonomy of reliability evaluation methods

2.2 Reliability-based Design Optimization

Structural engineers under competitive pressure always desire fast development of structural designs. This usually comes at the cost of reliability and robustness. In the early design phase it is always necessary to make decisions based on incomplete information and without physical or virtual tests. In order to facilitate this design methodology, it is necessary to find a way to assess system reliability and improve designs based on it. This is the main reason for the emergence of Reliability-based Design Optimization (RBDO).

In deterministic design the optimal solution usually has a high chance of failure due to the reason of uncertainties in the manufacturing process and uncertainties in the external environment. Uncertainties need be taken into consideration in design process. Optimized deterministic design solutions that do not consider uncertainties may be unreliable and may lead to structural failure. RBDO can be applied to address this problem. The objective of Reliability-based Design Optimization (RBDO) is to minimize the probability of failure of a structural design. While using RBDO, the design needs to make tradeoffs between reliability, weight, and cost.

The reliability of a structure design can be represented by a reliability index. The reliability index can be computed by performing a probabilistic reliability analysis. Some of the techniques used in reliability index evaluation include First Order Reliability Method (FORM) [Parkinson DB, 1978], Second Order Reliability Method (SORM)[Tvedt 1984, 1990][Hohenbichler 1987, 1988], and Monte Carlo Simulations (MCS) techniques. FORM and SORM are based on the Taylor series expansion and MCS/Latin Hypercube Sampling (LHS) are simulation methods that are computationally intensive. This section will give a brief review of a reliability-based method; more details of FORM and SORM can be found in [Choi,S.K. "Reliability-based Structural design"] and [Lemaire,M., "Structural Reliability"].

1. Limit state function and reliability index

The limit state function $g(X)$ is defined as the margin of safety between the resistance and the load:

$$g(X) := R(X) - S(X) \tag{2.1}$$

In which X are the random variables including design variables and material property variables and any other variables that can influence the structural strength.

Reliability-index or safety-index β is defined as:

$$\beta := \frac{\mu_{g(X)}}{\sigma_{g(X)}} \tag{2.2}$$

The reliability-index β can be interpreted as the distance from the mean performance point to the failure point, so it is desirable for β to be as big as possible.

If the limit-state function follows Gaussian distribution, there is a relationship between β and the probability of failure:

$$P_f = \Phi(-\beta) \tag{2.3}$$

In Gaussian limit-state function case, once β is known, the probability of failure can also be evaluated. The method to find β includes FORM and SORM, except for FORM and SORM. Other methods can also be used such as quadratic optimization methods.

2. MVFOSM

The limit-state function $g(X)$ can be approximated by first order Taylor series expansion at the mean value point $\overline{X}$:

$$g(X) \approx \tilde{g}(X) = g(\overline{X}) + \nabla g(\overline{X}) \cdot \left(X - \overline{X}\right) \tag{2.4}$$

Based on the mean value Taylor series expansion, if X_i's are independent to each other, the safety-index can be expressed as:

$$\beta = \frac{\mu_{\tilde{g}(X)}}{\sigma_{\tilde{g}(X)}} = \frac{g\left(\overline{X}\right)}{\sqrt{\sum_{i=1}^{n}\left(\frac{\partial g(\overline{X})}{\partial X_i} \cdot \sigma_{X_i}\right)^2}} \tag{2.5}$$

The drawback of MVFOSM is that for the same physical model, the different mathematical expression may result in different value of β. This drawback can be overcome by normalizing all the random variables to standard Gaussian distribution.

3. HL method

If all the random variables in the expression of a limit-state function have a Gaussian distribution, they can be normalized as standard Gaussian:

$$u_i = \frac{x_i - \mu_{x_i}}{\sigma_{x_i}} \; ; \qquad x_i = \mu_{x_i} + \sigma_{x_i} \cdot u_i$$

The Most Probable Point (MPP) U* is a point on $g(U)=0$ surface that has the minimum distance to the origin. The corresponding point in X coordinates is X*. Taking first order Taylor series expansion at MPP point U*,

$$\hat{g}(U) \approx \hat{g}(U^*) + \nabla \hat{g}(U^*) \cdot (U - U^*) \tag{2.6}$$

The safety-index can be express as:

$$\beta_{HL} = \frac{g(X^*) - \sum_{i=1}^{n} \frac{\partial g(X^*)}{\partial x_i} \sigma_{x_i} u_i^*}{\sqrt{\sum_{i=1}^{n} \left(\frac{\partial g(X^*)}{\partial x_i} \sigma_{x_i} \right)^2}} \tag{2.7}$$

4. SORM

SORM can be used to solve highly nonlinear problems. SORM is based on 2nd order Taylor series expansion of the limit-state function.

2.3 Simulation Methods

For reliability problems for which it is difficult to find an analytical solution, a simulation based method is an alternative method. However simulation-based methods usually requires a large number of samples to achieve high precision; thus, it demands high computational expenses.

1. Monte Carlo Simulation

Monte Carlo Simulation is based on Monte Carlo Sampling for generating random dataset that follow certain probabilistic distributions.

For a CDF function $y = F_X(x)$, the inverse CDF function is $x = F_X^{-1}(y)$. If one takes sample data of Y with uniform distribution on [0,1], the inverse CDF transform of Y will follow the distribution defined by the CDF function $F_X(x)$. This process is shown in Figure 2-2:

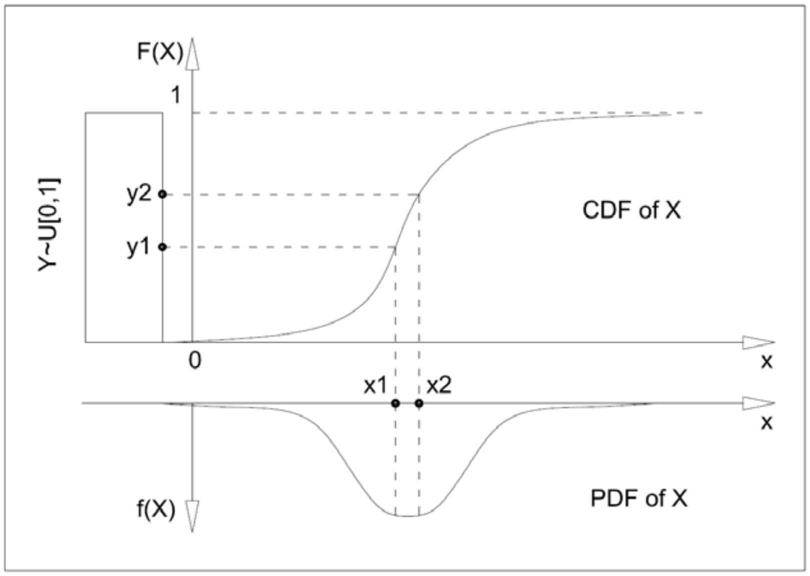

Figure 2-2: Monte Carlo sampling method

A dataset generated in this way will follow the desired distribution. It can be fed to the limit-state function to evaluate the probability of failure. Suppose there are *n* generated random data samples. After *n* times of simulation, the failure condition $g(X) < 0$ occurs *m* times. The probability of failure can be estimated as:

$$\widetilde{P}_f = \frac{m}{n}$$

An example of estimating probability of failure by Monte-Carlo Simulation is illustrated in Figure 2-3:

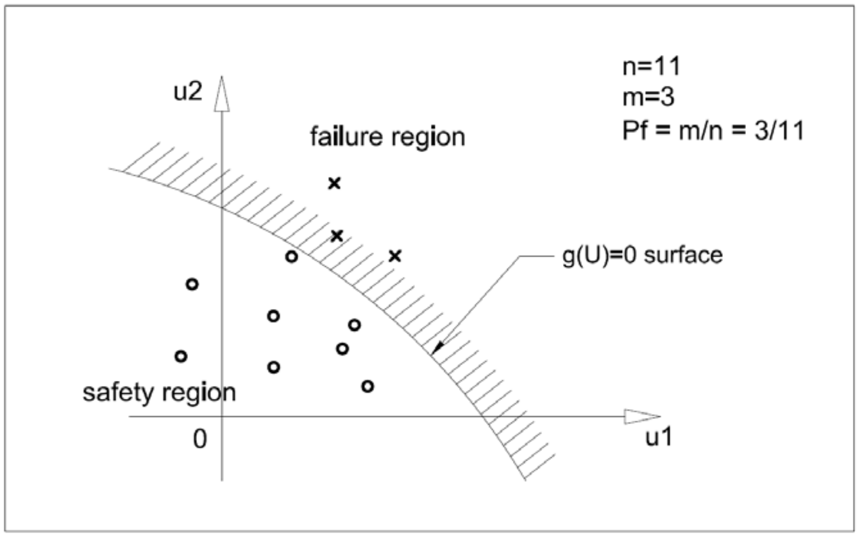

Figure 2-3: Illustration of failure rate in a simulation

2. Latin Hypercube Sampling

For multi random variables combination, LHS provides a method of high efficient sampling. The process of LHS is shown in Figure 2-4. Details about LHS can be found in [Choi, S.K., 2006].

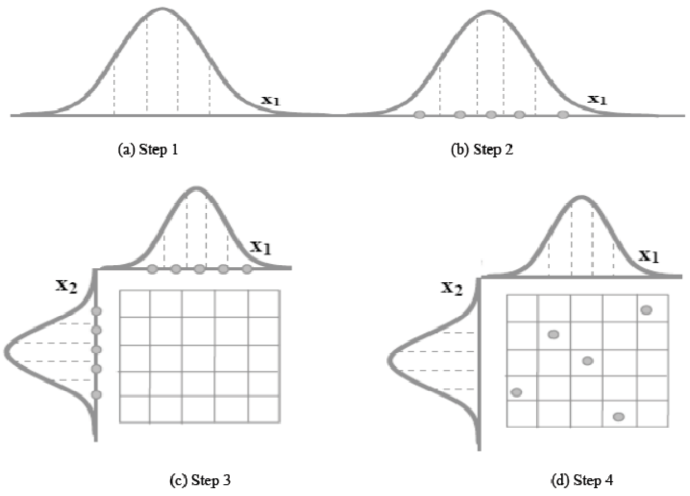

Figure 2-4: Latin Hypercube Sampling method (figure from Choi, S.K., 2006)

The procedure of LHS is listed below:

1) Divide the distribution of each variable into *n* equal probability non-overlapping intervals;

2) Sample one value on each of these *n* intervals in terms of its probability density;

3) For each random variable, select one sample from the *n* samples in a random manner;

4) Associate the random samples obtained from 3) of each random variables together, one group of random data is obtained.

2.4 Computation Expenses

For MCS/LHS based simulation method, to achieve high precision simulation results, a large number of samples are usually required. If the limit state function of the structure model is computationally intensive, e.g., based on finite element analysis, the simulation based on large number of samples will be computationally intensive and time consuming. It's necessary to find a method to reduce the computational expenses. There are two methods for this purpose: importance sampling and surrogate model.

Importance sampling is a variance reduction technique that can be used in the Monte Carlo method. The idea of importance sampling is that some values of the input random variables have more impact on the simulation than others. If these "important" values are sampled more frequently, the estimator variance can be effectively reduced. Thus, the basic method is to select a particular distribution that encourages the important values. Using these biased distributions will result in a biased estimator if these samples are applied directly in the simulation. To prevent this, the simulation outputs are corrected for the use of the biased distribution, and this makes sure that the new estimator is unbiased. [Doucet, A., et al, 2001].

Choosing a biased distribution that encourages the failure rate is important for importance sampling. If appropriate biased distribution are chosen, MCS would reduce the running time; if a bad distribution is chosen, the simulation time could be longer than that of the ordinary MCS.

Surrogate model is a model that mimics the original model as closely as possible. Surrogate model is computationally cheaper than original model. The inner work of surrogate model is not important. It can be similar to the original model or not, however, there should exist a resemblance in the input-output relation between the surrogate model and the original model. When there is only one variable as input, this method is usually known as curve fitting. The surrogate model can be constructed by interpolation/regression method, or by Artificial Neural Network [Papalambros, P.Y., 2000].

2.5 Random Variable Correlations and Dependence

One typical example of correlation and dependence is the resemblance in appearance between parents and their offspring. Another example of correlation and dependence is the demand and price of a product. Quote from Wikipedia: “Correlations are useful because they can indicate a predictive relationship that can be exploited in practice. For example, an electrical utility may produce less power on a mild day based on the correlation between electricity demand and weather. Correlations can also suggest possible causal, or mechanistic relationships; however, statistical dependence is not sufficient to demonstrate the presence of such a relationship.”

“Formally, dependence refers to any situation in which random variables do not satisfy a mathematical condition of probabilistic independence. In general statistical usage, correlation can refer to any departure of two or more random variables from independence, but most commonly refers to a more specialized type of relationship between mean values. There are several correlation coefficients, often denoted ρ or *r*, measuring the degree of correlation. The most common of these is the Pearson correlation coefficient, which is sensitive only to a linear relationship between two variables (which may exist even if one is a nonlinear function of the other). Other correlation coefficients have been developed to be more robust than the Pearson correlation, or more sensitive to nonlinear relationships” [from Wikipedia].

In RBDO, probability distributions are used to describe the stochastic nature of design variables and model parameters. Random variables are represented by standard deviations and mean values. It’s not usual for designers to consider the joint distributions of random variables because of lack of data and that most designers are unaccustomed to making judgments about the dependence of random variables. As a result, most researchers and designers estimate the marginal distributions of the random variables and make assumptions that the variables are independent. These assumptions can lead to significant errors in the estimation of probability of failure of a system. This can be seen from an illustrated example in Chapter 5. More details about independence and correlation can be found in [Papoulis, A. Pillai, S.U., 2002].

2.6 Optimization Method

In mathematics, optimization is a process to find a maximum or minimum value of a function with or without constraints. Optimization methods have significant importance because they help to find the best possible solutions for a design that can either maximize utility or minimize cost and environmental impact.

There are many optimization problems in engineering design and there are many optimization methods in practice. Generally, optimization methods can be divided into three categories: Experience based, Design of Experiments and Algorithm based [Roy,R., 2008]. Experience based optimization is a manual method, designers make design progress in an iterative manner. This is the typical traditional design optimization method. Design of Experiments (DoE) is a structured and organized method to determine the relationships between design factors and the performance. DoE is a scientific optimization method; however, DoE often requires that design variables be independent to each other which is usually not practical for most engineering design problems [Bailey, R.A. 2008]. Algorithm based optimization is now becoming popular. There are a variety of methods included in this category that can be applied to handle different kind of optimization problems. In this thesis, only the algorithm based optimization method will be applied.

Optimization problems can be divided into two categories: continuous optimization and discrete optimization. Continuous optimization is characterized by the continuity of design variables and the continuity of objective function. One typical example of continuous optimization is minimizing the weight of a cantilever beam with a square section under the constraint of maximum permissible stress. In this example, the objective function, beam weight, changes continuously as the design variable, length of the square side, changes. Discrete optimization includes: selecting the best route (i.e. least length) from the starting city to the destination city while passing every existing city on the map once only (Travelling Salesman Problem) and optimizing the topology of a structure to reduce the weight under the constraint of max permissible load.

For continuous optimization problems there are many solution methods such as linear programming, Nonlinear Programming (gradient based method), Genetic Algorithm, Lagrange multiplier method, etc [Roy,R., 2008]. For discrete optimization problems, it's usually difficult to obtain analytical solutions. It is also difficult to iterate exhaustively all the possibilities due to the NP-hard problem. For discrete optimization problems, a heuristic method can be applied, such as Ant Colony Optimization, which does not give exactly the best solution but it can find a relatively good solution in a short time [Bonabeau, E., 1999]. In this thesis only the gradient based optimization method will be discussed. The gradient based method (Gradient Projection Method and Reduced Gradient Method) will be applied in the reliability evaluation problem as an alternative method for evaluating the reliability index.

1. Unbounded Optimization Method

For unbounded optimization problems, local exploration can be applied. Since most objective functions do not have the simple polynomial form, local exploration can be used to approximate the objective function by first and second Taylor series expansion. There are two local exploration methods: Gradient Descent and Newton's Method. Gradient Descent is the first order approximation method, Newton's Method is the second order approximation method. More details can be found in [Papalambros, P.Y., 2000]. Gradient descent and Newton's method can be expressed as:

$$\begin{aligned} \mathbf{x_{k+1}} &= \mathbf{x_k} - \alpha_k \mathbf{g_k} \quad \text{(Gradient Descent)} \\ \mathbf{x_{k+1}} &= \mathbf{x_k} - H_k^{-1} \mathbf{g_k} \quad \text{(Newton's Method)} \end{aligned} \tag{2.8}$$

In which H is the Hessian matrix. Both of these two methods give a searching direction. Gradient Descent method searches along the opposite of the gradient direction while Newton's method does not necessarily search along the opposite gradient direction. Once the searching direction is determined, it's needed to find the minimum point along this direction. The objective is to find a step size α_k such that the objective function can obtain minimum value on this line:

$$\alpha_k = \arg\min_{0 \le x < \infty} f(\mathbf{x_k} + \alpha_k \cdot \mathbf{s_k}) \tag{2.9}$$

In which $\mathbf{s}_k$ is determined by Gradient Descent or Newton's method.

By taking the derivative and setting it equal to zero, the step size can be found as:

$$\alpha = \frac{\mathbf{g_k}^T \mathbf{g_k}}{\mathbf{g_k}^T H_k \mathbf{g_k}}; \tag{2.10}$$

The iteration becomes:

$$\mathbf{x_{k+1}} = \mathbf{x_k} - \frac{\mathbf{g_k}^T \mathbf{g_k}}{\mathbf{g_k}^T H_k \mathbf{g_k}} \mathbf{g_k} \tag{2.11}$$

2. Bounded Optimization Problem

Boundary optimization can be described by asking this question: if the objective function is subject to equality constraints and/or inequality constraints, while it's not easy to eliminate variables, how can constraint optimum be found?

Boundary optimization problems can usually be solved by introducing the Lagrangian function to incorporate the objective function with constraints. For inequality constraints, active inequality constraints and inactive inequality constraints can be separated; active inequality constraints can be regarded as equality constraints while inactive inequality constraints can be safely dropped.

The feasible iteration method is a method to generate feasible iterations (iterations that stays on the constraint surface) while decreasing the objective function at the same time. More details can be found in [Papalambros, “Principles of Optimal Design”, 2000]. Two methods of feasible iterations will be applied in Chapter 8 to solve a constrained optimization problem. The two feasible iterations are, respectively, Generalized Reduced Gradient Method and Gradient Projection Method.

2.7 Bilevel Optimization

When there are two different players in an optimization process, both of these players have their own objectives which are usually not the same or even conflictive. For this kind of problem, bilevel optimization method can be applied. The two players are identified respectively as leader and follower according to their importance and priority. The leader will first decide an optimal solution. Then the follower will make its decision to maximize its own utility based on the leader’s decision. Follower’s decision would be feedback to the leader. The leader would adjust its decision accordingly to maximize its own utility. This process continues until a balance between leader and follower is reached.

One of the most famous bilevel optimization models is the Stackelberg leadership model in which the leader moves first and then the follower move sequentially. In game theory terms, the players of this game are a leader and a follower and they compete on quantity [Simaan, M. and Cruz, J.B. Jr., 1973].

In this thesis, the structural optimization problem can also be modeled as a bilevel optimization problem with minimizing total weight as leader player and minimizing board weight as a follower player.

CHAPTER 3: AIRCRAFT FLOOR GRID LAYOUT OPTIMIZATION

Structural design is not only important to aircraft safety, but also plays a key role in reducing aircraft cost and increasing performance. The airplane cost is closely related to the structure weight. Typically aircraft cost is \$200-\$500 per pound [Desktop Aeronautics Inc., 1997-2006]. For military aircraft such as the B-2, the cost is even higher. Except for its direct impact on aircraft cost, the aircraft structural weight also affects performance. Every kilogram of increase in structural weight means one kilogram of less fuel and cargo when the take-off weight is fixed. In Figure 3-1, the left figure shows the concept of aircraft structure frame; the right figure shows the floor board in the fuselage.

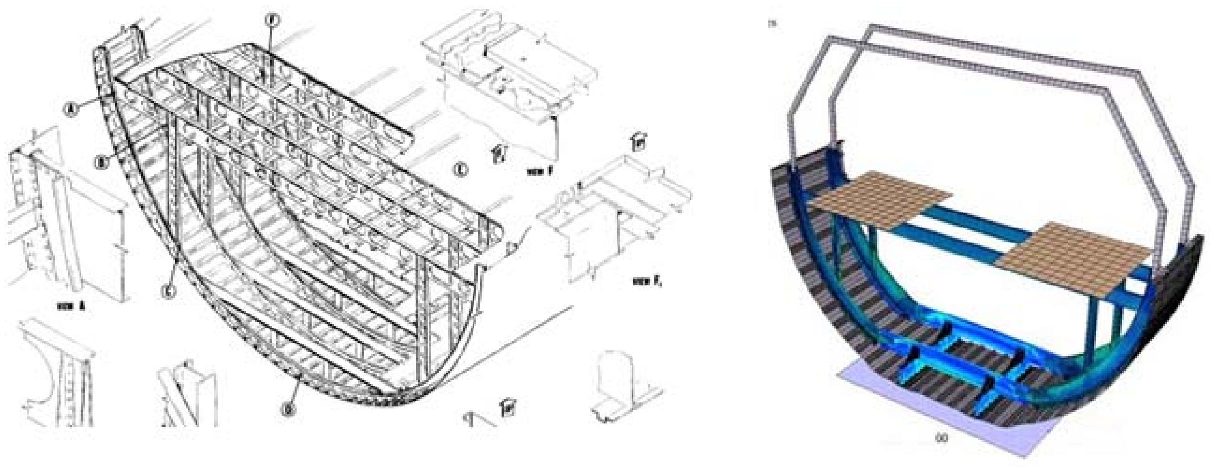

Figure 3-1: Aircraft structures (Pictures from internet)

The aircraft floor board is an important structural component in aircraft design. It divides the fuselage into an upper and lower compartment. The upper compartment is usually the passenger compartment. The lower compartments are usually cargo holds and the machinery room. Due to the arrangement of the passenger compartment, the floor board carries the load of passenger weight, luggage weight and seat weight. For some special areas like the kitchen and lavatory, floor boards and grids need special designs to support the extra loads. See Figure 3-2. Aircraft floor board and grid design also need to take into consideration the dynamic load caused by vertical accelerations during taking off and landing.

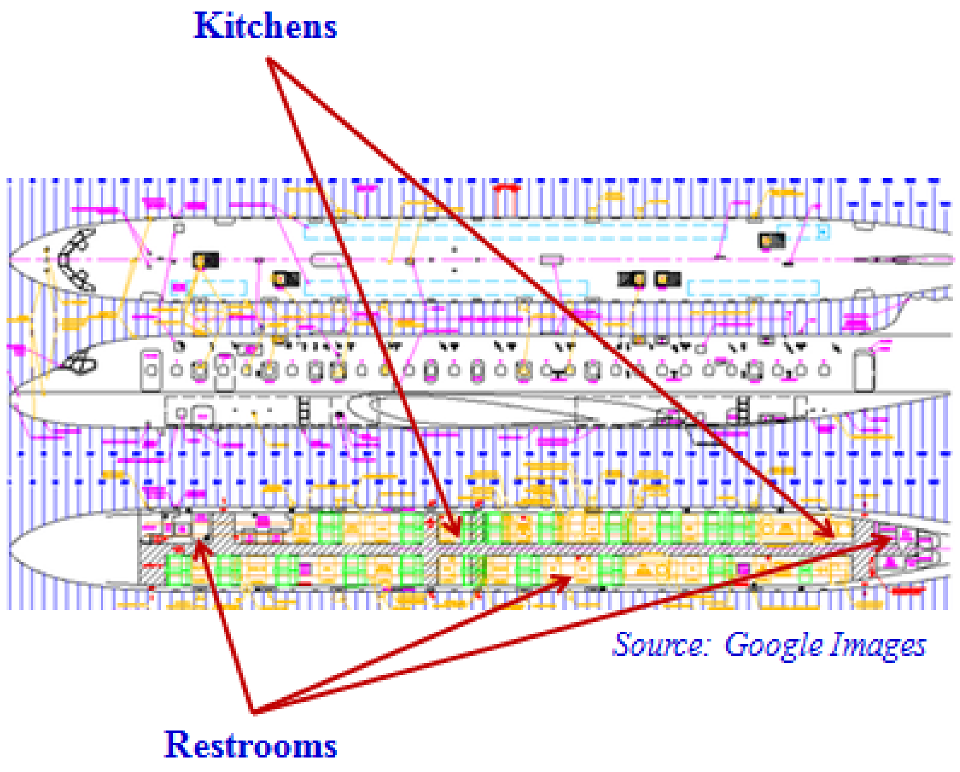

Figure 3-2: Aircraft general arrangement (Picture from internet)

Weight control is very important for airplane design. Weight reduction can be attained mainly by introducing new material that has low density and high strength, such as composite material and titanium, which have been widely applied in aviation industry. Another method to reduce weight is structural optimization. The objective of structural optimization is to reduce the structural weight while still satisfying the structural strength requirement.

One typical method of structural optimization applied in aircraft design is topology optimization. Topology optimization is a method based on finite element model. The basic idea is to find out the stress distribution on the structure via finite element analysis. Then decrease the material in areas that have less stress and increase material in areas with high stress. This is an iterative process. After several iterations it will find the most efficient structural design with less material and better structural strength. Figure 3-3 is an example of topology optimization process. The left figure is the original cantilever structure with a load P at the far end. The right figure is the optimized structure that has smaller weight while still keeping enough structural strength. Topology optimization is a process of reaching a tradeoff between minimizing material weight and maximizing structural strength. A brief introduction of topology optimization can be found in [Sigmund, O., 2000]

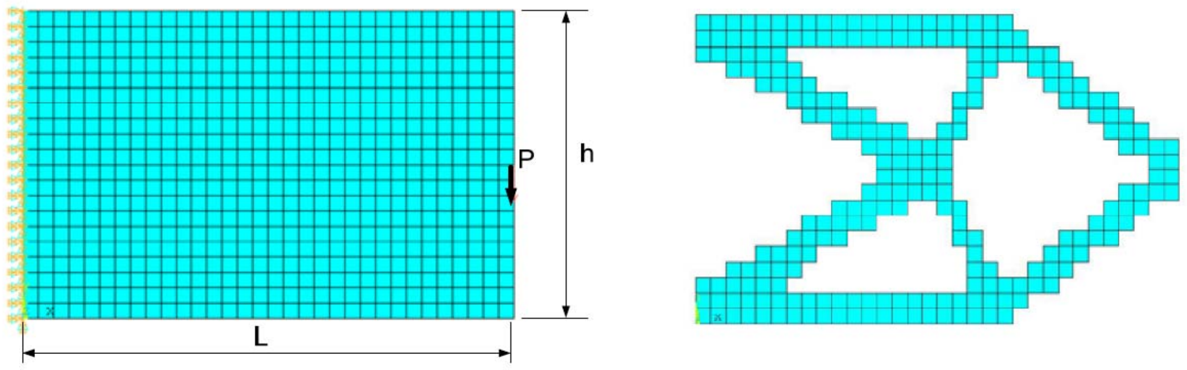

Figure 3-3: Topology optimization for a cantilever beam (figure from Kim S.R. et al, 2007)

The topology optimization method has been applied in aircraft wing truss structure design. This method also contributes to floor grid layout optimization.

Figure 3-4 is the top view of the passenger deck of an airplane at the front and the rear. The grids are the transverse beams and longitudinal tracks.

The loads on the deck are transmited to the longitudinals by deck board; the longitudinal tracks are supported by transverse beams; Transverse beams are fixed on both sides to the structural members on the side wall. This is illustrated in Figure 3-5.

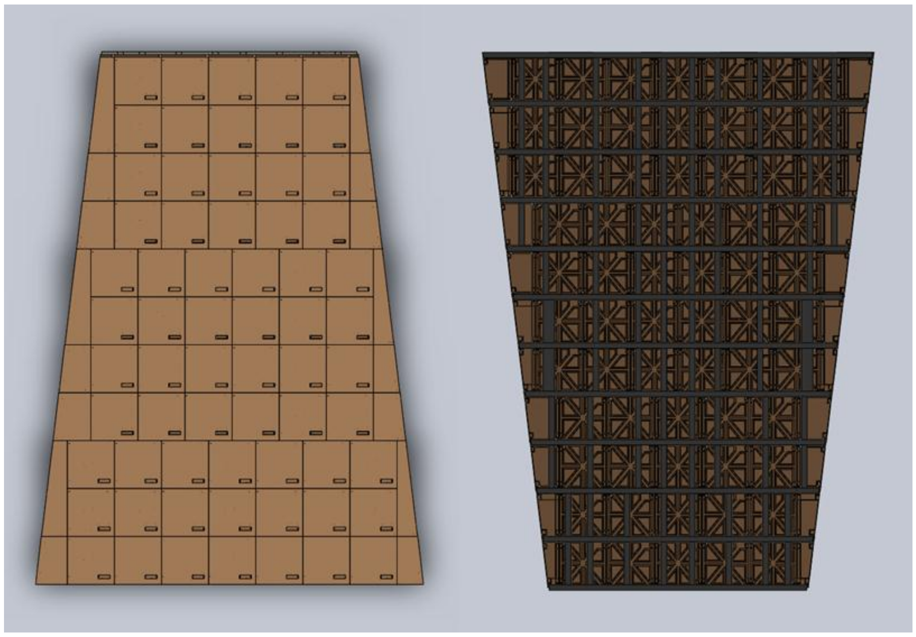

Figure 3-4: Aircraft floor board and floor grid

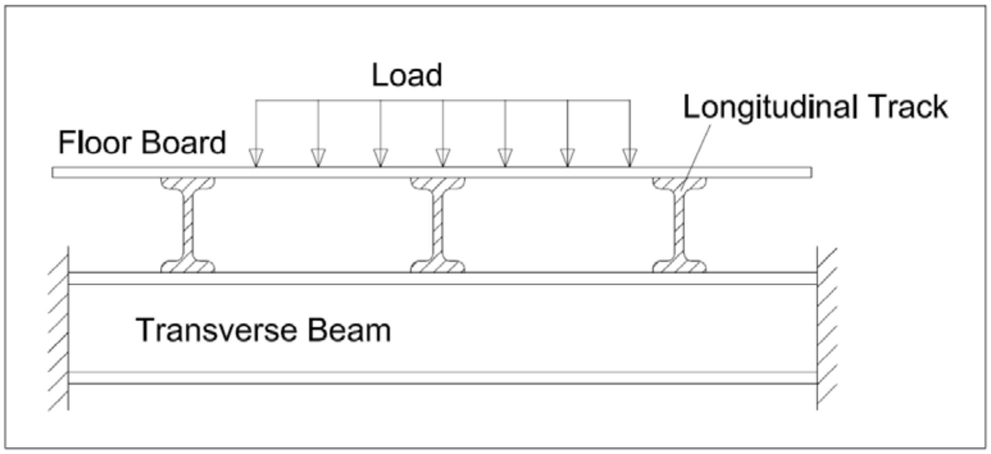

Figure 3-5: Aircraft floor grid, board, beams and tracks

Due to different loads on the floor at different regions of the aircraft, it's necessary to divide the whole floor board into different areas, each of which should be examined by applying the corresponding load conditions as shown in Figure 3-6.

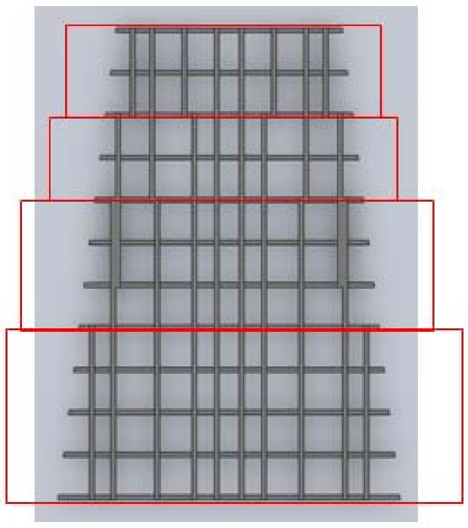

Figure 3-6: Floor grid in different areas has different load conditions

In this thesis, study will be focused on reliability-based floor grid layout optimization. The objective of the problem is to minimize the total weight of the floor grid and floor board. The constraints include the reliability requirement represented by reliability index. This model can be represented as:

$$\begin{aligned} &\text{minimize} \quad && W_{total}(\mathbf{X}) = W_{grid}(\mathbf{X}) + W_{board}(\mathbf{X}) \\ &\text{subject to} \quad && \boldsymbol{\beta}(\mathbf{X}, \mathbf{R}) \geq \boldsymbol{\beta}_0 \end{aligned} \tag{3.1}$$

In which $\boldsymbol{\beta}$ is a vector of reliability indices and $\boldsymbol{\beta}_0$ is the minimum permissible $\boldsymbol{\beta}$ value. $\mathbf{X}$ is a vector of design variables and $\mathbf{R}$ represents random variable distributions and correlations.

To satisfy the weight requirement, less material should be used; to satisfy the reliability requirement, more material should be used. This optimization needs to reach a balance or tradeoff between these two objectives or constraints.

CHAPTER 4: RELIABILITY-BASED STRUCTURAL DESIGN AND OPTIMIZATION

4.1 Introduction

Traditional deterministic design optimization leaves no margin for the variations of design variables. This is associated with high probability of product failure due to the uncertainties during manufacturing processes and uncertainties of the external load. Optimization without considering uncertainties from material properties and the manufacturing phase may result in catastrophic failure. The uncertainties coming from design and manufacturing can be represented by reliability. Reliability-based structural design and optimization methods can be introduced in structural design to address this problem.

Reliability evaluation provides the input for designer about the reliability of the design. The ultimate goal is to improve the design or optimize the design subject to the constraint of reliability constraints. The optimized design solution needs to satisfy the reliability requirement as well as the performance requirement. This process is illustrated in Figure 4-1.

The process of reliability-based structural design and the optimization process shown in Figure 4-1 can be explained as follows. First determine the design variables and random variables. Then according to the optimization objective, which is usually to minimize weight or cost, write the objective function as a function of design variables and random variables. Determine the performance constraints, which are usually requirements from manufacturability, material availability, design code, or product aesthetics. Performance constraints include both equality constraints and inequality constraints. The next step is to determine the reliability constraints which may include a stress reliability requirement and a deformation reliability requirement.

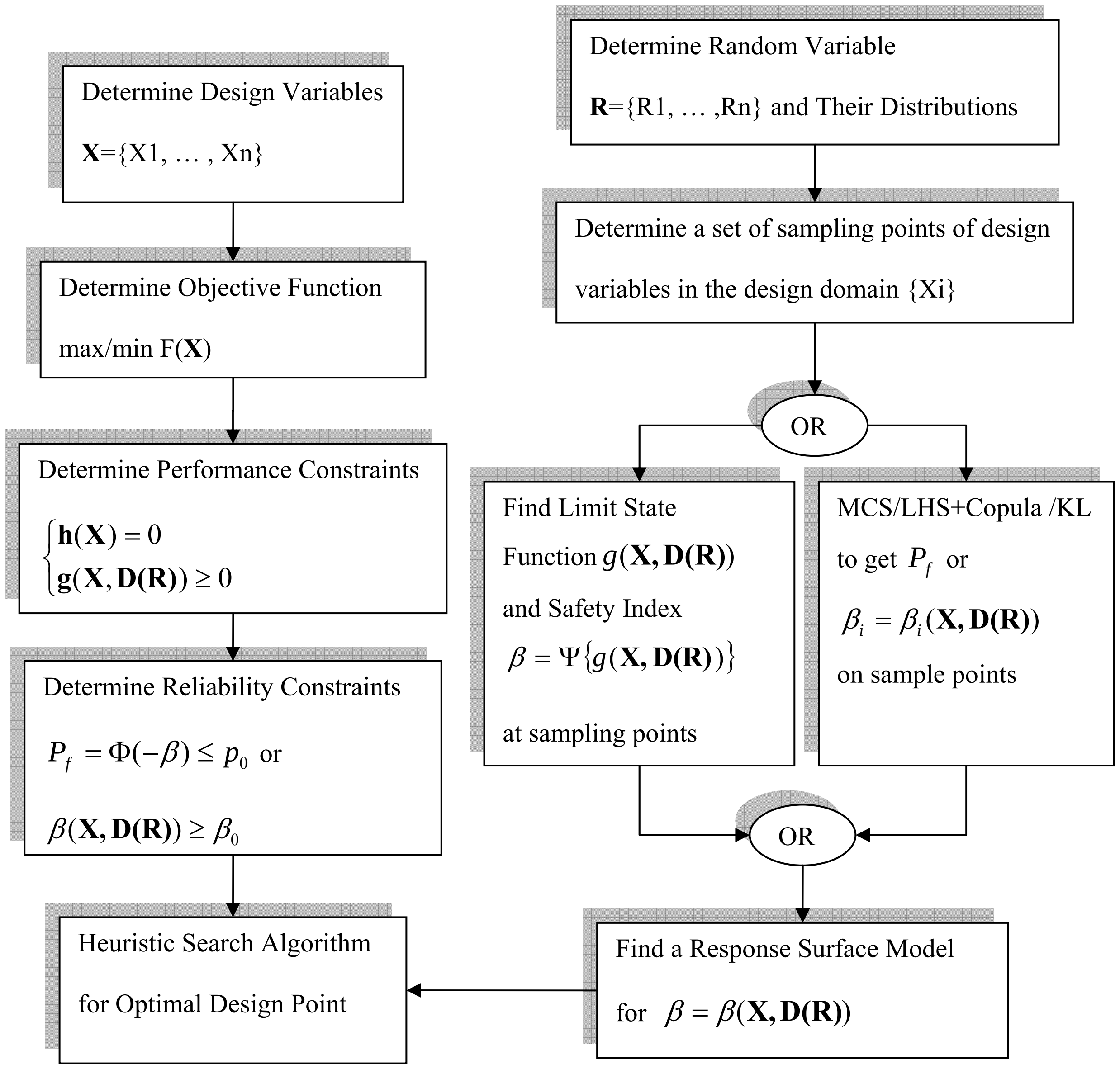
Determine Design Variables
X={X1, … , Xn}
Determine Objective Function
max/min F(**X**)
Determine Performance Constraints
$\begin{cases}\mathbf{h}(\mathbf{X})=0\\ \mathbf{g}(\mathbf{X},\mathbf{D}(\mathbf{R}))\ge 0\end{cases}$
Determine Reliability Constraints
$P_f=\Phi(-\beta)\le p_0$ or
$\beta(\mathbf{X},\mathbf{D}(\mathbf{R}))\ge\beta_0$
Heuristic Search Algorithm
for Optimal Design Point
Determine Random Variable
R={R1, … ,Rn} and Their Distributions
Determine a set of sampling points of design
variables in the design domain {Xi}
OR
Find Limit State
Function $g(\mathbf{X},\mathbf{D}(\mathbf{R}))$
and Safety Index
$\beta=\Psi\{g(\mathbf{X},\mathbf{D}(\mathbf{R}))\}$
at sampling points
MCS/LHS+Copula /KL
to get P_f or
$\beta_i=\beta_i(\mathbf{X},\mathbf{D}(\mathbf{R}))$
on sample points
OR
Find a Response Surface Model
for $\beta=\beta(\mathbf{X},\mathbf{D}(\mathbf{R}))$

Figure 4-1: Reliability-based structural design optimization flow chat

The reliability constraints usually take the form of $Pf\le P_0$, in which P_0 is the maximum allowable probability of failure. To utilize the existing reliability-based methods and algorithms, the constraint $Pf\le P_0$ is replaced by $\beta\ge\beta_0$, in which β is the reliability index. Then a nonlinear optimization problem is obtained. For nonlinear optimization problems, usually the gradient of the objective function and constraints are required. However, for RBDO problems, the reliability indices are usually computed by

Monte Carlo simulations and it’s difficult to obtain the analytical form of reliability indices. To overcome this dilemma, a response surface model based on interpolation is used. Given a point, the response surface model can provide a reliability index value at this point as well as the gradient information at this point. Since there is not a general form to represent every point on the response surface, each individual point as well as the gradient at this point needs be computed specifically. A heuristic searching algorithm is used to find the optimal point.

If the random variables are uncorrelated, many existing methods such as MVFOSM, HL and SORM can be used to compute the reliability index. However, if correlation exists for random variables, simulation based methods like MCS/LHS, copula, KL expansion can be used to obtain probability of failure as well as the reliability index. Results obtained by both analytical and simulation methods are individual discrete values.

In this chapter, literatures about reliability-based structural design will be briefly reviewed. The ten bar truss model, a classic model, will be presented as an example for all the methods and algorithms. In the end of this chapter, some prevalent reliability-based methods will be applied to this ten bar truss model.

4.2 Literature Review

For structural reliability problems, there are many different approaches. Most commonly applied methods include first order reliability method (FORM) [Parkinson, D.B. 1978] and second order reliability method (SORM) [Tvedt, L., 1984, 1990]. In [Rackwitz, R., 2001], methods and theories of reliability-based structural design methods are reviewed and briefly summarized, some extreme examples are demonstrated to explain where and why these methods do not work. Algorithm problems are also addressed in this thesis; some new potential application fields are outlined. Detailed introduction to structural reliability methods can be found in [Choi, S.K., “Reliability-based Structural Design”,2006]. Finite Element Method (FEM) has been widely used as a structural evaluation tool. Traditional Finite Element Analysis (FEA) method is based on dividing the continuous structures into discrete meshes and interpolating on each mesh. Based on the FEA method, Stochastic FEA has been systematically introduced by Ghanem in [Ghanem, R.G., 2003]. In Stochastic FEA, the material properties of discrete meshes are treated as random fields. Stochastic FEA is more realistic in simulation than traditional FEA, since in the real world material properties can usually be modeled as random fields. In [Choi, S.K., 2006], a stochastic analysis procedure for Gaussian random field in structural reliability estimation is proposed based on K-L transform and Polynomial Chaos Expansion. K-L transform can be used to produce multidimensional correlated random variables when the covariance matrix is given. More details of KL expansion can be found in [Papoulis, A., 2002].

Many airplane structural optimization problems can be solved using the topology optimization method. Topology optimization is also called layout optimization or shape optimization. The topology optimization method can find its origin from some papers published in the early 1900's by Michell, who proposed some optimal conditions for load-carrying structures. The emergence of numerical methods such as FEA and modern computer make it possible for topology optimization to be applied in the engineering field. Some introductory examples of topology optimization can be found in [Sigmund, O., 2000]. The systematic method of topology optimization has been developed by Bendsoe and Kikuchi [Bendsoe, M. and Kikuchi, N., 1988]. Topology optimization has been widely used in airplane structural design, especially in aircraft wing box ribs design [Li, G., Wang, H., et al, 2000],[Liu, B. et al, 2000]. The strategy of topology optimization is to divide the continuous structural material into discrete meshes as building blocks, and study the stress on each building block. Based on the different stresses on each building block, reduction can be made to remove the blocks that are carrying smaller loads. This is an iterative process; in the end the optimal shape can be obtained with minimized material usage and acceptable structural reliability.

4.3 Ten-Bar-Truss Model as a Typical Example

The ten bar truss model is a widely used model for reliability-based structural design as shown in Figure 4-2. The ten-bar-truss model is an over-constrained structural model comprised of 4 horizontal rods, 2 vertical rods and 4 diagonal rods. For simplicity, suppose all horizontal rods have the same section area, as well as vertical and diagonal rods. These areas are denoted by A1, A2 and A3. The math model of this ten-bar-truss model can be found in [Kumar, S., et al, 2009].

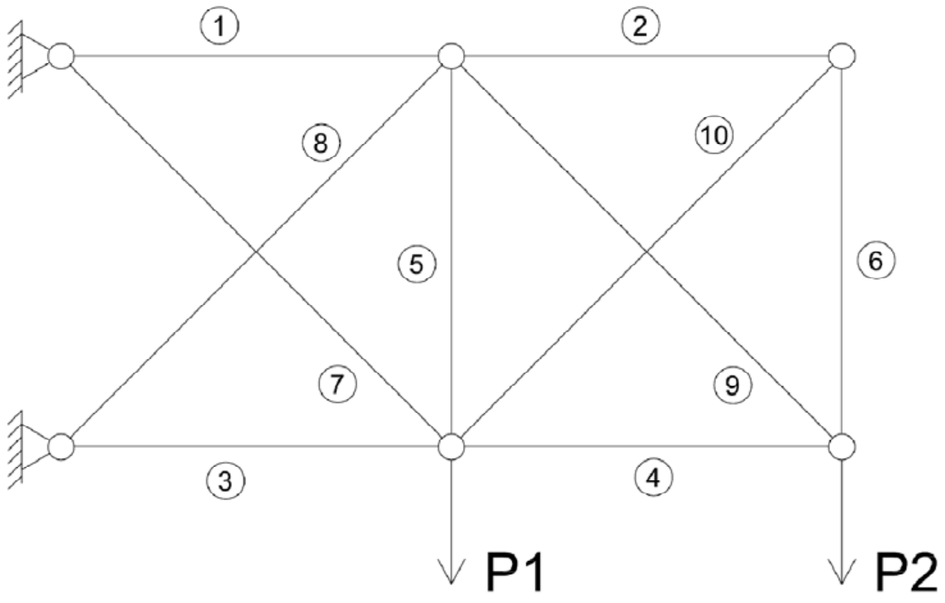

Figure 4-2: Ten-bar-truss model

In this thesis the ten-bar-truss will be studied from two perspectives:

Perspective 1: the rod section areas are fixed as deterministic variables and the external loads P1 and P2 are given as random variables. Study will be focused on the relation between P1 and P2 and its impact on structural reliability. Correlations between random variables are modeled by multivariate distribution method, Karhunen-Loeve expansion method and Copula.

Perspective 2: the external loads P1 and P2 are fixed values and the rod section areas (A1, A2, A3) (A1 is horizontal members, A2 vertical members, A3 diagonal members) are regarded as random variables following Gaussian distributions. This thesis will examine the structural reliability via safety index β and probability of failure Pf. The safety-index and probability of failure will be obtained by a variety of methods (MCS, MCS based on Kriging model, MVFOSM, HL method, correlated HL method, SORM, Gradient Projection Method, and Reduced Gradient Method). The results of these methods will be compared to each other in terms of precision and speed.

The relationship between the internal force Ni of each rod and the external loads P1 and P2 can be expressed as:

$$[N] = [A|B]\begin{bmatrix} N \\ P \end{bmatrix}$$

$$\Rightarrow [N] = [A][N] + [B][P]$$

$$\Rightarrow [I-A][N] = [B][P]$$

$$\Rightarrow [N] = [I-A]^{-1}[B][P]$$

Positive values of N represent extension; negative represents compression. The A and B matrices are respectively:

$$[A] = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{\sqrt{2}} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{\sqrt{2}} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{\sqrt{2}} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}; \qquad [B] = \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ -1 & -2 \\ 0 & -1 \\ 0 & -1 \\ 0 & 0 \\ \sqrt{2} & \sqrt{2} \\ a_1 & a_2 \\ 0 & \sqrt{2} \\ a_3 & a_4 \end{bmatrix}$$

In the [B] expression, a_i's are:

$$a_1 = \frac{-a_{22}\sqrt{2}\left(\frac{1}{A_3} + \frac{2\sqrt{2}}{A_7}\right)}{a_{11}a_{22} - a_{12}a_{21}}$$

$$a_2 = \frac{a_{22}\sqrt{2}\left(\frac{1}{A_1} - \frac{2}{A_3} - \frac{1}{A_5} - \frac{2\sqrt{2}}{A_7}\right) + a_{12}\left(\frac{\sqrt{2}}{A_4} + \frac{\sqrt{2}}{A_5} + \frac{4}{A_9}\right)}{a_{11}a_{22} - a_{12}a_{21}}$$

$$a_3 = \frac{a_{21}\sqrt{2}\left(\frac{1}{A_3} + \frac{2\sqrt{2}}{A_7}\right)}{a_{11}a_{22} - a_{12}a_{21}}$$

$$a_4 = \frac{-\left(a_{11}\left(\frac{\sqrt{2}}{A_4} + \frac{\sqrt{2}}{A_5} + \frac{4}{A_9}\right) + a_{21}\sqrt{2}\left(\frac{1}{A_1} - \frac{2}{A_3} - \frac{1}{A_5} - \frac{2\sqrt{2}}{A_7}\right)\right)}{a_{11}a_{22} - a_{12}a_{21}}$$

$$a_{11} = \left(\frac{1}{A_1} + \frac{1}{A_3} + \frac{1}{A_5} + \frac{2\sqrt{2}}{A_7} + \frac{2\sqrt{2}}{A_8}\right)$$

$$a_{12} = a_{21} = \frac{1}{A_5}$$

$$a_{22} = \left(\frac{1}{A_2} + \frac{1}{A_4} + \frac{1}{A_5} + \frac{1}{A_6} + \frac{2\sqrt{2}}{A_9} + \frac{2\sqrt{2}}{A_{10}}\right)$$

The numerical value of $[I - A]^{-1}[B]$ for the given section areas relates the internal forces to the external load, it is given in Table 4-1 for future reliability analysis. Internal forces can be computed by this table when external load P1 and P2 are known.

Table 4-1 Load coefficients

rod #	P1 coeff.	P2 coeff.
1	0.403894	1.521702
2	-0.0744	0.403894
3	-0.59611	-1.4783
4	-0.0744	-0.59611
5	0.329491	-0.0744
6	-0.0744	0.403894
7	0.843021	0.676416
8	-0.57119	-0.7378
9	0.105223	0.843021
10	0.105223	-0.57119

4.3.1 Two Random Variable Example

For the convenience of the example, without losing generality, set the section area of horizontal rod (A_hor), vertical rod (A_ver), diagonal rod (A_dia), maximum permissible stress and external forces P1 and P2 as follows:

A_hor=1e-4 m^2;

A_ver=0.3e-4 m^2;

A_dia=0.6e-4 m^2;

σ_{allow} = 100 MPa;

P1~Normal(3000, 500^2) N;

P2~Normal(3000, 500^2) N;

As an initial evaluation, setting external load P1 and P2 as their deterministic mean value, the stresses in each rod can be calculated. According to these, values the reliability of rod 3,7 will be examined in a later study.

The code stress.m, in the appendix can be used to experiment a series of external force combinations to evaluate the failure probabilities. To make comparisons, 4 different groups of experiments will be done to determine the probability of failure, with each group containing 1,000,000 data samples. The first group of experiments is based on uncorrelated bivariate Gaussian distribution of external loads P1 and P2. The second group of experiment is based on correlated bivariate Gaussian distribution of P1 and P2. The third group of experiment uses correlated P1 and P2 data generated by KL expansion. The fourth group of experiment is based on copula; this group includes sub-groups of 3 families of Archimedean copula.

4.3.2 Three Random Variable Example

In the previous case, the external loads P1 and P2 were considered random variables while other design variables were all deterministic. However, in this case, the limit state function of each rod is a linear function of P1 and P2. This can be seen from table 4-1. To make this example more general with nonlinear functions, P1 and P2 will be set as deterministic variables while the rod section areas will be set to random variables. A1, A2, A3 are, respectively, horizontal, vertical, diagonal rod section areas. They have normal distributions with a coefficient of variance 0.2.

In this three RV case, study will be focused on evaluating the value of safety-index β and the corresponding probability of failure Pf. This problem will be approached using 2 different methods, the first method is existing the first order and second order reliability method (FORM & SORM); the other is by viewing the problem as a constrained optimization problem, thus, solving it from a gradient based quadratic programming approach. The second method requires many iterations. However, as long as there are enough iterations, high precision can be achieved. Both of these two methods can achieve highly accurate results.

Usually reliability models are very computationally expensive, especially when the model is based on Finite Element. It's not uncommon that running a FEA model once would take several hours. However, MCS requires large number of experiments to obtain precise results. For computationally expensive models this will become mission impossible. That's why it's necessary to find a surrogate model to replace the original model. The surrogate model is required to be computation inexpensive at the same time having relatively high accuracy. There are many methods to construct a surrogate model, for example, interpolation method, curve fitting method, Artificial Neural Network method, and many others. In this thesis, a surrogate model based on Kriging will be discussed.

The 3 random variable model is shown below:

A1,A2,A3~Normal;

E(A1)=1e-4 m^2;

E(A2)=0.3e-4 m^2;

E(A3)=1e-4 m^2;

$c_v = 0.2$;

$\sigma_{allow} = 100$ MPa;

P1=P2=3000;

The limit state function of rod 3 and rod 7 are listed below [Kumar, S., et al, 2009]:

Rod3:

$$g(X) = \frac{P}{X_1}\left\{2 + \frac{X_1 X_2 X_3\left(2\sqrt{2}X_1 + X_3\right)}{D_T}\right\} - \sigma_{allow} \tag{4.1}$$

Rod7:

$$g(X) = \frac{P}{X_3}\left\{2 + \frac{\sqrt{2}X_1 X_2 X_3\left(2\sqrt{2}X_1 + X_3\right)}{D_T}\right\} - \sigma_{allow} \tag{4.2}$$

In which X1, X2, X3 are rod section area A1, A2 and A3, and

$$D_T = 4X_2^2\left(8X_1^2 + X_3^2\right) + 4\sqrt{2}X_1X_2X_3\left(3X_1 + 4X_2\right) + X_1X_3^2\left(X_1 + 6X_2\right)$$

4.4 Illustrated Examples

In this section, the ten bar truss model with three random variables example will be studied by several reliability-based method. Three analytical methods to find probability of failure are applied to evaluate the probability of failure of ten-bar-truss model. The methods include Mean Value First Order Second Moment method (MVFOSM); Hasofer-Lind (HL) method; Second Order Reliability Method (SORM). MVFOSM is a simple method to evaluate Pf, however it's based on first order Taylor series expansion at mean value, the result usually has big error for nonlinear limit-state functions. HL method first transforms all random variables to standard Gaussian distribution then take first order Taylor series expansion at Most Probable Point (MPP). HL method has relatively high precision. SORM is based on transforming random variable from standard Gaussian U space to Y space then take second order Taylor series expansion, it provides high accuracy evaluation of Pf.

4.4.1 Mean Value First Order Second Moment method (MVFOSM)

In this section, the probability of failure will be determined by the analytical method of Mean Value First Order Second Moment method (MVFOSM), comparison will be made between MVFOSM result and MCS result.

1. Rod section areas Xi independent, compute the Pf and safety index beta for rod 3 and rod 7 by MVFOSM method and MCS

One of the good properties of reliability index or safety index β is that it is directly related to probability of failure if the limit state function g follows Gaussian distribution. P_f can be computed conveniently when β is known.

$$
\begin{aligned}
P_f &= \int_{-\infty}^{0} f_G(g)dg = \int_{-\infty}^{0} \frac{1}{\sigma_g\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{g-\mu_g}{\sigma_g}\right)^2\right]dg \\
&= \int_{-\infty}^{0}\left[\frac{1}{\sqrt{2\pi}}\exp\left[-\frac{1}{2}\left(\frac{g}{\sigma_g}-\beta\right)^2\right]\right]d\left(\frac{g}{\sigma_g}-\beta\right) \\
&= \int_{-\infty}^{-\beta}\left[\frac{1}{\sqrt{2\pi}}\exp\left[-\frac{1}{2}(g')^2\right]\right]d(g') \\
&= \Phi(-\beta)
\end{aligned}
\tag{4.3}
$$

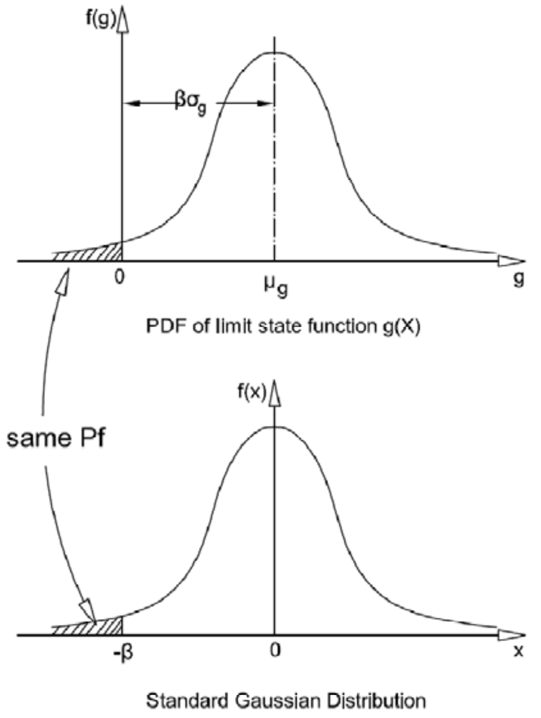
f(g)
βσg
0
μg
g
PDF of limit state function g(X)
same Pf
f(x)
-β
0
x
Standard Gaussian Distribution

Figure 4-3: Physical meaning of reliability index in MVFOSM

However, since g(x) is usually a nonlinear function of random variables, even the random variables follows Gaussian distributions, g(x) is not necessarily Gaussian. When g(x) is approximated by its first order Taylor expansion, the approximation, $\tilde{g}$, is a linear function of random Gaussian variables, $\tilde{g}(X) = g(\mu_X) + \nabla g(\mu_X)\cdot(X-\mu_X)$,thus $\tilde{g}$ follows Gaussian distribution also.

Based on the linearization approximation of limit-state function, β can be computed:

$$\beta = \frac{\mu_{g(X)}}{\sigma_{g(X)}} \approx \frac{\mu_{\tilde{g}(X)}}{\sigma_{\tilde{g}(X)}} = \frac{g(\mu_X)}{\sqrt{\sum_{i=1}^{n}\left(\frac{\partial g(\mu_X)}{\partial x_i}\sigma_{x_i}\right)^2}} \tag{4.4}$$

Based on β, the probability of failure can also be computed by

$$P_f = \Phi(-\beta) \tag{4.5}$$

To find the accuracy of MVFOSM, the results from MCS and from MVFOSM are listed in Table 4-2:

Table 4-2: Comparison between MVFOSM and MCS results

	MVFOSM reliability index		MCS (1000 samples)
Rod (Ai independent)	β	P_f	P_f
3	2.9953	0.0014	0.029
7	2.8889	0.0019	0.028

Obviously there is great difference between the results of MVFOSM reliability method and MCS method. Since MCS is more close to the real situation, the MVFOSM method resulted in big error in this example. The reason of this error can be attributed to the high nonlinearity of the model and the expansion point at mean value, while the MVFOSM is a linear approximation of the model, when expanding the limit state function at mean value rather than MPP point, error could be big. To overcome this drawback next step the Hasofer-Lind method will be examined.

4.4.2 Hasofer-Lind method

Rod section areas Xi independent, compute the Pf and safety index beta for rod 3 and rod 7 by Hasofer-lind method

In HL method, the random variables are normalized to independent standard Gaussian distributions with mean value 0 and standard variance 1. The Taylor first order expansion point is at Most Probable Point (MPP) instead of the mean value point in MVFOSM.

Steps for computing β_{HL}: (More details see Choi, S.K., "Reliability-based Structural Design", 2006)

1) Compute safety index β by MVFOSM. For the first step, the computing point X* is chosen to be the mean value point μ_X

$$\beta = \frac{g(\mu_X)}{\sqrt{\sum_{i=1}^{n}\left(\frac{\partial g(\mu_X)}{\partial x_i}\sigma_{x_i}\right)^2}} \tag{4.6}$$

2) Given a point X* and corresponding U*, compute $\cos\theta_{u_i}$

$$\cos\theta_{u_i} = -\frac{\frac{\partial \hat{g}(U^*)}{\partial u_i}}{\left|\nabla \hat{g}(U^*)\right|} = -\frac{\frac{\partial g(X^*)}{\partial x_i}\sigma_{x_i}}{\sqrt{\sum_{i=1}^{n}\left(\frac{\partial g(X^*)}{\partial x_i}\sigma_{x_i}\right)^2}} \tag{4.7}$$

3) Compute the new design point U* and the corresponding X*

$$\begin{aligned} u_i^* &= \beta\cos\theta_{u_i} \\ x_i^* &= \mu_{x_i} + \sigma_{x_i} u_i^* \end{aligned} \tag{4.8}$$

4) Compute the new β at the new X* point and new U* point

$$\beta = \frac{g(X^*) - \sum_{i=1}^{n}\frac{\partial g(X^*)}{\partial x_i}\sigma_{x_i}u_i^*}{\sqrt{\sum_{i=1}^{n}\left(\frac{\partial g(X^*)}{\partial x_i}\sigma_{x_i}\right)^2}} \tag{4.9}$$

5) Repeat step 2)—4) until β converges, or if not converging then stop the loop at a given iteration number.

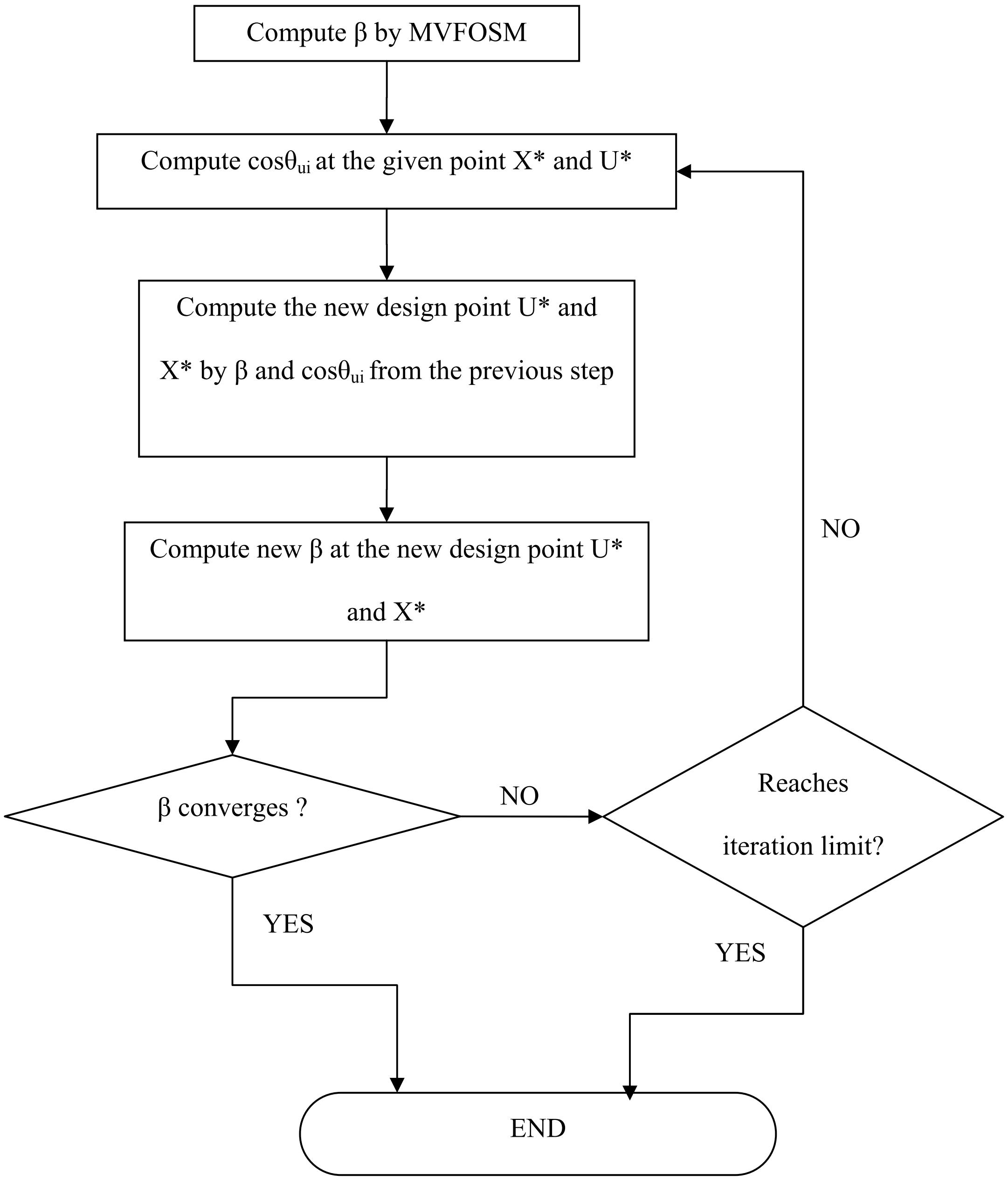
Compute β by MVFOSM
Compute $\cos\theta_{ui}$ at the given point X* and U*
Compute the new design point U* and X* by β and $\cos\theta_{ui}$ from the previous step
Compute new β at the new design point U* and X*
β converges ?
NO
Reaches iteration limit?
NO
YES
YES
END

Figure 4-4: Flow chat of HL method

Table 4-3: Iterations of HL method for rod3

iteration	1	2	3	4	5	6
beta	**2.9953E+00**	**2.2749E+00**	**1.9264E+00**	**1.8764E+00**	**1.8756E+00**	**1.8756E+00**
g(X)	3.7383E+07	-5.5392E+07	-1.4562E+07	-1.6430E+06	-2.6155E+04	
$\left.\frac{\partial g(X)}{\partial x_1}\right\|_{x_1=x_1^*}$	6.2395E+11	3.8451E+12	2.0891E+12	1.6443E+12	1.5924E+12	
$\left.\frac{\partial g(X)}{\partial x_2}\right\|_{x_2=x_2^*}$	2.8328E+10	8.5192E+10	5.9151E+10	5.1169E+10	5.0181E+10	
$\left.\frac{\partial g(X)}{\partial x_3}\right\|_{x_3=x_3^*}$	-6.2785E+09	-1.3354E+10	-1.0707E+10	-9.6877E+09	-9.5513E+09	
$\cos\theta_1$	-9.9986E-01	-9.9997E-01	-9.9995E-01	-9.9994E-01	-9.9994E-01	
$\cos\theta_2$	-1.3618E-02	-6.6465E-03	-8.4938E-03	-9.3351E-03	-9.4533E-03	
$\cos\theta_3$	1.0061E-02	3.4727E-03	5.1250E-03	5.8913E-03	5.9977E-03	
x1	1.0000E-04	4.0103E-05	5.4504E-05	6.1475E-05	6.2474E-05	6.2491E-05
x2	3.0000E-05	2.9755E-05	2.9909E-05	2.9902E-05	2.9895E-05	2.9894E-05
x3	1.0000E-04	1.0060E-04	1.0016E-04	1.0020E-04	1.0022E-04	1.0022E-04
u1	0.0000E+00	-2.9949E+00	-2.2748E+00	-1.9263E+00	-1.8763E+00	-1.8755E+00
u2	0.0000E+00	-4.0791E-02	-1.5120E-02	-1.6362E-02	-1.7516E-02	-1.7730E-02
u3	0.0000E+00	3.0136E-02	7.9000E-03	9.8726E-03	1.1054E-02	1.1249E-02

Table 4-4: Iterations of HL method for rod7

iteration	1	2	3	4	5	6
beta	**2.8889E+00**	**2.1841E+00**	**1.8761E+00**	**1.8392E+00**	**1.8388E+00**	**1.8388E+00**
g(X)	3.6299E+07	-4.8523E+07	-1.2012E+07	-1.1696E+06	-1.3444E+04	
$\left.\frac{\partial g(X)}{\partial x_1}\right\|_{x_1=x_1^*}$	-3.1393E+09	-4.6553E+09	-4.3870E+09	-4.2155E+09	-4.1930E+09	
$\left.\frac{\partial g(X)}{\partial x_2}\right\|_{x_2=x_2^*}$	4.0061E+10	1.2143E+11	8.8566E+10	7.8180E+10	7.7063E+10	
$\left.\frac{\partial g(X)}{\partial x_3}\right\|_{x_3=x_3^*}$	6.2813E+11	3.4425E+12	1.9497E+12	1.5886E+12	1.5523E+12	
$\cos\theta_1$	4.9968E-03	1.3522E-03	2.2499E-03	2.6534E-03	2.7009E-03	
$\cos\theta_2$	-1.9130E-02	-1.0582E-02	-1.3627E-02	-1.4762E-02	-1.4892E-02	
$\cos\theta_3$	-9.9980E-01	-9.9994E-01	-9.9990E-01	-9.9989E-01	-9.9989E-01	
x1	1.0000E-04	1.0029E-04	1.0006E-04	1.0008E-04	1.0010E-04	1.0010E-04
x2	3.0000E-05	2.9668E-05	2.9861E-05	2.9847E-05	2.9837E-05	2.9836E-05
x3	1.0000E-04	4.2233E-05	5.6321E-05	6.2482E-05	6.3219E-05	6.3228E-05
u1	0.0000E+00	1.4436E-02	2.9534E-03	4.2209E-03	4.8802E-03	4.9664E-03
u2	0.0000E+00	-5.5265E-02	-2.3111E-02	-2.5564E-02	-2.7152E-02	-2.7383E-02
u3	0.0000E+00	-2.8884E+00	-2.1840E+00	-1.8759E+00	-1.8390E+00	-1.8386E+00

Comparison can be made between MCS, MVFOSM and HL method in Table 4-5:

Table 4-5: comparison between MVFOSM, MCS and HL results

	MCS (1e3 smaples 50 seconds)	MVFOSM reliability index		HL reliability index	
Rod (Ai independent)	P_f	β	P_f	β_{HL}	P_f
3	0.029	2.9953	0.0014	1.8756	0.0304
7	0.028	2.8889	0.0019	1.8388	0.0330

It can be seen from Table 4-5 that the result of HL method is very close to that of MCS method. For first order reliability methods, HL method provides more accurate results than MVFOSM.

4.4.3 Second Order Reliability Method (SORM)

For rod 3, the safety index β was obtained using HL: β=1.8765. The probability of failure can be computed as:

$$P_f = \Phi(-\beta) = 0.0304$$

Compute the first order and second order derivatives of the limit state function at U* and form the B matrix:

$$\frac{\partial \hat{g}(U)}{\partial u_i} = \frac{\partial \hat{g}(U)}{\partial x_i}\sigma_i$$

$$\frac{\partial^2 \hat{g}(U)}{\partial u_i^2} = \frac{\partial^2 \hat{g}(U)}{\partial x_i^2}\sigma_i^2$$

$$\frac{\partial^2 \hat{g}(U)}{\partial u_i \partial u_j} = \frac{\partial^2 \hat{g}(U)}{\partial x_i \partial x_j}\sigma_i \sigma_j$$

$$\left|\nabla \hat{g}(U^*)\right| = \sqrt{\sum_{i=1}^{n}\left(\frac{\partial \hat{g}(U^*)}{\partial u_i}\right)^2} = 3.1833\text{e}+007$$

$$\nabla^2 \hat{g}(U^*) = \begin{bmatrix} \frac{\partial^2 \hat{g}(U^*)}{\partial u_1^2} & \frac{\partial^2 \hat{g}(U^*)}{\partial u_1 \partial u_2} & \frac{\partial^2 \hat{g}(U^*)}{\partial u_1 \partial u_3} \\ \frac{\partial^2 \hat{g}(U^*)}{\partial u_2 \partial u_1} & \frac{\partial^2 \hat{g}(U^*)}{\partial u_2^2} & \frac{\partial^2 \hat{g}(U^*)}{\partial u_2 \partial u_3} \\ \frac{\partial^2 \hat{g}(U^*)}{\partial u_3 \partial u_1} & \frac{\partial^2 \hat{g}(U^*)}{\partial u_3 \partial u_2} & \frac{\partial^2 \hat{g}(U^*)}{\partial u_3^2} \end{bmatrix}$$

$$B = \frac{\nabla^2 \hat{g}(U^*)}{|\nabla \hat{g}(U^*)|} = \begin{bmatrix} -0.6368 & -0.0037 & 0.0016 \\ -0.0037 & -0.0008 & -0.0007 \\ 0.0016 & -0.0007 & 0.0021 \end{bmatrix}$$

Orthogonal matrix H can be computed based on H0:

$$H_0 = \begin{bmatrix} -\frac{\partial \hat{g}(U^*)/\partial u_1}{|\nabla \hat{g}(U^*)|} & -\frac{\partial \hat{g}(U^*)/\partial u_2}{|\nabla \hat{g}(U^*)|} & -\frac{\partial \hat{g}(U^*)/\partial u_3}{|\nabla \hat{g}(U^*)|} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -0.9999 & -0.0095 & -0.0060 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

By Gram-Schmidt orthogonalization method, the orthogonal matrix can be computed. The Gram-Schmidt orthogonalization process is carried out by the matlab m file: Gram_Schmidt_Process.m, which takes column vectors as input, for this example the H0 matrix should first be transposed then be processed by the m function. The returned orthogonal matrix should also be transposed. Then moving the first row to the last for the orthogonal matrix, the H matrix can be obtained:

$$H = \begin{bmatrix} -0.0095 & 1 & 0.0001 \\ 0.0060 & 0 & 1 \\ -0.9999 & -0.0095 & 0.0060 \end{bmatrix}$$

Compute the main curve kj by solving the eigenvalues of HBH^T:

$$HBH^T = \begin{bmatrix} -0.0008 & -0.0007 & -0.0024 \\ -0.0007 & 0.0021 & 0.0022 \\ -0.0024 & 0.0022 & -0.6368 \end{bmatrix}$$

Eigenvalue of the first 2x2 sub-matrix can be computed as:

k1= 0;

k2=0.0022;

Probability of failure thus can be obtained:

$$P_f = \Phi(-\beta)\prod_{j=1}^{n-1}(1+k_j\beta)^{-1/2}$$
$$= \Phi(-1.8765)(1+k_1\beta)^{-1/2}(1+k_2\beta)^{-1/2}$$
$$= 0.0303$$

By the same procedure the Pf of rod 7 can also be computed:

$$P_f = 0.0329$$

A comparison between precise MCS, MVFOSM, HL method and SORM is made in Table 4-6: This table shows that both HL and SORM have high precision. MVFOSM is not suitable for limit-state functions with high nonlinearity.

Table 4-6: Comparison between MCS, MVFOSM, HL and SORM results

	Precise MCS	MVFOSM reliability index		HL reliability index		SORM Breitung
Sample size	1e4	/	/	/	/	/
Running time (s)	500	/	/	/	/	/
Rod (Ai independent)	P_f	β	P_f	β_{HL}	P_f	P_f
3	0.0330	2.9953	0.0014	1.8756	0.0304	0.0303
7	0.0322	2.8889	0.0019	1.8388	0.0330	0.0329

4.5 Summary

In this chapter, the basic reliability-based structural design and optimization method is outlined. A ten-bar-truss model is given for example and further studies.

From the example, it can be concluded that for limit state functions of large nonlinearity, MVFOSM method has greater error than HL and SORM method. HL method and SORM method can obtain more accurate results than that of MVFOSM. However due to the limitation of computation expenses, MCS takes only 10,000 samples, which is not large enough to give a precise result.

CHAPTER 5: COPULAS MODELING OF RANDOM VARIABLE CORRELATION

In this chapter, three methods for modeling random variable correlations will be introduced. First, the well known Pearson's correlation coefficient, multivariate distribution functions and KL expansion will be introduced. Then the concept of copula and its properties will be introduced. The method of producing correlated random variables will also be explained. The ten bar truss model will again be used as example for copula and other related methods. It will be explained in the example why for a same correlation coefficient different family of copula gives different simulation result. It will also be explained how different correlation effects probability of failure. In the end of this chapter, based on the example, come conclusions about the advantages and disadvantages of copula method are made.

5.1 Methods for Modeling Random Variable Correlations

5.1.1 Pearson's correlation coefficient

Pearson's correlation coefficient is defined as.

$$\rho = \frac{COV(X,Y)}{\sqrt{DX} \cdot \sqrt{DY}} \tag{5.1}$$

Pearson's correlation coefficient is a measure of linear relationship only. It's invariant to nonlinear transformation. Even if there is very strong nonlinear relationship between 2 random variables, pearson's coefficient may remain zero. Consider an example: X~Uniform on [0,1], the Pearson's coefficient between sets X and X^3 is around 90% despite the fact that the 3 variables are deterministically related. For higher nonlinearity it could be even smaller.

This is a drawback of Pearson's correlation coefficient. It will be shown later that for rank based correlation coefficient, i.e., Kendall's tau, this drawback does not exist.

5.1.2 Multivariate distribution functions

Multivariate Gaussian Distribution is one of the most well-known multivariate distributions. All the marginal distributions of Multivariate Gaussian follow single variate Gaussian distribution. For two-dimensional case, the joint PDF of Gaussian distribution is defined as:

$$f(x,y) = \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} \exp\left(-\frac{1}{2(1-\rho^2)} \left[\frac{(x-\mu_x)^2}{\sigma_x^2} + \frac{(y-\mu_y)^2}{\sigma_y^2} - \frac{2\rho(x-\mu_x)(y-\mu_y)}{\sigma_x\sigma_y} \right] \right)$$

(5.2)

(formula from Papoulis,A., “Probability, Random Variables and Stochastic Processes”, 2002)

In this case:

Expected value: $\mu = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}$

Covariance Matrix: $C = \begin{pmatrix} \sigma_x^2 & \rho\sigma_x\sigma_y \\ \rho\sigma_x\sigma_y & \sigma_y^2 \end{pmatrix}$

ρ is the correlation coefficient, it indicate the linear dependency of the 2 marginal distributions. When ρ equals zero, both marginal distributions are independent to each other. Illustrated in Figure 5-1 are 2 PDF plot of bivariate Gaussian for ρ=0 and ρ=0.8 respectively.

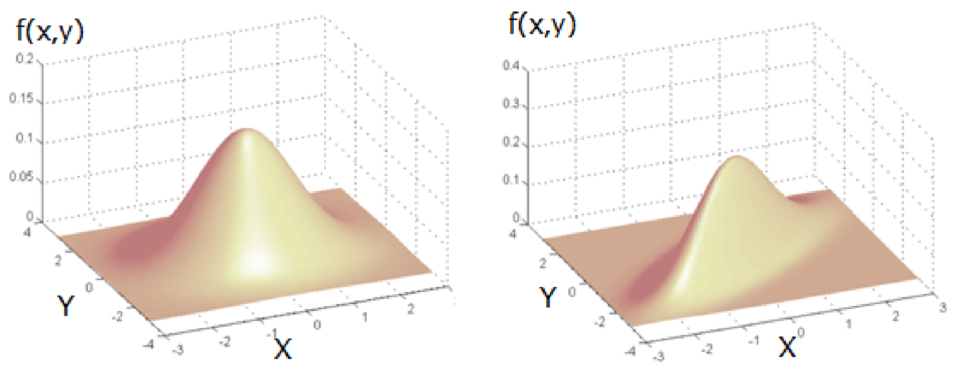

Figure 5-1: PDF plot of bivariate Gaussian for =0 and =0.8 respectively

5.1.3 Karhunen-Loeve expansion

The covariance between random variables can be determined by the distance between them. Covariance function is a method to determine covariance between 2 points by the distance between them. The covariance function need satisfy the requirement that with the distance increasing, the covariance would decrease. There are 3 well-known covariance functions [Choi, S.K., 2006] explained below.

1) Exponential model

The exponential covariance function decays exponentially with increasing distance:

$$C_{ij} = C_0 \exp\left[-\left| \frac{x_{ij}}{l} \right| \right] \tag{5.3}$$

Where C_0 is the variance, x_{ij} is the distance between two points, l is the correlation length. The covariance between 2 points drops rapidly with the increase of distance.

2) Gaussian model

The Gaussian model is a another widely used covariance function:

$$C_{ij} = C_0 \exp\left[-\left(\frac{x_{ij}}{l} \right)^2 \right] \tag{5.4}$$

Gaussian model is similar to exponential model, however it has the shape of a Gaussian PDF function. The difference with exponential model is that at the origin the shape is flat. The covariance decays very rapidly as distance is greater than l .

3) Nugget-effect model

Nugget-effect model is an uncorrelated case:

$$C_{ij} = \begin{cases} C_0 : & \mathrm{i} = \mathrm{j} \\ 0 : & \mathrm{i} \neq \mathrm{j} \end{cases}$$

Karhunen-Loeve expansion can be viewed as a general orthogonal series expansion of the covariance function [Papoulis, A., Pillai, S.U., 2002] [Choi, S.K., 2006]. KL expansion can be expressed as:

$$w(x) = \sum_{i=1}^{\infty} \sqrt{\lambda_i} \xi_i \phi_i(x) \tag{5.5}$$

Where ξ_i is a set of random variables, λ_i is the eigenvalue and $\phi_i(x)$ is the eigenfunction of the covariance function. $w(x)$ generated in this way will satisfy the covariance function $K(x)$

In the discrete case, KL expansion can be expressed as:

$$[P][\Lambda] = [K][P] \tag{5.6}$$

Where [K] is a covariance matrix, [P] is the orthogonal eigenvector matrix; $[\Lambda]$ is the eigenvelue matrix.

Covariance matrix can be decomposed as:

$$[K] = [P][\Lambda][P]^T = [A] \cdot [A] \tag{5.7}$$

Where $[A]$ is the transform matrix defined as $[A]=[P][\Lambda]^{1/2}$. The [A] matrix can be used to produce correlated random variables:

$$[Y]=[A][X]$$

Where $[X]$ is a matrix of random vectors X_i, $[Y]$ is a matrix of column vectors that possess the covariance matrix $[K]$

KL expansion can be used to find the principal components of a transformation, it thus is also called Principal Component Analysis. To reduce the data dimension, only several principal components can be kept while minor components can be discarded, thus the data dimension can be reduces significantly.

Figure 5-2 is an illustration of how to reduce dimension of random variables by KL transform. For the convenience of description, from the upper left to lower right, the figures are numbered consecutively from (1) to (5).

Figure (1) is the uncorrelated Gaussian random variable case, based on which KL transform is taken. Generated correlated random variables are shown in figure (2), rho=0.9. Figure (3) shows the coordinates of points in figure (2) projected on eigenvectors. To reduce the dimension, the minor coordinate values are discarded as in figure (4); Figure (5) shows the result when projected back to the original coordinate system. The result of these steps is the reduction of dimension of random vectors. For multidimensional problems, KL transform can reduce random vector dimensions significantly.

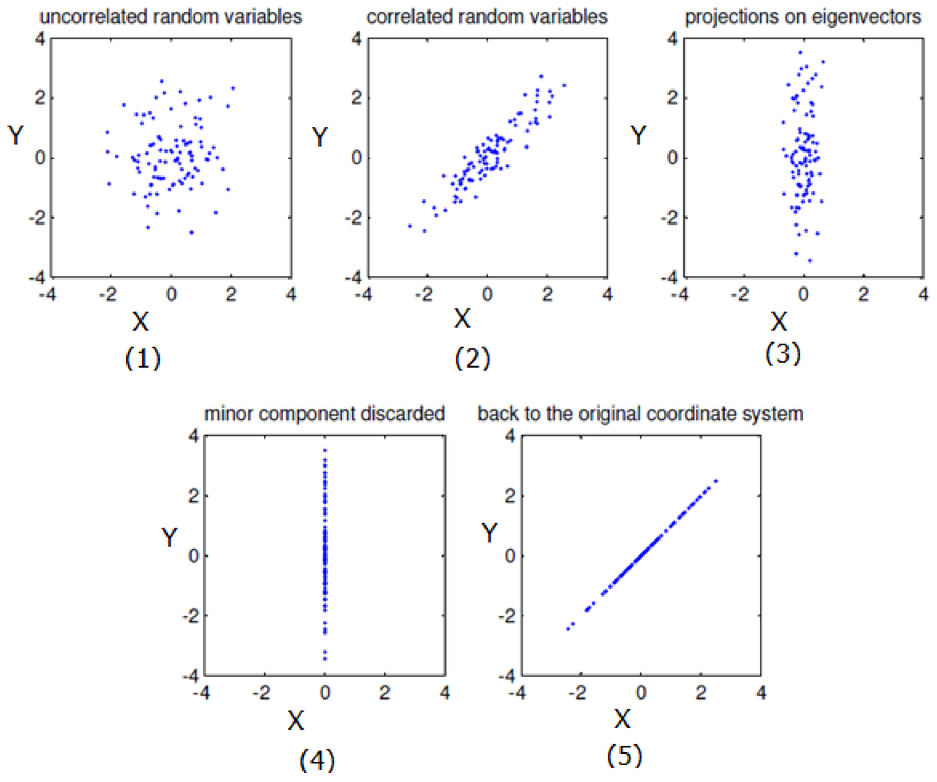

Figure 5-2: Example of using KL expansion to reduce data dimensions

5.2 Copula Review

The word “copula” is a Latin word that means “bond, connection” [“World English Dictionary”]. It was first employed as a statistic term by Abe Skla (1959) to describe the correlation between random variables and to link their marginal distributions with their joint distribution. Copula was most early studied by Wassily Hoeffding in the 1940’s. It has been discovered independently by various authors, such as Frechet, Sklar, Kimeldorf and Sampson by various terms [Nelsen, R.B. 1999].

Some of the famous authors about copulas today include: Nelsen, R.B.; Genest, C.; Frees, E.; Joe H. These authors are mainly mathematicians. There are many other authors who apply copulas in their respective research areas, especially in actuarial studies and stock market studies.

5.2.1 Literature Review

[Nelsen,R.B., 1999]: This book introduces the basic concept of copulas, the method for constructing copulas, the concept and construction method of Archimedean families of copulas, and measure of concordance, ie, Kendall's tau and Spearman's Rho. This book is a good introduction book for copulas from mathematical perspective.

[Genest, C., 1987]: Properties of Frank's copula are investigated in this paper. Nonparametric estimation methods for rank dependence parameters are suggested.

[Genest, C., 1993]: This paper examines the problem of selecting an Archimedean copula providing a suitable representation of the dependence structure between two variables X and Y by studying a set of random samples. The estimation method of copula coefficient is based on the empirical form of Kendall's tau. For different copulas to fit the same data sample, the method for examine the goodness of fit is provided to help finding the best fit copula. Using a uranium exploration dataset, the paper illustrates how to use the procedure to choose a copula.

[Genest,C., 2007]: This paper introduces copula based on rank methods. By working out a small numerical example, the writer exhibits the steps involved in investigating the dependence between 2 random variables and in modeling it using copulas. Methods are provided for selecting an appropriate copula model and parameter estimation, as well as checking its goodness-of-fit.

[Frees,E.W., et al, 1998]: This article introduces actuaries to the concept of copulas. Basic concept and properties are introduced in this article, some potential application in insurance business such as estimation of joint life mortality are explored. An annotated bibliography is provided for readers to continue their study of copulas.

[Genest, C., MarcKay, J., 1986]: In this paper, the concept of copula is introduced. Many of the basic properties is derived by element calculus. This paper shows how copulas can be used to illustrate the existence of distributions with singular components and to give a geometric interpretation to Kendall's tau.

[Abbas, A.E., 2009]: In this paper, the author defines the concept of “Utility Copula”, a comparison was made between utility copula and Sklar’s copula. Archimedean copula is defined as a simple mechanism for generating a large family of utility copulas.

5.2.2 Normalized Form of Correlated Random Variables

5.2.2.1 Concept of dependence and example of dependent bivariate random variables generation

Multivariate Gaussian distribution provides a good example of bivariate independence and dependence relations. Suppose two models, in model 1 the two random variables are independent, in model 2 the 2 random variables are dependent to each other with linear correlation coefficient rho equals to 0.8. Example scatter plots of the two models are shown in Figure 5-3:

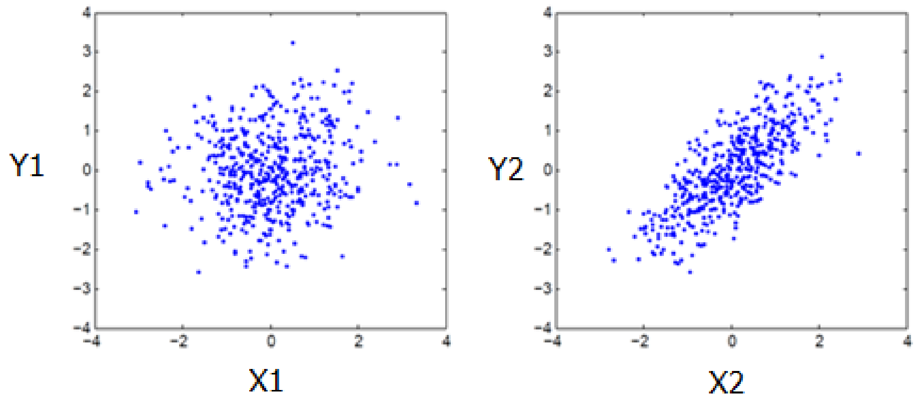

Figure 5-3: Scatter plots of Independent and Dependent Bivariate Gaussian Distributions

Figure 5-3 shows that in the correlated model there is a trend that large values of x tend to appear together with large values of y, while in the independence model such trend does not exist.

If exponential is taken for the above variables, the rank relation is preserved in the transformed variables since the exponential function is monotone increasing. The scatter plot of the transformed variables of the above 2 models are shown in Figure 5-4:

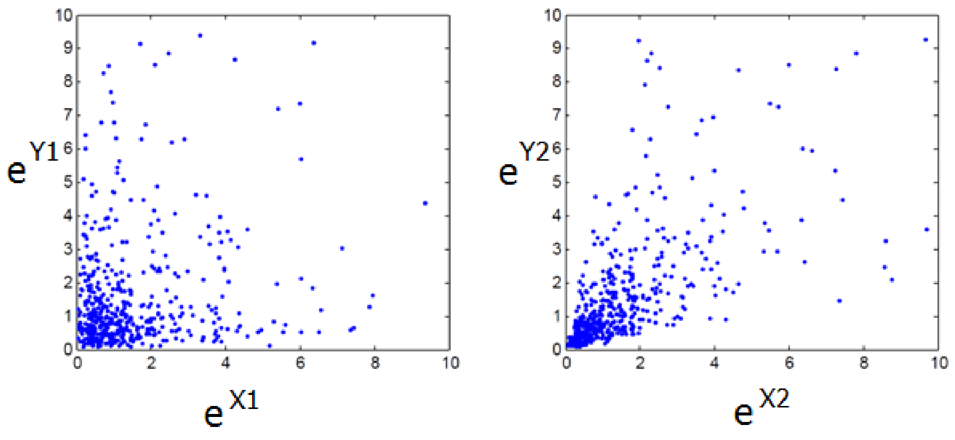

Figure 5-4: Taking exponential for independent and dependent Gaussian distribution

It's obvious that in the right figure the transformed variables still keep the rank relations, though the correlation coefficient rho might not be the same. (App. Code 32)

It should be noted that not only exponential function could keep the rank relations, any monotone increasing function could keep this correlation structure. The monotone function for the two random variables could be the same or not the same as long as they are strictly increasing. The correlation structure can be expressed by rank correlation coefficient, ie, Kendall's tau.

5.2.2.2 Correlated random variables generation by bivariate Gaussian distribution

The method in the previous section shows that a pair of correlated bivariate Gaussian distribution variables can be served as a tool for generating further correlated random variables that are not necessarily confined to bivariate Gaussian. This method provides only a special case for correlated random variable generation. The drawback of this method is that the marginal distributions are not controllable. To find a more general method to produce correlated random variables with any desired marginal distribution, CDF transformation could be used. From the correlated random variable example in the previous section, taking CDF transform of monovariate Gaussian distribution for each of the two random variables, the scatter plot of the original random variables X and the scatter plot of CDF transformed random variables U are shown in Figure 5-5 on the upper row; the histogram of U are shown in the lower row, it is shown that U follows uniform distribution: (App. Code 33)

Although the transformed plot looks different, the rank correlation of the original scatter plot is precisely preserved due to the fact that CDF function and inverse CDF function are strictly increasing. Since the transformed form by CDF transformation is in the range of [0,1] and has uniform marginal distribution, it can be regarded as the normalized form of the rank correlation of the original correlated random variables.

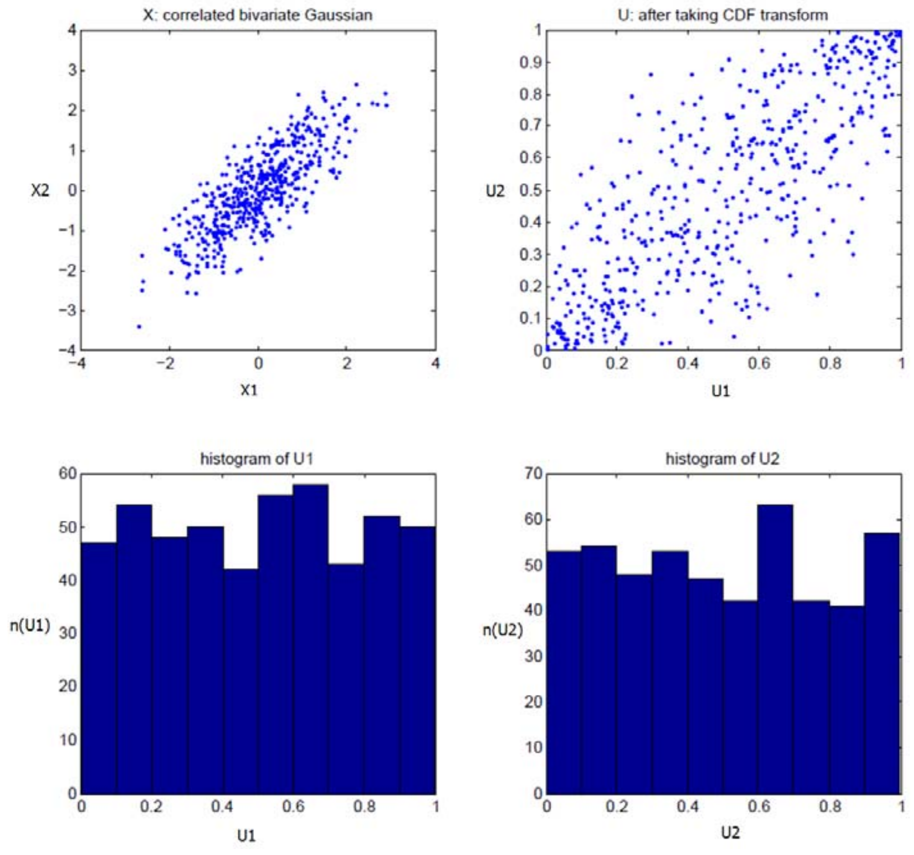

Figure 5-5: Upper: Illustration of bivariate Gaussian before and after CDF transform; Lower: histogram in two dimensions after CDF transform

If the above normalized random variables U is used as input to an inverse CDF transform of any distribution, the generated points would still keep the rank correlation information since the inverse CDF transform in the is strictly increasing. As an example, take inverse CDF transform on U generated in the previous step, one dimension as Gamma distribution and another as t distribution. The transformed scatter plot is shown in Figure 5-6. The left figure is for the normalized random variables U, the right figure is the result after inverse CDF transform.

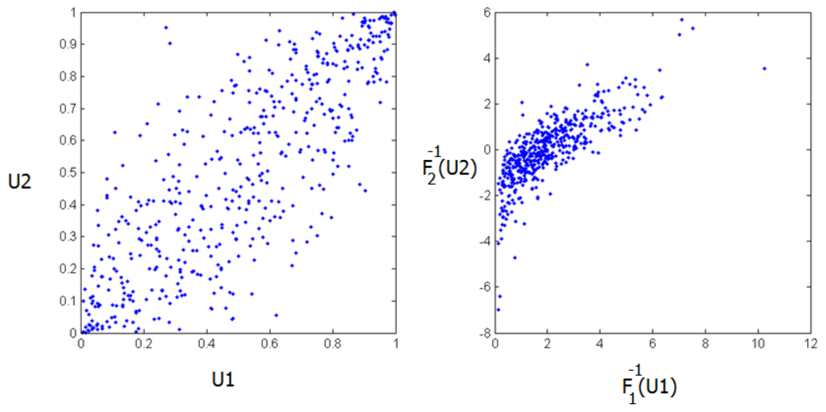

Figure 5-6: Normalized random variables before and after inverse CDF transform (X: Gamma; Y: t)

If both inverse CDF transforms are based on Gamma distribution, the scatter plot is shown in Figure 5-7

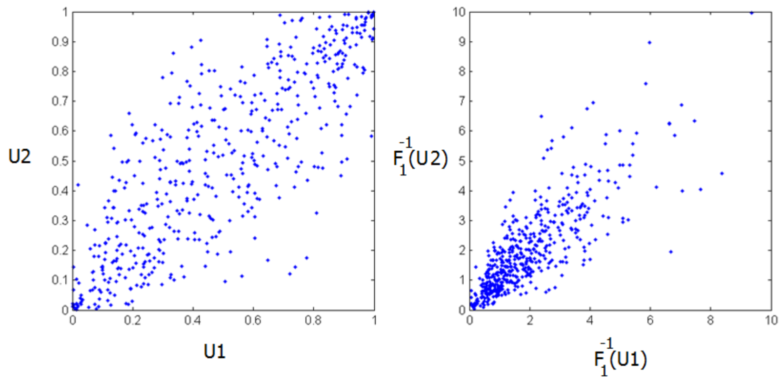

Figure 5-7: Normalized random variables before and after inverse CDF transform (X: Gamma; Y: Gamma)

If it's needed to generate correlated random variables with particular marginal distributions, just change the inverse CDF transformation to that of the desired distributions. The rank correlation information will be preserved by this procedure no matter what particular marginal distribution it is.

Method introduced in this section is based on multivariate Gaussian distribution and the related CDF transform.

5.2.2.3 Correlated random variables generation by bivariate t-distribution

Just like the multivariate Gaussian method discussed in the previous sub section to generate correlated random variables of any marginal distribution, multivariate t-distribution can also be served for the same purpose. The first step is to generate correlated multivariate t-distribution points; the second step is to make CDF transform to get normalized rank correlation form that has the uniform marginal distributions on [0,1] while keeping the rank correlation information of the original multivariate t-distribution; The 3rd step is to generate any desired distribution data samples by taking inverse CDF transform on the uniform distributed marginals of the normalized form, the generated data will still keep the rank correlation information as in the normalized form.

Illustrated in Figure 5-8 is an example of the comparison between random variable generation using Gaussian distribution method and t-distribution method. The 1st row is Gaussian, 2nd row is t, the column from left to right are: correlated Gaussian/t distribution; normalized form; generated correlated data with marginals be Gamma and t respectively. It should be noted that although the two methods used the same linear dependence coefficient rho=0.8, they show some difference this is because of the different correlation structure between Gaussian copula and t copula.

5.2.3 Copula Properties and Applications

As explained in previous sections, in step 2 a set of normalized random variables that has the uniform marginal distribution on [0,1] is obtained. These normalized variables represent rank relations and dependence structure, however it does not impose any requirement on the marginal distributions. As a fact any marginal distribution can be implemented based on the given normalized random variables by inverse CDF transform. It can be concluded that this method decouples the marginal distributions with their correlation. The relationship between the normalized uniform distributed random variables is copula.

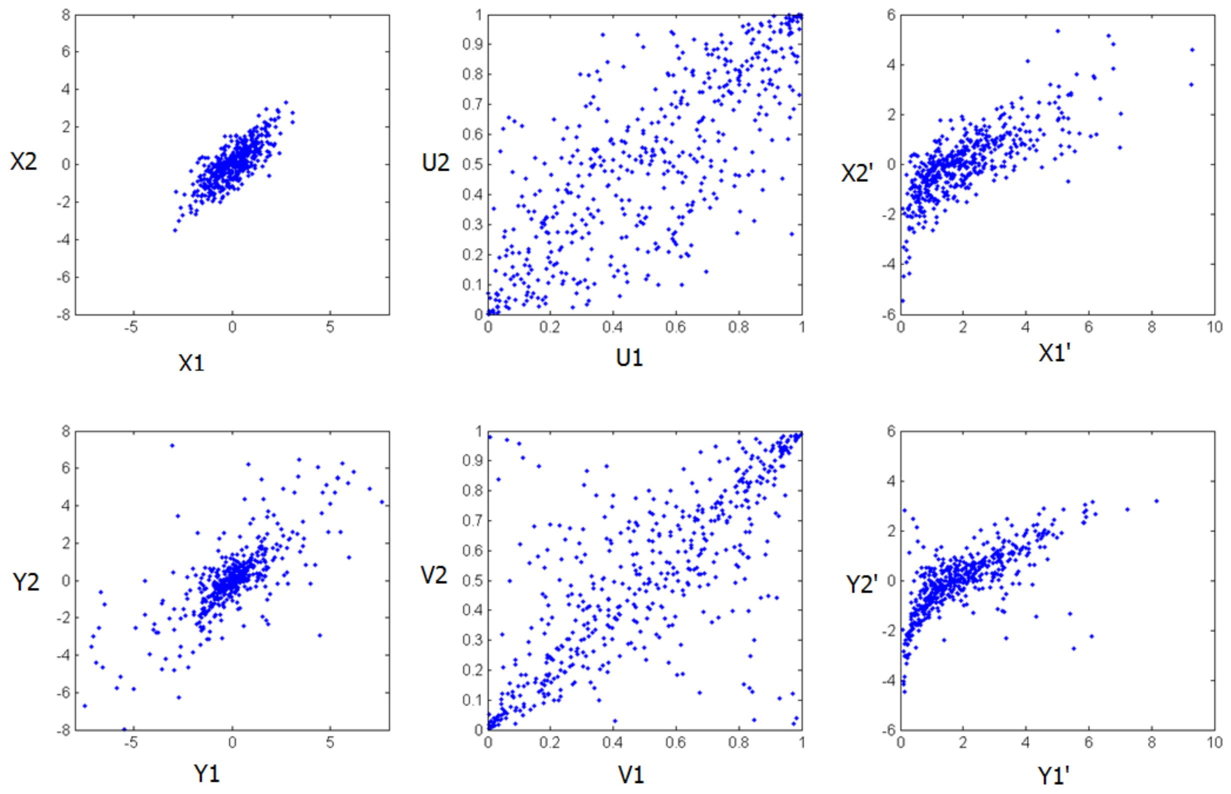

Figure 5-8: Using bivariate Gaussian and t distribution to generate normalized random variables and by which to produce any desired marginal distributions

Copula is formally defined as a joint distribution function whose marginal distributions are uniformly distributed on [0,1] [Nelsen, 1999]. Uniformly distributed marginals provide a potential for any inverse CDF transform that can be used to produce desired marginal distributions similar to the method used by Monte Carlo Simulation. The coming part of this section will give a brief conclusion of some most important copula properties, detailed proof can be found in [Genest 1993, 2007].

5.2.3.1 Properties of copula

1) Frechet-Hoeffding's bounds

   For every C and (u,v) in $[0,1]^2$, $W(u,v) \leqslant C(u,v) \leqslant M(u,v)$

   in which M(u,v)=min(u,v); W(u,v)=max(u+v-1,0)

   This relation can be illustrated in Figure 5-9. Left figure is the upper bound; right one is the lower bound.

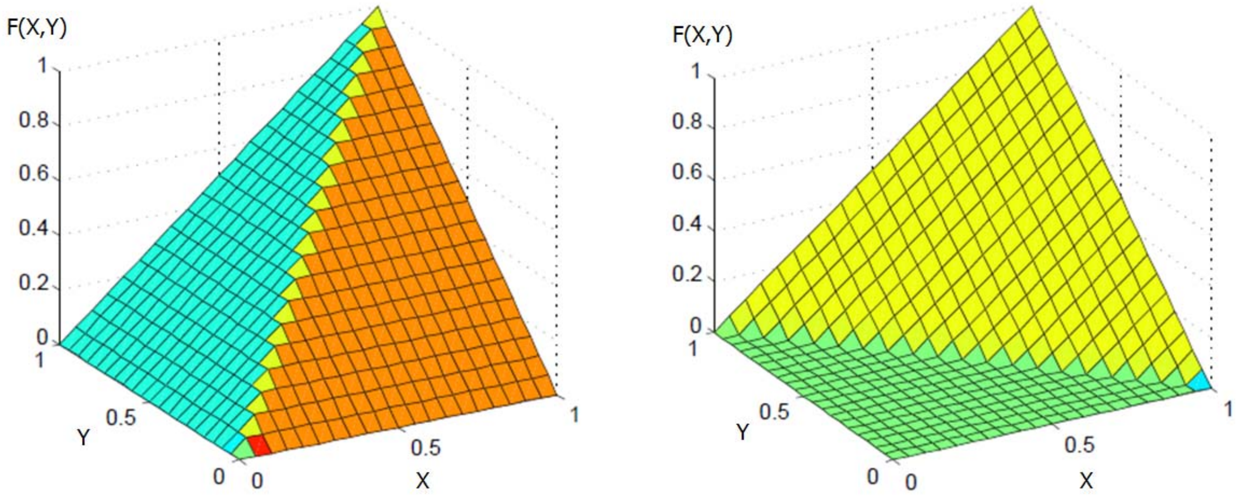

Figure 5-9: Upper and lower boundaries of copula

2) Copulas for monotone functions

   if f and g are strictly increasing then

$$C_{f(x),g(x)}(u,v) = C_{X,Y}(u,v) \tag{5.8}$$

   This property shows that random variables after strictly increasing transform and the copula remains the same. This property can be easily proved in this way:

   Suppose the random variables and their distribution functions are:

Table 5-1: Random variables and their respective CDF functions

Random Variables	CDF
X	$F_1(x)$
Y	$G_1(y)$
$\alpha(X)$	$F_2(x)$
$\beta(Y)$	$G_2(y)$

$$\begin{aligned} &C_{\alpha(X),\beta(Y)}\left(F_2(x),G_2(y)\right)=\Pr\{\alpha(X)\leq x,\beta(Y)\leq y\} \\ &=\Pr\left\{X\leq\alpha^{-1}(X),Y\leq\beta^{-1}(Y)\right\} \\ &=C_{X,Y}\left(F_1\left(\alpha^{-1}(x)\right),G_1\left(\beta^{-1}(y)\right)\right) \end{aligned} \tag{5.9}$$

Since

$$F_1\left(\alpha^{-1}(x)\right)=\Pr\left[X\leq\alpha^{-1}(x)\right]=\Pr\left[\alpha(X)\leq x\right]=F_2(x) \tag{5.10}$$

Then:

$$C_{f(x),g(x)}(u,v)=C_{X,Y}(u,v)$$

5.2.3.2 Archimedean Copula

In previous examples Gaussian copula and t-copula have been discussed, while Archimedean is also a widely used copula due to its ease of construction [Nelsen, 1999].

Archimedean copula is defined as C(u,v)=□$^{[-1]}$(□(u)+ □(v)), in which $\varphi^{[-1]}$(x) is the psudo-inverse of the function φ(x) and □(x) is the generator function that is continuous and strictly decreasing function from [0,1] to [0,+∞) with □(1)=0. Different generator functions correspond to different Archimedean families of copulas. Listed in Table 5-2 are several commonly used Archimedean families of copulas.

Figure 5-10 are scatter plots of Clayton, Frank and Gumbel copulas with 1000 sample points and Kendell's tau equals to 0.7.

Table 5-2: Selected Archimedean copulas and their respective generator functions (content partly from Nelsen, 1999 and Frees, E.W., 1998)

	Generator □(t)	θ range	copula
Clayton	$\frac{1}{\theta}(t^{-\theta}-1)$	[-1,+∞)\{0}	$\left[max\left(u^{-\theta}+v^{-\theta}-1,0\right)\right]^{-1/\theta}$
Frank	$-\ln\frac{e^{-\theta t}-1}{e^{-\theta}-1}$	(-∞,+∞)\{0}	$-\frac{1}{\theta}\ln\left(1+\frac{(e^{-\theta u}-1)(e^{-\theta v}-1)}{e^{-\theta}-1}\right)$
Gumbel-Hougaard	$(-\ln t)^{\theta}$	[1, +∞)	$\exp\left(-\left[(-\ln u)^{\theta}+(-\ln v)^{\theta}\right]^{1/\theta}\right)$

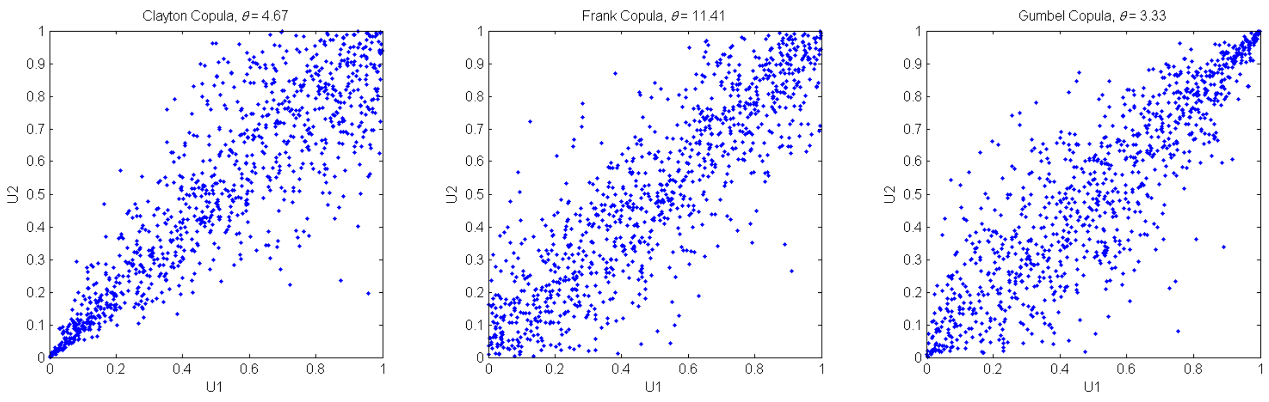

Figure 5-10: Scatter plots of Clayton, Frank and Gumbel copulas (n=1000, τ=0.7)

The relationship between copula CDF and copula PDF can be expressed as:

$$c_{X_1,X_2}(u,v)=\frac{\partial^2 C_{X_1,X_2}(u,v)}{\partial u\partial v} \qquad (5.11)$$

The PDF and CDF of Gumbel-Hougaard copula are shown in Figure 5-11, the copula coefficient θ=2:

The joint PDF of random variables is expressed by copula PDF and marginal PDF:

$$\begin{aligned} f_{X_1,X_2}(x_1,x_2) &= \frac{\partial^2 F_{X_1,X_2}(x_1,x_2)}{\partial x_1 \partial x_2} \\ &= \frac{\partial^2 C_{X_1,X_2}(u,v)}{\partial u \partial v} \cdot \frac{\partial u}{\partial x_1} \cdot \frac{\partial v}{\partial x_2} \\ &= c_{X_1,X_2}(u,v) \cdot f_{X_1}(x_1) \cdot f_{X_2}(x_2) \end{aligned} \tag{5.12}$$

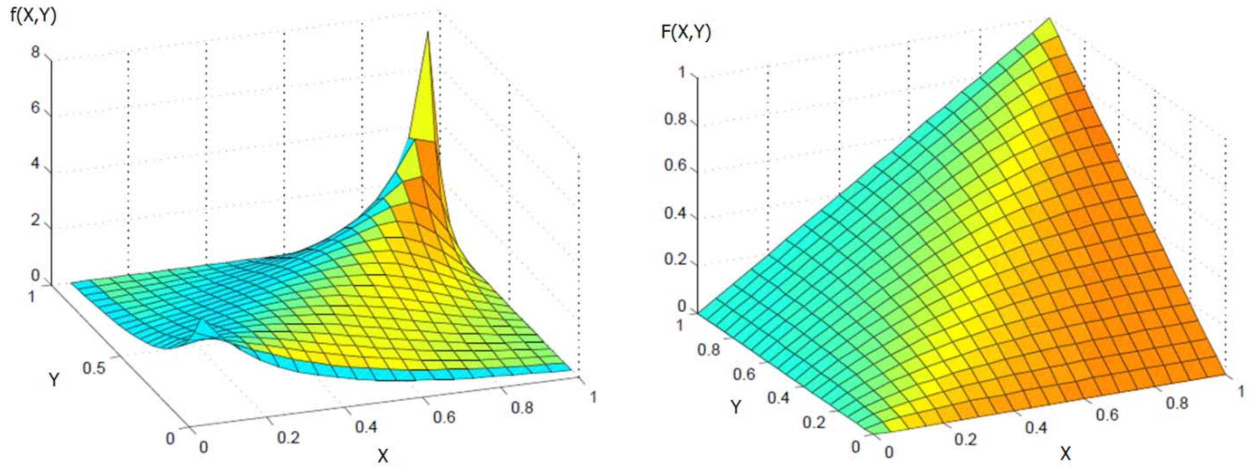

Figure 5-11: PDF and CDF of Gumbel-Hougaard copula (θ=2)

5.2.3.3 Rank correlation coefficient

Rank correlation measures the degree of concordance between 2 random variables. Concordance is a trend that big values of X and big values of Y tend to appear in pairs, small values of X and small values of Y also tend to appear in pairs. Figure 5-12 illustrates the concept of concordance and discordance.

Rank correlation coefficient is defined on [-1, 1], 1 denotes deterministic positive concordance while -1 denotes deterministic negative concordance. If X and Y are uncorrelated then their rank correlation coefficient is zero, but the converse does not hold.

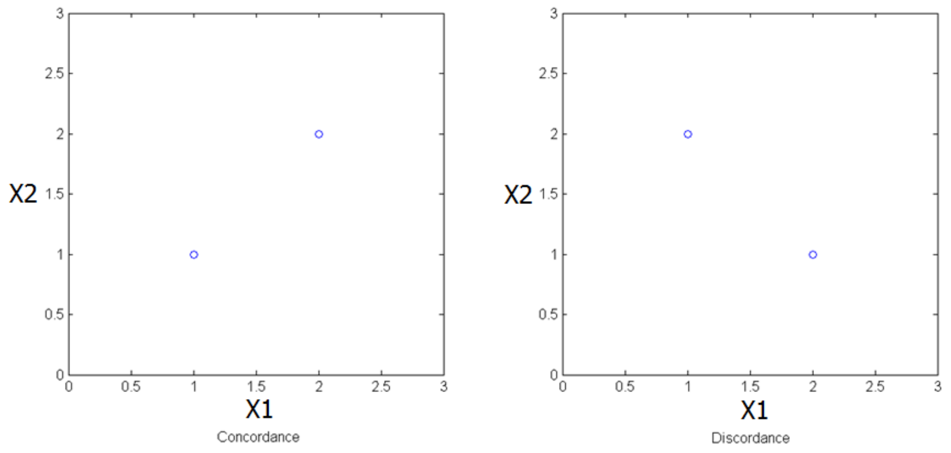

Figure 5-12: Illustration of the concept of concordance and discordance

Kendall's tau and Spearman's rho are two commonly used rank correlation coefficients.Kendall's tau is defined as the probability of concordance minus the probability of discordance [Genest, C. 2007]:

$$\tau_{XY} = \Pr\{(X_1 - X_2)(Y_1 - Y_2) > 0\} - \Pr\{(X_1 - X_2)(Y_1 - Y_2) < 0\} \qquad (5.13)$$

Kendall's tau is different from Pearson's correlation coefficient for its scale-freeness. Kendall's tau is a measure of rank correlation, regardless of whether the random variables have a linear dependence or not. As an example, consider a deterministic nonlinear relationship between 2 random variables: $Y=X^2$. In this case Kendall's tau is 1, but Pearson's correlation coefficient is 0.968 rather than 1 even the relationship is deterministic. This example shows the advantage of Kendall's tau over Pearson's rho.

Although Kendall's tau and linear dependence coefficient are different, they have a deterministic relationship for each specific copula. More details can be found in [Genest, C., Favre, A.C., 2007].

5.2.3.4 Estimating Kendall's tau by experiment

Rank correlation coefficient Kendall's tau can be estimated by obtaining a set of sample data from experiments. This process can be found in [Genest, C. and Rivest, L.P. 1993] and [Frees, E.W. 1998]

5.2.3.5 Generating correlated bivariate dataset based on copula

To estimate structural reliability by Monte Carlo Method, it's necessary to consider the correlations of random variables. For example, it's natural to model wind load and wave load as correlated random variables. Taking the correlation into account is very important in reliability evaluation. Fail to do so may result in the failure of design. In this chapter the impact of load correlation on structural strength will be shown.

Procedure for generating bivariate correlated random dataset by copula is listed below, more details can be found in [Nelsen, R.B., 1999].

1) Select a copula, say, Frank's family;

2) Generate a uniformly distributed random variable $u_0 \in [0,1]$, plug u_0 in to this derivative expression to get a function of v:

$$c = \left.\frac{\partial C}{\partial u}\right|_{u=u_0} = c(v) \tag{5.14}$$

3) Generate a uniformly distributed random variable $w \in [0,1]$, let $c(v) = w$ then v can be computed as $v = v_0$

4) Make inverse CDF transform for v_0 and u_0 and a pair of correlated random variables are obtained:

$$\begin{aligned} x_1 &= F_{X_1}^{-1}(u_0) \\ x_2 &= F_{X_2}^{-1}(v_0) \end{aligned} \tag{5.15}$$

5.3 Illustrated Examples

The ten bar truss model with two random variables will be used as an example for this chapter.

5.3.1 Uncorrelated Monte Carlo Simulation to Evaluate Pf

Matlab code refer to App. Code03. Random data are generated as independent variables that follow Gaussian distributions. For 1e5 samples, it takes 12 seconds.

Sample number: 1e5;

Running time: 12 seconds;

Pf: 0.003869;

5.3.2 Correlated MCS to Evaluate Pf by Bivariate Gaussian Distribution

Matlab code refer to App. Code 04.

The covariance matrix for the bivariate Gaussian distribution is defined as :

$$C = \sigma^2 \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix} = 500^2 \begin{bmatrix} 1 & 0.6 \\ 0.6 & 1 \end{bmatrix}$$

Sample number: 1e5;

Running time: 12 seconds;

Pf: 0.017161;

5.3.3 Correlated MCS to Evaluate Pf by KL expansion

Matlab code refer to App. Code 06.

The covariance matrix for the bivariate Gaussian distribution is also defined as :

$$C = \sigma^2 \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix} = 500^2 \begin{bmatrix} 1 & 0.6 \\ 0.6 & 1 \end{bmatrix}$$

Sample number: 1e5;

Running time: 12 seconds;

Pf: 0.01746;

Some data samples generated by KL expansion are listed in Table 5-3, to get an intuitive idea of correlation, a modified dataset P-μ(P) is also given. The trend of correlation can be obviously observed from these samples.

5.3.4 Correlated MCS to Evaluate Pf by Copula

In this section Pf will be evaluated by MCS, the data samples are generated by the method of copula. This experiment will be based on 3 different groups of data samples that are generated by 3 different families of copulas that all belong to a same family of Archimedean copula. The method of generating correlated data samples by copula will be demonstrated; the different simulation results from the 3 different copulas will also be shown and the reason will be discussed.

Table 5-3: Data samples generated by KL transform for simulation

P1	P2	P1-μ(P1)	P2-μ(P2)
2955.984125	2878.71653	-44.01587518	-121.2834704
3253.471998	3630.153653	253.4719983	630.1536527
2668.317816	2632.830431	-331.6821843	-367.1695692
2460.629117	2694.549808	-539.370883	-305.4501919
2358.502215	2887.405016	-641.4977852	-112.5949839
2367.28159	2680.568469	-632.7184102	-319.4315312
3430.533565	3615.73031	430.5335652	615.7303099
2615.471435	2948.195317	-384.5285646	-51.80468349
2697.920465	3132.002761	-302.0795354	132.0027606
3502.29927	2755.373606	502.29927	-244.6263938
2531.618788	2632.872126	-468.3812121	-367.1278744
2776.705405	3116.449212	-223.2945947	116.4492115
3079.565125	3185.210901	79.56512475	185.2109013
3924.162462	3055.823941	924.1624619	55.82394079
3421.053904	2946.906724	421.0539042	-53.09327572
2697.682987	2883.045385	-302.3170125	-116.954615

Copula is a bivariate cumulative distribution function (CDF) whose marginal distributions are all uniform distributions on [0,1]. The marginal distributions of copula can be mapped to the CDF of any univariate distributions. In this way, copula could realize the uncoupling of 2 or more correlated random variables.

The mathematical form of a bivariate copula can be expressed as:

$$C_{X,Y}(u,v) = C_{X,Y}(F_X(x), F_Y(y)) = F_{X,Y}(x,y) \tag{5.16}$$

The joint PDF of this 2 random variable can be expressed as:

$$f_{X,Y}(x,y) = \frac{\partial^2 F(x,y)}{\partial x \partial y} = \frac{\partial^2 C(u,v)}{\partial u \partial v} . \frac{\partial u}{\partial x} . \frac{\partial v}{\partial y} = \frac{\partial^2 C(u,v)}{\partial u \partial v} . f_X(x) . f_Y(y) \tag{5.17}$$

In this expression of joint PDF, $f_X(x)$, $f_Y(y)$ are marginal PDF's, while $\frac{\partial^2 C(u,v)}{\partial u \partial v}$ can be regarded as a correlation term.

To make comparisons, it is still required that the linear dependence coefficient to be the same as that of the previous experiment. The models in the previous experiments are bivariate normal, for bivariate normal distributions with linear dependent coefficient r, Kruskal has proved that

$$\tau = \frac{2}{\pi} \arcsin(r)$$

In this case linear dependent coefficient r=0.6, corresponding Kendall's τ=0.4097.

5.3.4.1 Different copula families

Correlated random data samples will be produced by three copulas (Clayton, Gumbel and Frank) that all belong to a same Archimedean family. All the three copulas will have a same Kendall's tau that equals to 0.4097.

Matlab code refer to App. Code 08

Experiment 1 Clayton copula to generate correlated RV's and MCS to get Pf:

Sample number: 1e5;

Running time: 12 seconds;

Pf: 0.00756;

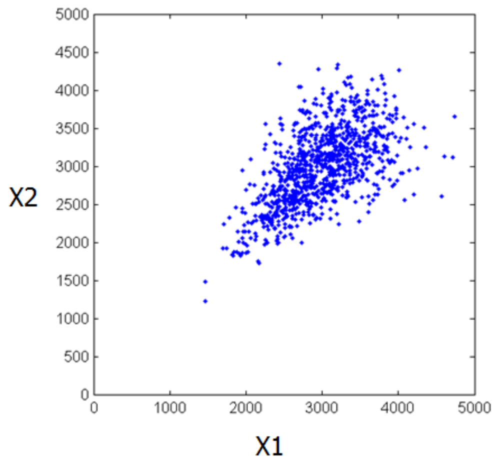

Figure 5-13: Scatter plot of Clayton copula random variables

Experiment 2 Gumbel copula to generate correlated RV's and MCS to get Pf:

Sample number: 1e5;

Running time: 12 seconds;

Pf: 0.02166;

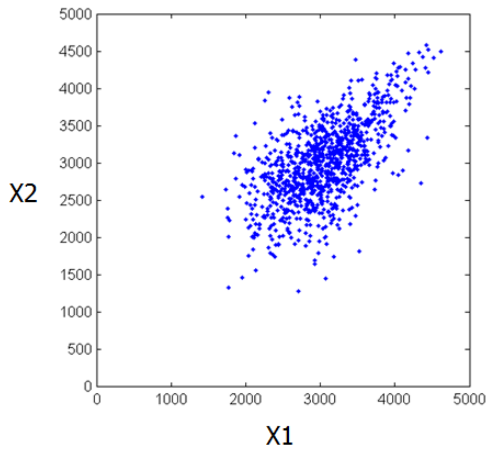

Figure 5-14: Scatter plot of Gumble copula random variables

Experiment 3 Frank copula to generate correlated RV's and MCS to get Pf:

Sample number: 1e5;

Running time: 12 seconds;

Pf: 0.01154;

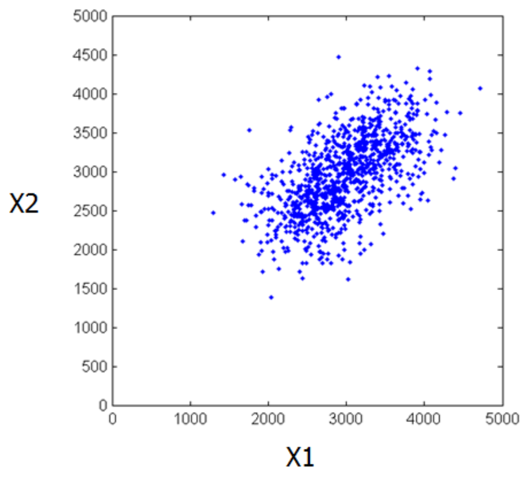

Figure 5-15: Scatter plot of Frank copula random variables

The simulation results show that although the three families of copulas (Clayton, Gumbel, Frank) have the same rank coefficient Kendall's tau which equals to 0.6 in this case, however the simulations results in 3 different value of Pf. This is because that Kendall's tau is a measure of rank only, or in another word, it measures the relative greatness between RV's only, no matter the actual value of RV's. This can be explained by this example:

Suppose there are only 2 points P1 and P2 on a copula scatter plot:

Case1: P1: (0.2 0.2); P2: (0.3 0.3)

Case2: P1: (0.2 0.2); P2: (0.6 0.6)

Their marginal distributions are both standard normal. The corresponding marginal points can be obtained by inverse Gaussian CDF transform, as shown in Table 5-4:

Table 5-4: Inverse Gaussian CDF table

Probability	Inverse Normal CDF
0.2	-0.8416
0.3	-0.5244
0.6	0.2533

For case 1 and case 2, the mean values on the marginal are respectively:

Q1=(-0.683, -0.683)

Q2=(-0.29415, -0.29415)

Although the mean values of the 2 cases are different, their measures of rank, or Kendall's tau, are the same according to the definition of Kendall's tau:

$$\tau = \Pr((x_1 - x_2)(y_1 - y_2) > 0) - \Pr((x_1 - x_2)(y_1 - y_2) < 0)$$

For this example in both cases since only concordant pairs exist, thus in both cases tau=1. This example shows that although with the same value of Kendall's tau, the probabilistic properties may be different. That explains why for three different families of copulas (Clayton, Gumbel, Frank) with the same Kendall's tau, the probabilities of failure are different.

The results of the above simulations are listed in Table 5-5:

σ_{allow}=100MPa;

$P_i \sim N(3000, 500^2)$, i=1,2

Table 5-5: Comparison of simulation results of different families of copula

Simulation Method		Linear dependent coefficient ρ	Kendall's tau	Simulation samples	Results (P_f)
Uncorrelated MCS		0	/	1e6	0.003869
Bivariate Gaussian		0.6	/	1e6	0.017161
KL expansion		0.6	/	1e6	0.01746
Positive correlated copula	Clayton copula	/	0.409666	1e5	0.00756
	Gumbel copula	/	0.409666	1e5	0.02166
	Frank copula	/	0.409666	1e5	0.01154
Negative correlated copula	Clayton copula	/	/	/	/
	Gumbel copula	/	/	/	/
	Frank copula	/	-0.409666	1e5	0.00042

It should be noted that the value of Kendall's tau is derived from the relationship of values between correlation coefficient ρ and Kendall's tau for bivariate Gaussian distribution, however in copula based simulations there's no implementation of bivariate Gaussian, thus the comparison between copula and non-copula cases are not necessarily meaningful.

5.3.4.2 Generate correlated random variables by copula

Gumbel-Hougaard copula will be taken as an example, for other copulas just substituting the Gumbel-Hougaard copula with other copulas and the steps remain the sames.

The Gumbel-Hougaard copula has this form [Nelsen 1999]:

$$C_\theta(u,v) = \exp\left\{-\left[(-\ln u)^\theta + (-\ln v)^\theta\right]^{\frac{1}{\theta}}\right\} \tag{5.18}$$

When the correlation coefficient θ is given, the steps for generating 2 randomly correlated random variables are listed below:

1. Generate a uniformly distributed random variable u_0 on [0,1];

2. For a given u_0, the below expression is a function of v;

$$\begin{aligned} c(v) &= \left.\frac{\partial C_\theta(u,v)}{\partial u}\right|_{u=u_0} \\ &= \exp\left\{-\left[(-\ln u_0)^\theta + (-\ln v)^\theta\right]^{\frac{1}{\theta}}\right\}\left[\left((-\ln u_0)^\theta + (-\ln v)^\theta\right)^{\frac{1-\theta}{\theta}}\right]\left[\frac{(-\ln u_0)^{\theta-1}}{u_0}\right] \end{aligned} \tag{5.19}$$

3. Generate a uniformly distributed random variable w on [0,1];

4. Let the c(v) expression equal to w, the value of v can be calculated as v_0;

5. Generate X and Y by inverse CDF transformation based on any desired marginal distribution.

$$\begin{aligned} X &= F_X^{-1}(u_0) \\ Y &= F_Y^{-1}(v_0) \end{aligned} \tag{5.20}$$

For example, if X and Y are both Gaussian distribution then

$$F_X(x) = \frac{1}{\sigma_x\sqrt{2\pi}}\int_{-\infty}^{x}\exp\left[-\frac{1}{2}\left(\frac{t-\mu_x}{\sigma_x}\right)^2\right]dt, -\infty < x < \infty$$

$$F_Y(y) = \frac{1}{\sigma_y\sqrt{2\pi}}\int_{-\infty}^{y}\exp\left[-\frac{1}{2}\left(\frac{t-\mu_y}{\sigma_y}\right)^2\right]dt, -\infty < y < \infty \qquad (5.21)$$

X and Y generated in this way will satisfy the correlation specified by this Gumbel-Hougaard copula. One small problem of this procedure is that the root finding function fzero.m embedded in matlab has an accuracy problem, it usually fails when the copula marginal values u_0 and v_0 are close to the margins, 0 or 1.

Matlab code for generating random variables based on copula method is listed in App. Code 10, the 2 root finding function copulafunc01 and copulafunc02 are also in App. CODE COPULAFUNC01 and App. CODE COPULAFUNC02. The scatter plot Figure 5-16 shows the random variables generated by this method. Data table is also listed below in which a fraction of the generated data samples are given.

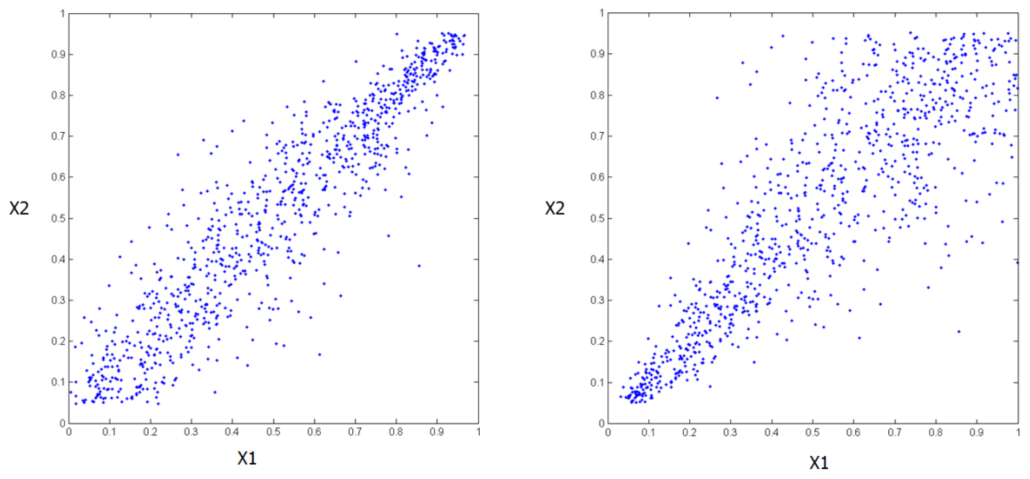

Figure 5-16: Scatter plot for data samples generated by Gumbel-Hougaard (left) and Clayton (right) copula

Table 5-6 shows that when the correlation between 2 random variables are both positive concordant, different copula may generate close results, as can be seen from the table that v0 from Gumbel-Hougaard copula and v0 from Clayton copula tend to be close to each other.

Table 5-6: Data samples generated by Gumbel-Hougaard and Clayton copula

u0	w	v0(Gumbel-Hougaard)	v0(Clayton)
0.49500634	0.657534063	0.534995266	0.581262507
0.448410818	0.356282473	0.402889409	0.420354036
0.488448956	0.434591449	0.46520529	0.485529779
0.802255248	0.703079518	0.824432405	0.86824191
0.700554743	0.212614295	0.615086769	0.570082037
0.126555527	0.931439432	0.30783879	0.221414899
0.09257591	0.697358495	0.14990779	0.114180137
0.812984413	0.428723755	0.793888784	0.762200991
0.470034524	0.38205225	0.431739518	0.449334519
0.153963177	0.321816061	0.129734509	0.140638192
0.796514658	0.514381006	0.790200979	0.787168704
0.666719646	0.806944557	0.731159155	0.83351066
0.534661541	0.968731381	0.726089535	0.908696513
0.781711601	0.028200965	0.574501833	0.43446959
0.645114315	0.253335718	0.567992264	0.548448775
0.062540808	0.34055407	0.057550025	0.058031066
0.1279603	0.536264105	0.150948172	0.138186664
0.686504975	0.874953389	0.769340903	0.891857229
0.210680584	0.385192038	0.193207037	0.202636557

It should be noted from the simulation result table that even though with the same marginal distribution, simulation with positive and negative correlation result in very different failure rate. For the Frank's copula case, the positive correlation results in a failure rate 24 times higher than that of the negative case. This can be explained by Figure 5-17. The left figure is the positive correlation case; the right figure is the negative correlation case. For both positive and negative correlations, the marginal distributions are the same X~N(3000, 500^2). However the positive correlation case has much higher chance of falling in the failure region. This example shows that it's important to take into consideration of the correlation between random variables during design or reliability evaluation.

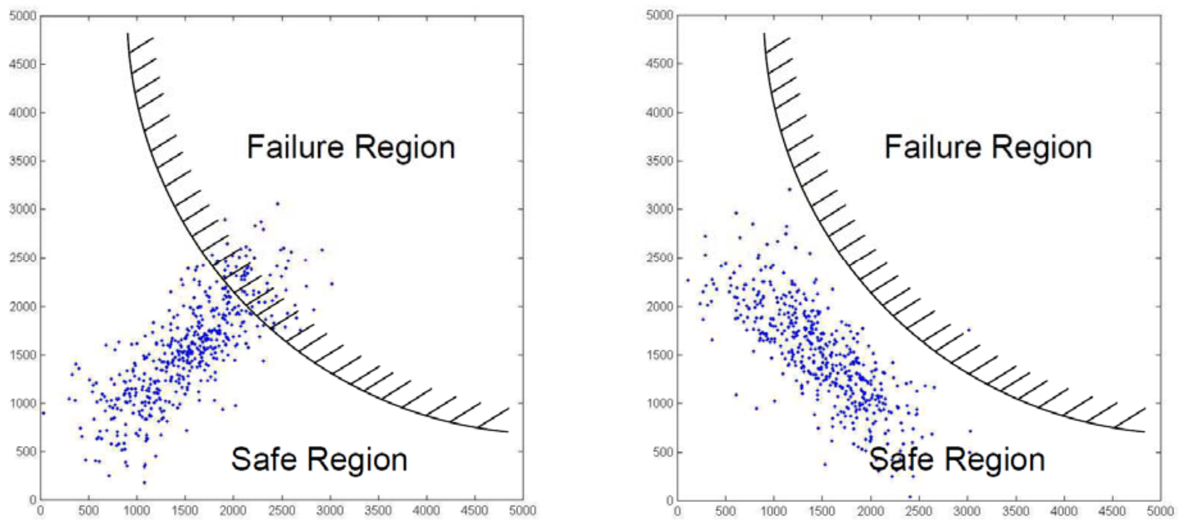

Figure 5-17: Illustration of how positive and negative correlation influence failure rate

5.4 Summary

From the discussions above, one may find some advantages and drawbacks of copula for modeling random variable correlations.

5.4.1 Advantages of Copula

Compared with existing methods of modeling correlations, copula has many advantages:

1) Copulas have very simple mathematical forms.

2) Copula method is a **scale free** measure of correlation, it represent the rank correlations only regardless of the scale. Thus both linear and different degree of nonlinearity can be modeled by a same copula. Property of scale freeness also determines that monotone increasing transform does not change copula.

This is an advantage over other method, ie, Pearson's correlation coefficient $\rho = \dfrac{COV(X,Y)}{\sqrt{DX}\cdot\sqrt{DY}}$.

3) Copula method decouples the marginal distribution with joint distribution. Different marginal distributions may share a same copula.

4) Copula can be used to model dependences for any marginal distributions.

5.4.2 Disadvantages of Copula

With a same correlation coefficient, say, the same Kendall's tau, data samples from different families of copula will produce different simulation results.

CHAPTER 6: OPTIMIZATION BASED ON STRUCTURE SURROGATE MODEL

6.1 Simulation Problem

The MCS simulation based on the precise model are usually very time consuming. Taking author's computer for an example, the author's computer has an 8 core CPU with each of which running at 1.6 GHz, memory is 8GB. With this computer configuration for 1e6 samples of simulation for ten bar truss model, it takes 14 hours. For complex models and FEA models it could take longer. For satisfactory precision of MCS, large data sample is required, which even increase the simulation time. It's necessary to find methods to reduce the computation expenses.

6.2 Solutions for the Problem

6.2.1 Importance Sampling

In Importance Sampling, sample values of the random variables are generated using a sampling PDF that yields samples of high probability of failures, instead of the original true PDF. Since the occurrence of failure is higher than the true model, the sample size can be considerably decreased. Details about Importance sampling can be found in [Doucet, A., et al, 2001].

6.2.2 Surrogate Model

Surrogate model is based on replacing the original computationally expensive model by a computationally affordable substitute model that is a close approximate of the true model. Introduction of surrogate model can help reduce the computational expenses greatly. Commonly used methods to obtain a surrogate model include Artificial Neural Network, interpolation or regression methods. In this thesis Kriging will be used to find a surrogate model for the limit-state function.

6.3 Kriging

Kriging is a group of geostatistical techniques to interpolate the value of a random field (e.g., the elevation, z, of the landscape as a function of the geographic location) at an unobserved location from observations of its value at nearby locations. Kriging is basically an interpolation method that is used to estimate the value at a given spatial point when the values at a set of neighborhood sampling points are given. The results of Kriging are the estimated function value at any given point and the estimating variance. Details about kriging can be found in [Isaaks, E.H., et al, 1989].

In this thesis, Kriging is applied as a method of regression/interpolation. The regression model obtained by Kriging is served as a surrogate model for a computationally expensive function. Compared with the original model, the surrogate model based on Kriging is computationally less expensive, thus it makes Monte Carlo Simulation of millions of samples possible.

Illustrated in Figure 6-1 is an example of Kriging interpolation. The line in the center is the estimated value at each point; the encompassing lines form the upper and lower bound of 95% confidence intervals; sampling point with values are rectangles.

Suppose the measured value at sampling points P1…Pn are φ1, … φn respectively, the value at point Pu need be estimated. The general interpolating method can be expressed as:

$$\phi_u = \sum_{i=1}^{n} \lambda_i \phi_i \tag{6.1}$$

λ_i is the Kriging weight, it is a decreasing function of distance from Pu to Pi. For Kriging, the selection of λ_i need also satisfy the requirement of minimizing estimating error variance.

This section will give a brief review of Kriging method, more details can be found in [Issaks, E.H., 1989]. General form of Kriging can be expressed as:

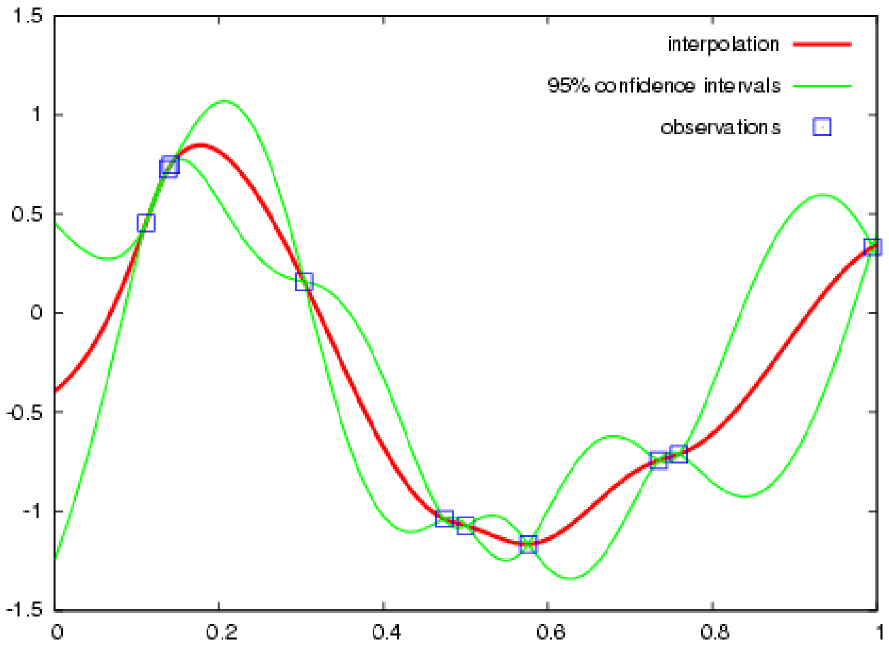

Figure 6-1: Kriging in 2D plane (figure from Wikipedia.com)

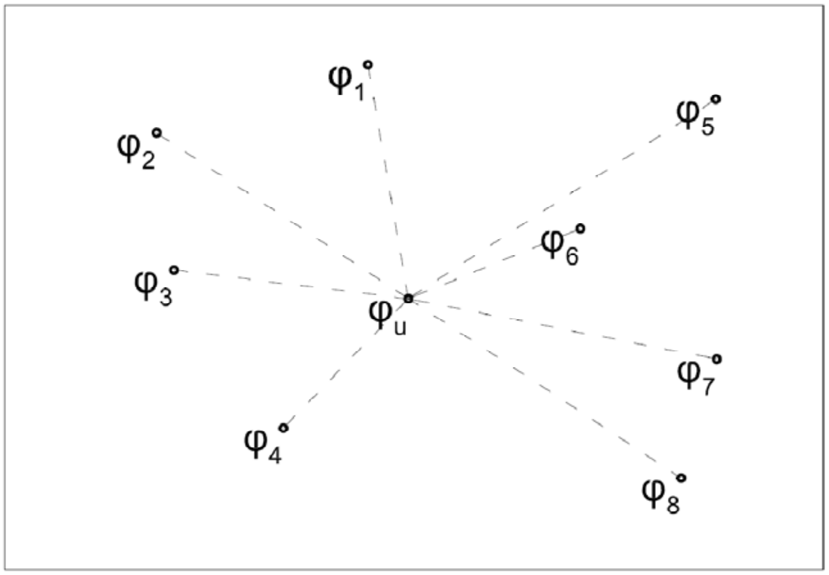

Figure 6-2: Kriging using sampling points values to estimate value at a given point

$$Z^*(\mathbf{u}) - m(\mathbf{u}) = \sum_{\alpha=1}^{n} \lambda_\alpha \left[Z(\mathbf{u}_\alpha) - m(\mathbf{u}_\alpha) \right] \qquad (6.2)$$

In which:

$Z^*(\mathbf{u})$: Estimation of Z(u);

$m(\mathbf{u})$: Mean value at point u;

λ_α: Kriging weight that need to be determined;

$Z(\mathbf{u}_\alpha)$: Measured value at point u_α;

$m(\mathbf{u}_\alpha)$: Mean value at point u_α;

The objective is to determine the kriging weights λ_α such that the variance of the estimator is minimized:

$$\sigma_E^2(\mathbf{u}) = Var\{Z^*(\mathbf{u}) - Z(\mathbf{u})\} \qquad (6.3)$$

Under the constraint:

$$E\{Z^*(\mathbf{u}) - Z(\mathbf{u})\} = 0 \qquad (6.4)$$

In which $Z(\mathbf{u})$ is the value of random process at the estimating point.

$Z(\mathbf{u})$ can be decomposed to mean (or trend) $m(\mathbf{u})$ and residual $R(\mathbf{u}) = Z(\mathbf{u}) - m(\mathbf{u})$. Residual satisfies the mean and covariance conditions:

$$E\{R(\mathbf{u})\} = 0 \qquad (6.5)$$

$$COV\{R(\mathbf{u}), R(\mathbf{u}+\mathbf{h})\} = E\{(R(\mathbf{u}) - 0)\cdot(R(\mathbf{u}+\mathbf{h}) - 0)\} = E\{R(\mathbf{u})\cdot R(\mathbf{u}+\mathbf{h})\} = C_R(\mathbf{h}) \qquad (6.6)$$

In which $C_R(\mathbf{h})$ is the covariance function of residuals. It is a function of distance h. Covariance functions have already been discussed in KL transform section.

Based on the treatment of mean value $m(\mathbf{u})$, there are 3 main kriging method: Simple Kriging; Ordinary Kriging; Kriging with a trend. The first two methods will be used in this thesis.

1) Simple Kriging

In simple kriging, the mean $m(\mathbf{u})$ is taken as a constant:

$$m(\mathbf{u}) = 0 \tag{6.7}$$

Thus (*) becomes:

$$Z_{SK}^{*}(\mathbf{u}) = m + \sum_{\alpha=1}^{n} \lambda_{\alpha}^{SK}(\mathbf{u}) \cdot \left[Z(\mathbf{u}_{\alpha}) - m \right] \tag{6.8}$$

Based on this the estimation error can be found thus the estimation error variance can be found. Take derivatives for the estimation error variance with regard to each kriging weight $\lambda_{\alpha}^{SK}(\mathbf{u})$ and set to zero one get a system of equations:

$$\sum_{\beta=1}^{n} \lambda_{\beta}^{SK}(\mathbf{u}) \cdot C_R\left(\mathbf{u}_{\alpha} - \mathbf{u}_{\beta}\right) = C_R\left(\mathbf{u}_{\alpha} - \mathbf{u}\right) \qquad \alpha = 1,\ldots,n \tag{6.9}$$

Express this equation system in the form of matrix as below:

$$\mathbf{K}_{SK} \cdot \lambda_{\beta}^{SK}(\mathbf{u}) = \mathbf{k} \tag{6.10}$$

In which:

$\mathbf{K}_{SK}$ is the covariance matrix between sampling points;

$\mathbf{k}$ is the covariance vector between sampling points and estimating point.

Kriging weights can be computed:

$$\lambda_{\beta}^{SK}(\mathbf{u}) = \mathbf{K}_{SK}^{-1} \cdot \mathbf{k} \tag{6.11}$$

2) Ordinary Kriging

Assume that the mean is constant at the neighborhood of each sampling point:

$$m(\mathbf{u}_\alpha) = m(\mathbf{u}) \tag{6.12}$$

Let the sum of Kriging weight equal to 1: $\sum_{\alpha=1}^{n} \lambda_\alpha(\mathbf{u}) = 1$,

This constraint optimization problem can be solved by introducing Lagrangian:

$$L = \sigma_E^2(\mathbf{u}) + 2\mu_{OK}(\mathbf{u}) \cdot \left[1 - \sum_{\alpha=1}^{n} \lambda_\alpha^{OK}(\mathbf{u}) \right] \tag{6.13}$$

In which $\mu_{OK}(\mathbf{u})$ is the Lagrangian parameter

Take derivative for each Kriging weight $\lambda_\alpha^{OK}(\mathbf{u})$, take derivative for Lagrange parameter $\mu_{OK}(\mathbf{u})$, this system of equations is obtained:

$$\begin{cases} \sum_{\beta=1}^{n} \lambda_\beta^{OK}(\mathbf{u}) \cdot C_R(\mathbf{u}_\alpha - \mathbf{u}_\beta) + \mu_{OK}(\mathbf{u}) = C_R(\mathbf{u}_\alpha - \mathbf{u}) & \alpha = 1,\ldots,n \\ \sum_{\alpha=1}^{n} \lambda_\alpha^{OK}(\mathbf{u}) = 1 \end{cases} \tag{6.14}$$

It has a matrix form:

$$\begin{bmatrix} \mathbf{K} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} \end{bmatrix} \cdot \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \ldots \\ \lambda_n \\ \mu \end{bmatrix} = \begin{bmatrix} k_1 \\ k_2 \\ \\ k_n \\ 1 \end{bmatrix} \tag{6.15}$$

Solve it and the Kriging weight and Lagrange parameter can be obtained:

$$\begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \dots \\ \lambda_n \\ \mu \end{bmatrix} = \begin{bmatrix} \mathbf{K} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} \end{bmatrix}^{-1} \cdot \begin{bmatrix} k_1 \\ k_2 \\ \\ k_n \\ 1 \end{bmatrix} \tag{6.16}$$

6.4 Illustrated Examples

6.4.1 Surrogate Model for Ten Bar Truss Case

It is necessary to find a surrogate model to replace the original model. This surrogate model must be computationally less expensive with acceptable precision. Usually computational efficiency is in contrast with precision, one can get high speed computation at the expenses of precision or get high precision at the expense of long running time.

A surrogate model is an approximation of the original model, it can be regarded as a curve fitting or interpolating method. Usually these methods are applied in application: Least Square Fit; Artificial Neural Network; Kriging. In this example, Kriging method will be applied to obtain a surrogate model for the ten bar truss model.

The covariance function is selected as:

$$C(d) = 1 - 1.5\left(\frac{d}{d_{\max}}\right) + 0.5\left(\frac{d}{d_{\max}}\right)^3 \tag{6.17}$$

The characteristics of this covariance function can be observed. The covariance between 2 points decreases as the distance between these 2 points increase. If the 2 points overlap, the covariance is 1; if the distance reaches the maximum allowable distance, the covariance is 0, as shown in Figure 6-3.

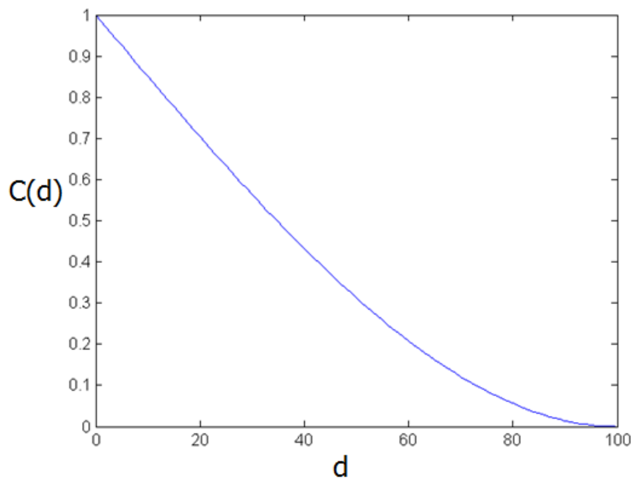

Figure 6-3: Covariance function

$$C(d) = \begin{cases} 1 : d = 0 \\ 0 : d = 1 \end{cases}$$

Kriging method is an exact interpolation method, it preserves the values at the observation points. For this ten-bar-truss problem, the observation points are chosen by these rules:

1) Sampling points are sampled from three dimensions: X1, X2, X3;

2) On each axis 5 points are sampled;

3) For each axis, the left most point is μ-2σ; right most point is μ-2σ; interval is σ;

4) 5 points on each of the 3 axes are combined to each other, totally there are 125 sampling points.

Since here the mean value of the random field is not constant, Ordinary Kriging (OK) will be used. Objective is to find a set of Kriging coefficients that minimize the estimation variance at each estimation point with the constraint that sum of the coefficients equate to 1:

Minimize: $\sigma^2(U)$

Subject to: $\sum_{i=1}^{n(U)} \lambda_i^{OK}(U) = 1$

For a given estimation point, a set of Kriging coefficients $\lambda_i^{OK}(U)$ (i=1…n(U)) can be obtained by solving this constrained optimization problem. The estimation value at any point can be computed by

$$Z_{OK}^{*}(U) = \sum_{i=1}^{n(U)} \lambda_i^{OK}(U) \cdot Z(U_i) \tag{6.18}$$

Where U_i's are the sampling points and $Z(U_i)$ are the observation values at these sampling points.

To achieve high computational speed, the inverse of the covariance matrix of observation points and the observation value of sampling points are store in a stand-alone data file.

The comparison between the results of original precise MCS and that of the Kriging method and several other methods are listed below in the table. The Kriging surrogate model reduces the computation time more than 30 times and the precision is acceptable.

Comparison of time consumption is made in Table 6-1 between precise MCS and Kriging-based surrogate model:

Table 6-1: Comparison between Kriging-based method and other methods

	Precise MCS	Kriging model		MVFOSM reliability index		HL reliability index	
Sample size	1e4	1e4	1e6	/	/	/	/
Running time (s)	500	16	1600	/	/	/	/
Rod (Ai independent)	P_f	P_f	P_f	β	P_f	β_{HL}	P_f
3	0.0330	0.0346	0.0343	2.9953	0.0014	1.8756	0.0304
7	0.0322	0.0404	0.0382	2.8889	0.0019	1.8388	0.0330

Table 6-2 shows the limit-state function value computed from precise model and surrogate model. The outlier values comes from those point that are very far away from the sampling point set. Although the outlier values brings greater errors, the trend that for each negative value from precise model, the surrogate model nearly always also gives a negative value is valuable for evaluating failure rate as each negative value represent a failure case regardless of the magnitude.

Table 6-2: The results of the precise limit-state values and the results from kriging approximation, with error percentage

	Rod 3			Rod 7		
#	Precise Value	Kriging Approximation	Error Percentage	Precise Value	Kriging Approximation	Error Percentage
1	3.530E+07	3.557E+07	0.77%	4.875E+07	4.913E+07	0.78%
2	3.787E+07	3.790E+07	0.08%	4.445E+07	4.434E+07	-0.24%
3	4.829E+07	4.844E+07	0.32%	2.795E+07	2.860E+07	2.35%
4	4.686E+07	4.683E+07	-0.07%	4.422E+07	4.411E+07	-0.24%
5	3.091E+07	3.144E+07	1.72%	3.783E+07	3.784E+07	0.03%
6	4.438E+07	4.453E+07	0.33%	1.466E+07	1.305E+07	-10.97%
7	4.798E+07	4.805E+07	0.15%	4.667E+07	4.672E+07	0.10%
8	2.893E+07	2.925E+07	1.12%	3.119E+07	3.147E+07	0.89%
9	1.840E+07	1.735E+07	-5.73%	4.518E+07	4.526E+07	0.16%
10	2.178E+07	2.160E+07	-0.84%	4.836E+07	4.871E+07	0.71%
11	3.327E+07	3.354E+07	0.80%	4.388E+07	4.379E+07	-0.20%
12	1.018E+07	7.899E+06	-22.39%	2.751E+07	2.775E+07	0.86%
13	5.171E+07	5.236E+07	1.27%	3.573E+07	3.613E+07	1.13%
14	4.141E+07	4.127E+07	-0.35%	3.052E+07	3.070E+07	0.60%
15	3.520E+07	3.541E+07	0.60%	3.434E+07	3.455E+07	0.59%

For examination, the random variables used for the computation are listed in Table 6-3:

6.4.2 Optimization Based on Surrogate Model

Methods for evaluating the probability of failure by analytical and simulation methods have been discussed in the previous sections. For most of these methods iterative numerical algorithms have been employed which makes it difficult to find the analytical relationships between design variables and Pf. The optimization requirement requires that the designer finds a design point of X1 X2 and X3 such that the probability of failure of rod 3 and rod 7 is less than permissible value. Since analytical relations between design variables and objective function are difficult to obtain, numerical interpolation method can be used to find the optimal design point.

Table 6-3: Random variables used for the computation

	Random Variables		
#	X1	X2	X3
1	9.7258E-05	2.1434E-05	1.2517E-04
2	1.0103E-04	2.7155E-05	1.1507E-04
3	1.2102E-04	2.9693E-05	8.8338E-05
4	1.1761E-04	3.9391E-05	1.1390E-04
5	9.0766E-05	2.6485E-05	1.0267E-04
6	1.1211E-04	3.2160E-05	7.4217E-05
7	1.2058E-04	3.1655E-05	1.1972E-04
8	8.8393E-05	2.0886E-05	9.2993E-05
9	7.6452E-05	3.9376E-05	1.1561E-04
10	8.0029E-05	3.3204E-05	1.2331E-04
11	9.4264E-05	2.1228E-05	1.1423E-04
12	6.9929E-05	1.9081E-05	8.8261E-05
13	1.2983E-04	2.8651E-05	9.9298E-05
14	1.0626E-04	4.1380E-05	9.0942E-05
15	9.6257E-05	3.7530E-05	9.6502E-05

The problem can be described as:

$$\text{Objective function: Min } aX_1 + bX_3 \tag{6.19}$$

$$\text{Subject to: } \begin{cases} P\{g_3(X_1, X_3)\} \le 0.01 \\ P\{g_7(X_1, X_3)\} \le 0.01 \\ X_2 = 0.3e-4 \end{cases} \tag{6.20}$$

In which a and b are 2 constants related with rod numbers and rod lengths, objective function $aX_1 + bX_3$ represents the total truss weight; g3 and g7 are the limit-state function of rod 3 and rod 7 respectively. Since the value of X2 does not have much influence on the safety of rod 3 and rod 7, in this study X2 will be set as a constant value 0.3e-4. In this study only the mean value X1 and X3 are decision variables, the variance σ1 and σ3 are state variables set as a σ=0.2X.

Since the analytical form of $P\{g_i(X_1, X_3)\} \le 0.01$ is difficult to obtain, numerical multivariate interpolation method will be used. Sample points are taken at 1e-3 and 1.2e-3, totally there are four sampling points, the Pf at these points are computed in Table 6-4 by any of the previous methods (HL, SORM, Gradient, etc):

Table 6-4: Sampling points and their values

	X1	X3	Pf3	Pf7
1	1.00E-03	1.00E-03	0.0303	0.003
2	1.00E-03	1.20E-03	0.0307	0.00899
3	1.20E-03	1.00E-03	0.0083	0.0331
4	1.20E-03	1.20E-03	0.0084	0.00904

There are several multivariate interpolation methods, i.e. bilinear, bicubic, spine, Kriging, inverse-distance weighted, minimum surface curvature, etc. For this problem bilinear interpolation will be used. Bilinear interpolation is not linear since it has a quadratic term in it. Basic idea of bilinear interpolation is, given 4 sampling points, first carry out the linear interpolation between 2 pairs of points on 2 opposite edges; after this step the point values on the opposite 2 edges are known; any value inside the area can be interpolated linearly by the two points on the two opposite edges whose values have been interpolated in the previous step. One example of bilinear interpolation is illustrated in Figure 6-4: in this example, 4 sampling points are (0 0), (0 1), (1 0), (1 1); the respective function values on the four points are respectively: 0, 1, 1, 0. (App. Code 44)

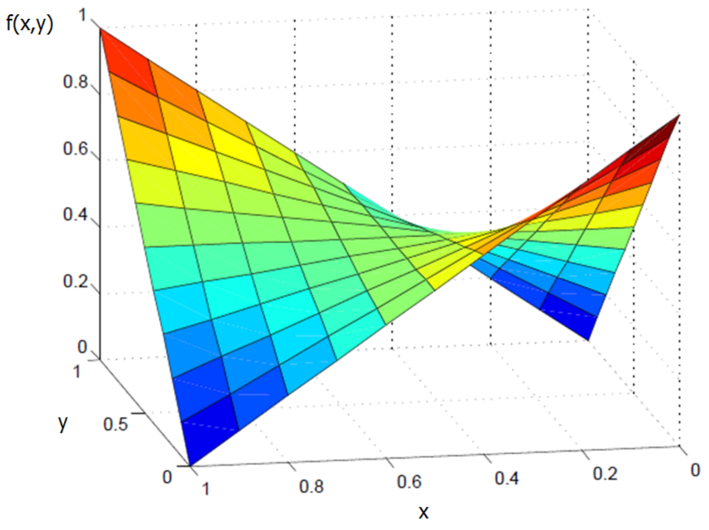

Figure 6-4: Example of bilinear interpolation

The bilinear interpolation has this equation form:

$$f(x,y) = c_1 + c_2 x + c_3 y + c_4 xy = \begin{bmatrix} 1 & x & y & xy \end{bmatrix} \cdot \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} \tag{6.21}$$

If four sample points and their respective function values are obtained, $[c_i]$ can be solved:

$$\begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} = \begin{bmatrix} 1 & (x)_1 & (y)_1 & (xy)_1 \\ 1 & (x)_2 & (y)_2 & (xy)_2 \\ 1 & (x)_3 & (y)_3 & (xy)_3 \\ 1 & (x)_4 & (y)_4 & (xy)_4 \end{bmatrix} \cdot \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} = \begin{bmatrix} 1 & (x)_1 & (y)_1 & (xy)_1 \\ 1 & (x)_2 & (y)_2 & (xy)_2 \\ 1 & (x)_3 & (y)_3 & (xy)_3 \\ 1 & (x)_4 & (y)_4 & (xy)_4 \end{bmatrix}^{-1} \cdot \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{bmatrix} \tag{6.22}$$

The coefficient vector is calculated as,

For rod3, $\begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} = \begin{bmatrix} 0.1308 \\ -1.025e3 \\ 95 \\ -7.5e5 \end{bmatrix}$; for rod7, $\begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix} = \begin{bmatrix} 0.1513 \\ 17.5 \\ -1.188e3 \\ -1.25e5 \end{bmatrix}$

The constraints:

$$\begin{cases} P\{g_3(X_1, X_3)\} \le 0.01 \\ P\{g_7(X_1, X_3)\} \le 0.01 \end{cases}$$

are both active, they could be replaced by the corresponding equality constraints:

$$\begin{cases} P\{g_3(X_1, X_3)\} = 0.01 \\ P\{g_7(X_1, X_3)\} = 0.01 \end{cases}$$

Since these two constraints can uniquely determine a pair of point on XY plane, the objective function becomes redundant. The pair of point that satisfies the 2 constraints is solved as $\begin{cases} X1 = 1.1856e-4 \\ X2 = 1.1920e-4 \end{cases}$.

Plug this point into the reliability computation algorithm and find the corresponding Pf of rod 3 and 7:

Pf3=0.0091;
Pf7=0.0095;
These two values satisfy the constraints and are very close to the optima. The error is caused by the approximation error caused by bilinear interpolation.
To achieve higher precision, such steps can be carried out in iterations in smaller intervals around the point obtained in the previous step. After two or three iterations, satisfactory optimum point can be found.

6.5 Comparison between Kriging and other interpolation methods

There are many other interpolation methods that can be used to build surrogate model, such as Artificial Neural Network, nearest neighbor interpolation, inverse distance squared weighting, splines, etc. Each interpolation method has its advantage and disadvantage depending on the sample data configuration and the specific problem setting. Whether one interpolation method is superior to another depends on many factors, it's difficult to draw a general conclusion which one is better than other ones. This section will give some comparisons of various interpolation methods by quoting published papers.

In [Chowdhury, M., Alouani, A., Hossain, F. 2010], the paper applies both Kriging and Artificial Neural Network (ANN) method to predict the spatial mapping of arsenic contamination of groundwater. The paper observes that ANN as a highly nonlinear interpolation method, is more accurate when compared to ordinary kriging under the same sampling data set. The paper shows that for the case studied, the successful detection rate of ANN is 15% higher than that by Kriging and that the false hope rate of ANN is 10% lower than that of Kriging.

In [WaternBerg, D., et al, 1991], the author estimates ground water contaminations from a limited number of observations by three different interpolation methods: Kriging, nearest neighbor interpolation and inverse distance squared weighting. The results show that although kriging is an optimal estimation method, it's not necessarily better than other simpler interpolation methods, though it usually is more complex to use. However, Kriging by its complex configurations allows further improvement.

In [Laslett, G.M., 1994], the author makes an empirical comparison of the predictive performances between Kriging and Splines by dividing dataset into modeling sets and prediction sets. The author finds that in the particular case, kriging sometimes outperforms splines by a considerable margin, except for that, kriging never performs worse than splines. The paper also concludes that when the sampling set are big enough, both methods perform equally well; when the sampling set are sparse, kriging outperforms spline.

In [Zimmerman, D., et al, 1999], a comparison of interpolation accuracy was made between ordinary and universal kriging and two types of inverse squared-distance weighting. The experiment considers also the effect of other factors such as sampling patterns. The paper observes that "...the two kriging methods were substantially superior to the inverse distance weighting methods over all levels of surface type, sampling pattern, noise and correlation". Except for that, the author also finds that the interpolation accuracies of all these methods are very sensitive to sampling pattern, the interpolation accuracy deteriorate as the sampling method changes from hexagonal to inhibited to random, and the worst sampling method is clustered. The author's findings about kriging and inverse squared-distance weighting agree with the conclusions from some authors, however it is contradictory with several other authors. By investigating the non-performance-related aspects, the author also concludes that kriging has more advantages than inverse squared-distance weighting in some other aspects, for example, kriging gives the estimation variance information while inverse squared-distance weighting does not.

6.6 Summary

This example shows that Kriging can be used to effectively reduce the simulation time. It can be used to build a computationally inexpensive surrogate model for many simulation problems. However, if the computation point is far from the sample points, the interpolation algorithm will show great error. This requires that the computation point be within the coverage of sampling point set. For highly nonlinear problems, to achieve satisfactory results, it's necessary to increase the number of sampling points at areas where nonlinearity is high, which in turn increases the computation expenses. To overcome this problem, the max covariance radius d_{max} can be set smaller so that less sampling points are included in computation. Since the number of sample points decreases geometrically as the max covariance radius decreases, this method can effectively reduce the computational expenses. However, if the max covariance radius is too small, the interpolative surface will be less smooth, which is also undesirable. The selection of max covariance radius need reach a trade-off or balance between the requirement of high performance and high accuracy. Except for Kriging, other interpolation methods such as Artificial Neural Network can also be applied to build surrogate models.

CHAPTER 7: HEURISTIC GRADIENT BASED SEARCHING ALGORITHM

7.1 Problem Description

The reliability-based optimization problems can usually be expressed in this form:

$$\begin{aligned} \min \quad & f(\mathbf{X}) \\ s.t. \quad & \mathbf{h}(\mathbf{X}) = \mathbf{0} \\ & \mathbf{g}(\mathbf{X}) \le \mathbf{0} \end{aligned} \tag{7.1}$$

In which $f(\mathbf{X})$ is the objective function, $\mathbf{h}(\mathbf{X})$ are the equality constraints, $\mathbf{g}(\mathbf{X})$ are inequality constraints.

For this nonlinear programming problem, if both equality constraints $\mathbf{h}(\mathbf{X})$ and inequality constraints $\mathbf{g}(\mathbf{X})$ have analytical form and gradient information is obtainable, then an optimal point X* can be found by solving

$$\begin{cases} \nabla f(\mathbf{X}^*) = \sum_{i=1}^{m} \lambda_i^* \nabla h_i(\mathbf{X}^*) + \sum_{j=1}^{l} \gamma_j^* \nabla g_j(\mathbf{X}^*) \\ \gamma_j^* g_j(\mathbf{X}^*) = 0 & (j = 1, \cdots, l) \\ \gamma_j^* \ge 0 & (j = 1, \cdots, l) \end{cases} \tag{7.2}$$

X* solved from this system of equation is KKT point. If certain convex conditions are met, the KKT point is the global optimum. More details about KKT condition can be found in [Papalambros, P.Y., et al, 2000]. For reliability-based optimization problems, reliability constraints often appear as inequality constraints. Since most applicable algorithms for computing reliability index such as MCS gives a set of discrete numerical values, it's difficult to obtain the analytical form and the usual nonlinear programming algorithms cannot be used.

To solve this problem, a response surface of the reliability constraints is built.

The response surface is based on an interpolation model, for instance, kriging. The general form of kriging interpolation can be expressed as:

$$\phi_u = \sum_{i=1}^{k(u)} \lambda_i(d_i)\phi_i \tag{7.3}$$

Let P$_u$ be an arbitrary point in design space, $\{P_1,...P_{k(u)}\}$ is a set of adjacent sample points that satisfy the requirement that $\{P_i, i = 1\cdots k(u) \| \|P_i - P_u\| \le d_{\max}\}$. In which $d_{\max}$ is the maximum distance, beyond which the covariance is zero. The interpolation value at a certain point depends on the adjacent sample points within the maximum distance, at different estimation point, the adjacent sample point set are not necessarily the same. Or, for different point P$_u$, the set $\{P_1,...P_{k(u)}\}$ is not necessarily the same. Both point set and number of points in the set can be different for different P$_u$. As such, ϕ_u is a segment function and it does not have a uniform form on the design space. A uniform form is a expression that have an identical form over the design space, such as this polynomial: $\beta(x) = ax^2 + bx + c$ defined on $(-\infty,+\infty)$. However, for most optimization algorithms, it is required that the constraints have uniform forms. To solve optimization problem in which some constraints are segment functions or do not have uniform form, a heuristic gradient-based searching algorithm is introduced. An example of segment function is shown in Figure 7-1.

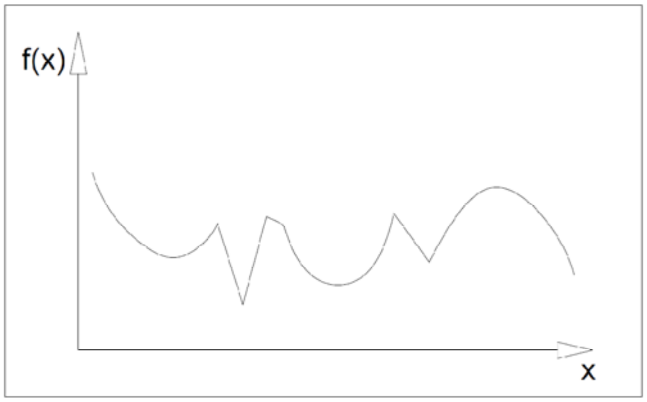

Figure 7-1 Example of segment function

7.2 Response Surface for Constraints without Analytical Form

As has been explained in the previous section, since reliability index β is calculated by numerical iterative process or MCS method, it's difficult to find the analytical form of β in terms of design variables. Without an analytical form of constraint function it's impossible to solve the optimization problem by gradient based searching method.

To obtain an analytical form of reliability index β as a function of design variables, an interpolative response surface can be used. There are many ways to obtain an interpolative response surface, for example, Inverse-Distance Weighting, Polynomial Interpolation, Cubic-Spline Interpolation, Volume-Spline Interpolation, Kriging, etc. In this study multiquadric or kriging interpolation method will be used to obtain response surface.

The general form of multiquadric interpolation is expressed as:

$$f(x,y,z)=\sum_{i=1}^{N} b_i \sqrt{d_i^2+\alpha^2} \tag{7.4}$$

In which,

$f(x,y,z)$: interpolated value at any given point (x,y,z);

d_i: Euclidian distance from interpolation point (x,y,z) to a sample point Pi;

α: an arbitrary constant, α=1 is common practice, in this case, α=1e-4 achieves better results;

b_i: coefficients, can be determined by:

$$\begin{bmatrix} 0 & \sqrt{d_{12}^2+\alpha^2} & \sqrt{d_{13}^2+\alpha^2} & \cdots & \sqrt{d_{1n}^2+\alpha^2} \\ \sqrt{d_{21}^2+\alpha^2} & 0 & \sqrt{d_{23}^2+\alpha^2} & \cdots & \sqrt{d_{2n}^2+\alpha^2} \\ \sqrt{d_{31}^2+\alpha^2} & \sqrt{d_{32}^2+\alpha^2} & 0 & \cdots & \sqrt{d_{3n}^2+\alpha^2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sqrt{d_{n1}^2+\alpha^2} & \sqrt{d_{n2}^2+\alpha^2} & \sqrt{d_{n3}^2+\alpha^2} & \cdots & 0 \end{bmatrix} \cdot \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \qquad (7.5)$$

The term d_{ij} is the Euclidian distance between sample point P_i and P_j; v_i is the function value at the sample point P_i.

Given a set of sampling points $\{P_i | i = 1 \cdots n\}$ and their function values $\{v_i | i = 1 \cdots n\}$, interpolation coefficients $\{b_i\}$ can be calculated. With interpolation coefficients $\{b_i\}$ known, the function value at any interpolation point (x,y,z) can be calculated by equation (7.4).

Selection of sample point is usually based on heuristic or experience, as a common practice, sample point can be selected inside and around feasible region, sampling points on each axis can be made in equal intervals. Once the sampling points are determined, various methods can be applied to compute reliability index. These methods include but not limited to: MVFOSM, HL method, correlated HL method, MCS method, SORM method, Gradient Projection. The selection of these methods need consider the correlations between random variables, if correlation exists, only correlated HL and MCS can be used.

7.3 Heuristic Gradient Based Searching Algorithm

Heuristic gradient-based searching algorithm is a numerical method to find optimal point for optimization problem with constraints that are segment-defined functions, or constraints that do not have uniform expression on design space. It is partially inspired by Gradient Projection method. Consider a starting point P0, if inequality constraints are not considered, the searching direction should be the negative gradient direction of the objective function. Make a step movement toward this direction, the new point becomes: $P_1 = P_0 - \alpha \cdot \nabla f(P_0)$. However due to the existence of the inequality constraint, this P1 may not still be inside the feasible area (points in which satisfies all the constraints). A verification is made to determine whether P1 is still inside the feasible area. If YES, then repeat the previous step and find the next point; if NO, determine all the violated constraints. In this step, the violated inequality constraints become active. A temporary equality constraint optimization problem is obtained. This temporary equality constraint optimization problem can be solved by Gradient Projection method or Reduced Gradient method. This temporary problem exists only for one step. When the temporary optimal point is found, it can be served as the new starting point. These steps repeat until the objective function converges.

This process can be interpreted in 2D plane by Figure 7-2 :

Arrows in the figure are the negative gradient field. From starting point P0 searching along the negative gradient direction, P1 is obtained. Verify inequality constraints violation, answer is NO. Repeat the previous step, find point P2'. Now the inequality constraint is violated. h(X) now becomes active. Project P1P2' to the h(X) plane, P2 is obtained. Repeat this process until convergent point is found.

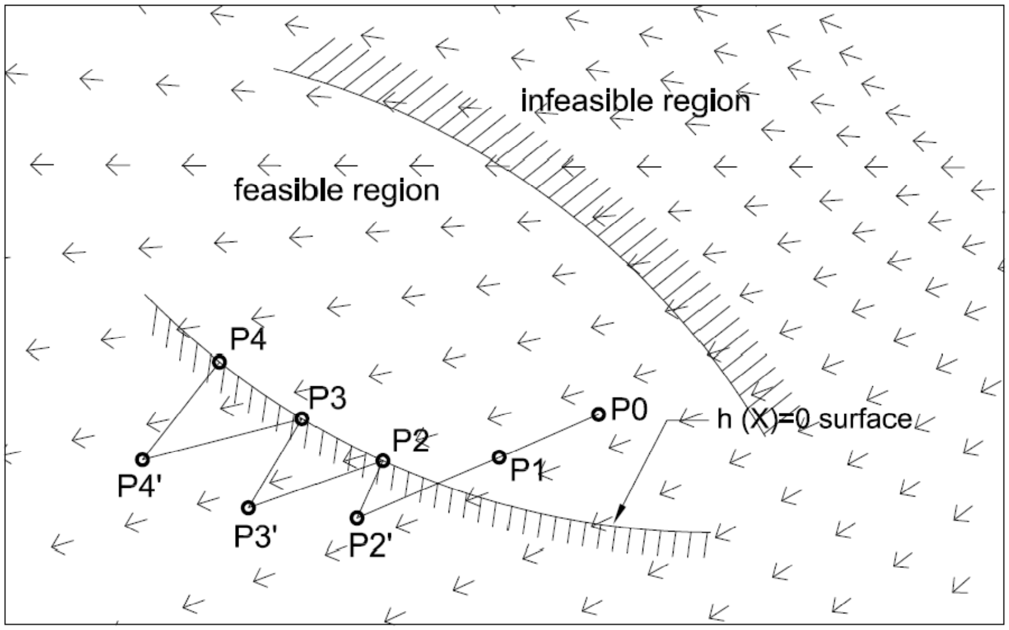

Figure 7-2: Heuristic searching algorithm (arrows denote gradient descent directions)

General steps can be described as:

1) Given a starting point that satisfies all the inequality constraints;

2) Searching along the negative gradient direction of the objective function at the current point;

3) After each moving verify if there's any constraint violated, if any, the violated constraints are now active;

4) Move back to the previous unviolated point, project the negative gradient of the objective function on the tangent plane of the active constraints, move along this direction and find next feasible point;

5) Repeat step2 to step4 until convergent point is found.

Flow chart of the algorithm is shown in Figure-7-3:

7.4 Summary

For optimization problems that have constraints that have no analytical form, but the value and gradient can be computed by iterative numerical algorithms or by response surface method, heuristic searching algorithm can be used to find the optimal point. Heuristic searching algorithm is based on gradient projection method, it checks the constraints violation in each step. Heuristic searching algorithm can be used to find the optimal point for RBDO problems. However, as an iterative numerical searching algorithm that include nested iterative gradient projection algorithm, heuristic searching algorithm is usually time-consuming, especially when there are nested searching steps. Another problem of heuristic searching algorithm is that it is easy to converge to local minimum. To avoid converging to local minimum, large number of starting points need be choosed.

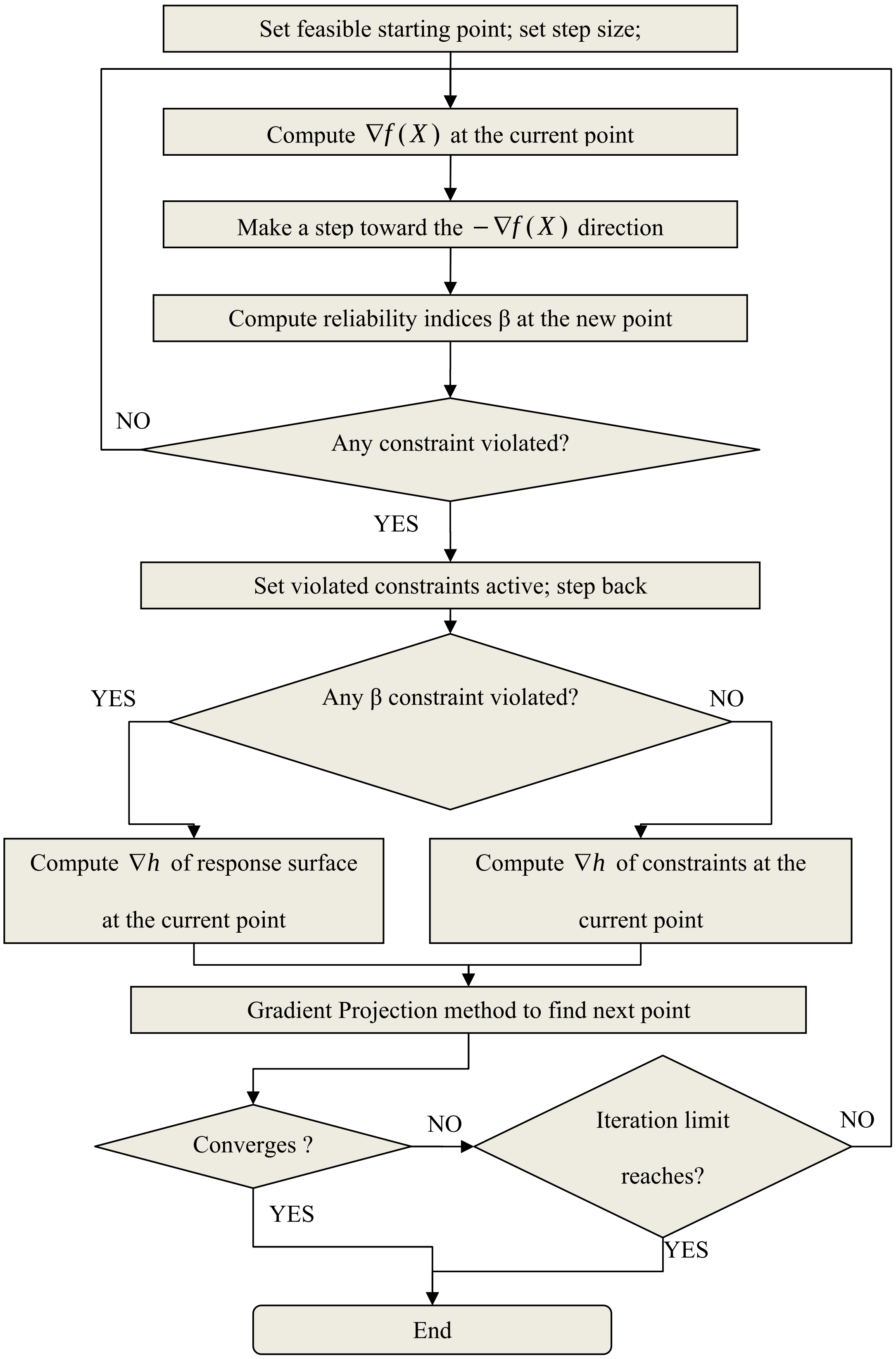
Set feasible starting point; set step size;
Compute $\nabla f(X)$ at the current point
Make a step toward the $-\nabla f(X)$ direction
Compute reliability indices β at the new point
Any constraint violated?
NO
YES
Set violated constraints active; step back
Any β constraint violated?
YES
NO
Compute ∇h of response surface at the current point
Compute ∇h of constraints at the current point
Gradient Projection method to find next point
Converges ?
NO
Iteration limit reaches?
NO
YES
YES
End

Figure-7-3 Flow chart of heuristic search algorithm under response surface constraints

CHAPTER 8: ALTERNATIVE ALGORITHMS TO EVALUATE BETA AND CORRELATED HL METHOD

8.1 Evaluating Pf by Optimization Method

8.1.1 Gradient Projection Method

HL method makes improvement than MVFOSM method by transforming the Gaussian distributed design variables to standard Gaussian distribution. The transformation can be expressed as:

$$U = \frac{X - \mu_X}{\sigma_X} \tag{8.1}$$

Safety index is defined as $\beta = \frac{\mu_{g(X)}}{\sigma_{g(X)}}$ in MVFOSM method, in HL method it can also be interpreted as the shortest distance from the origin to the transformed limit-state zero plane ($\hat{g}(U) = 0$). The safety index β can be obtained by solving this optimization problem:

$$\begin{aligned} &\beta = \min\ f(U) = \left(U^T U\right)^{\frac{1}{2}} = \sqrt{\sum_{i=1}^{n} u_i^2} \\ &\text{subject to : } \hat{g}(U) = 0 \end{aligned} \tag{8.2}$$

The Objective is to find the minimum distance from the origin to the limit-state surface; the constraint is the point should be located on the limit-state surface. See Figure 8-1.

This optimization problem can be solved by gradient projection method. The basic idea of Gradient Projection Method is searching on the constraint plane (in this case the constraint plane is the limit-state function zero plane) along the projection on the constraint plane of the negative gradient direction of objective function until the objective function reaches a minimum.

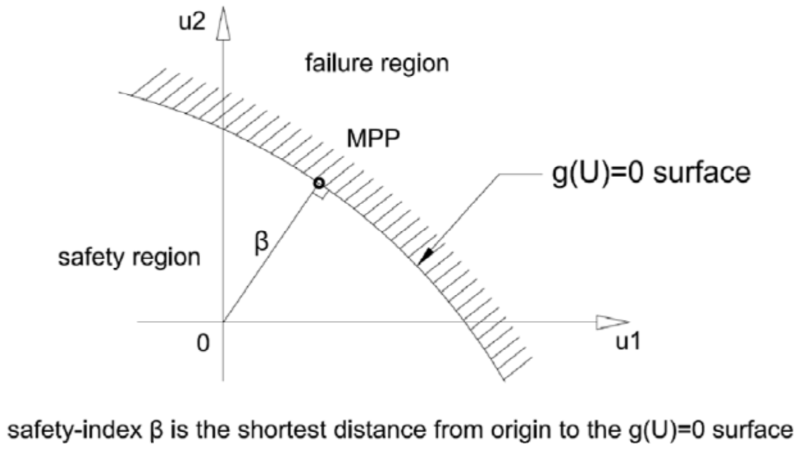
u2
failure region
MPP
g(U)=0 surface
β
safety region
0
u1
safety-index β is the shortest distance from origin to the g(U)=0 surface

Figure 8-1: Geometrical meaning of reliability index

The procedure of Gradient Projection Method can be described as: (More details can be found in [Papalambros, P.Y., 2000])

1) Select a starting point Xk on the constraint plane, the negative objective function gradient is $-\nabla f_k$; project it on the tangent plane that passes the starting point Xk and one get $-\mathbf{P}_k \nabla f_k$. $\mathbf{P}_k$ is the projection matrix;

2) Find a next iteration point X'$_{k+1}$ along the direction $-\mathbf{P}_k \nabla f_k$ on the tangent plane passing point X$_k$, the new point $x'_{k+1} = x_k - \alpha \mathbf{P}_k \nabla f_k$ where α is the step size.

3) Since α is on the tangent plane, usually it is not on the constraint plane, thus it requires to return to the constraint plane. This can be done by drawing a straight line that passes point X'$_{k+1}$ and is perpendicular to the tangent plane. The intersection of this line with the constraint plane X$_{k+1}$ is the next iteration point. Compute the objective function value at this new point.

4) Repeat step1-3 until the objective function value converges.

The procedure is illustrated in Figure 8-2

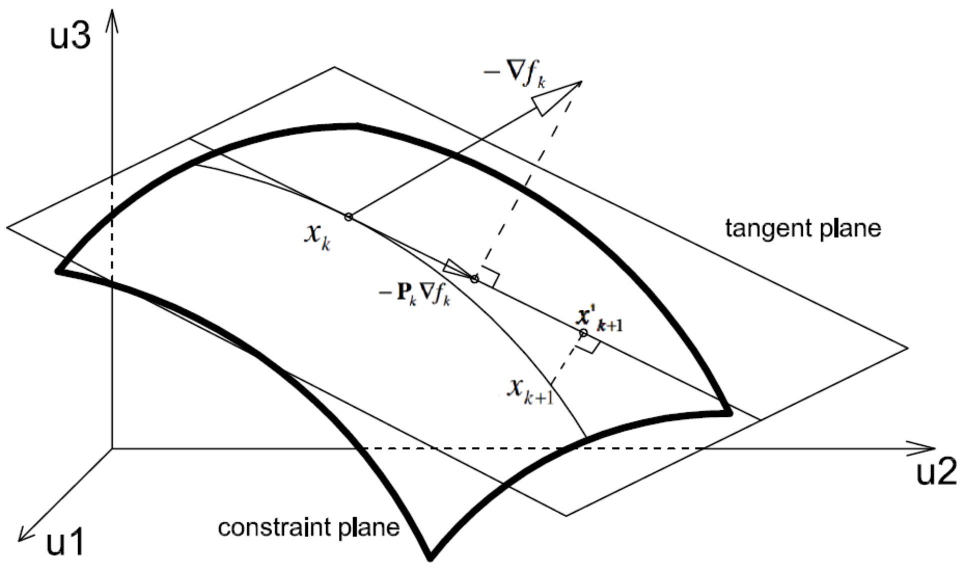

Figure 8-2: Illustration of gradient projection method (1)

The tangent subspace of the constraint plane at point X_k can be defined as:

$T = \{\mathbf{t} \in \Re^n \mid \nabla^T \mathbf{g}_k \cdot \mathbf{t} = 0\}$ Each vector in this subspace are orthogonal to the gradient $\nabla \mathbf{g}_k$;

The normal subspace of the constraint plane at point X_k can be described as:

$$N = \{\mathbf{n} \in \Re^n \mid \nabla \mathbf{g}_k \cdot \mathbf{p} = \mathbf{n}, \mathbf{p} \in \Re^m\} \qquad (8.3)$$

Each vector in this subspace is a linear combination of m constraint functions' gradients at point X_k

T and N are complementary to each other with respect to space $\mathbf{R}^n$, that is to say, any vector **v** in space $\mathbf{R}^n$ can be described as $\mathbf{v} = \mathbf{t} + \mathbf{n}$, where $\mathbf{t} \in T$ and $\mathbf{n} \in N$. Express the objective function gradient in terms of **t** and **n** (Figure 8-3):

$$\nabla f_k = \mathbf{t} + \mathbf{n} \qquad (8.4)$$

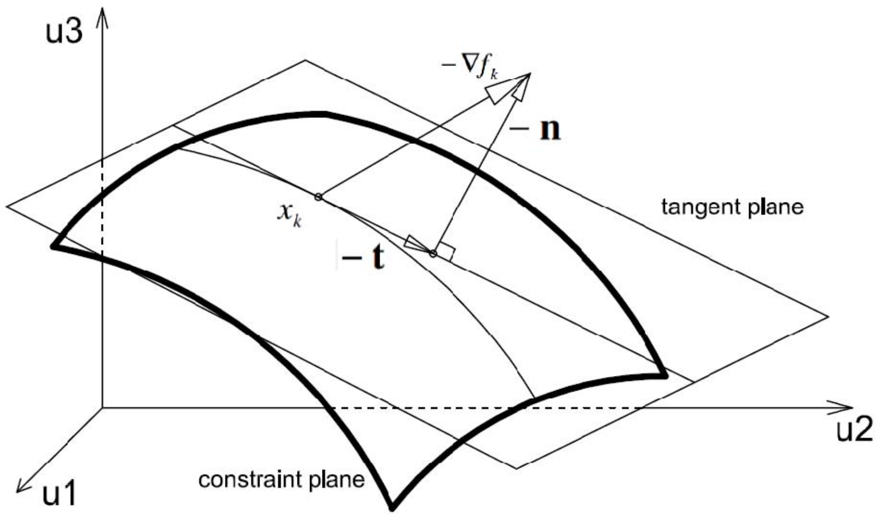

Figure 8-3: Illustration of gradient projection method (2)

Left side multiply the above equation by the gradient of constraint function at X_k:

$$\nabla^T \mathbf{g}_k \nabla f_k = \nabla^T \mathbf{g}_k (\mathbf{t} + \mathbf{n}) = \nabla^T \mathbf{g}_k \cdot \mathbf{t} + \nabla^T \mathbf{g}_k \cdot \mathbf{n} = 0 + \nabla^T \mathbf{g}_k \cdot \mathbf{n} = \nabla^T \mathbf{g}_k \cdot \mathbf{n} \qquad (8.5)$$

Plug (8.3) into (8.5):

$$\nabla^T \mathbf{g}_k \nabla f_k = \nabla^T \mathbf{g}_k \cdot \nabla \mathbf{g}_k \cdot \mathbf{p} \qquad (8.6)$$

Thus $\mathbf{p} = \left(\nabla^T \mathbf{g}_k \cdot \nabla \mathbf{g}_k\right)^{-1} \cdot \nabla^T \mathbf{g}_k \cdot \nabla f_k$ plug it into (**) the relation between ∇f_k and its projection on the tangent plane is obtained:

$$\mathbf{t} = \nabla f_k - \mathbf{n} = \left(\mathbf{I} - \nabla \mathbf{g}_k \cdot \left(\nabla^T \mathbf{g}_k \cdot \nabla \mathbf{g}_k\right)^{-1} \cdot \nabla^T \mathbf{g}_k\right) \cdot \nabla f_k \qquad (8.7)$$

Since $\mathbf{t} = \mathbf{P}_k \nabla f_k$, projection matrix $\mathbf{P}_k$ is obtained:

$$\mathbf{P}_k = \left(\mathbf{I} - \nabla \mathbf{g}_k \cdot \left(\nabla^T \mathbf{g}_k \cdot \nabla \mathbf{g}_k\right)^{-1} \cdot \nabla^T \mathbf{g}_k\right) \qquad (8.8)$$

Once $\mathbf{P}_k$ is obtained, the next point on the tangent plane X'$_{k+1}$ can be obtained by $x'_{k+1} = x_k - \alpha \mathbf{P}_k \nabla f_k$. Step size α can be determined by experience or determined heuristically. If α is chosen too small, the algorithm will converge very slowly; if α is chosen very large, there will be overshoot or will obtain erroneous solutions. For the ten-bar-truss case, the step size is chosen as α=0.5. It’s been proved that this step size is moderate for this case.

From definition of X_{k+1} there exists this relationship between X'_{k+1} and X_{k+1}:

$$\nabla^T \mathbf{g}_k \cdot \left(x'_{k+1} - x_{k+1} \right) = \left(h'_{k+1} - h_{k+1} \right) \tag{8.9}$$

Since $h_{k+1} = 0$, the above equation becomes:

$$\nabla^T \mathbf{g}_k \cdot \left(x'_{k+1} - x_{k+1} \right) = h'_{k+1} \tag{8.10}$$

x_{k+1} can be obtained from the above equation by least square method:

$$\begin{aligned} &\nabla^T \mathbf{g}_k \cdot \left(x'_{k+1} - x_{k+1} \right) = \left(h'_{k+1} - h_{k+1} \right) \\ &x_{k+1} = x'_{k+1} - \nabla \mathbf{g}_k \left(\nabla^T \mathbf{g}_k \cdot \nabla \mathbf{g}_k \right)^{-1} h'_{k+1} \end{aligned} \tag{8.11}$$

Iteration is required for X_{k+1} to return to the constraint plane:

$$\left(x_{k+1} \right)_{j+1} = \left(x_{k+1} - \nabla \mathbf{g}_k \left(\nabla^T \mathbf{g}_k \cdot \nabla \mathbf{g}_k \right)^{-1} h_{k+1} \right)_j \tag{8.12}$$

Another iteration to compute X_{k+1} is:

$$\left(x_{k+1} \right)_{j+1} = \left(x_{k+1} - \nabla \mathbf{g}_{k+1} \left(\nabla^T \mathbf{g}_{k+1} \cdot \nabla \mathbf{g}_{k+1} \right)^{-1} h_{k+1} \right)_j \tag{8.13}$$

Ten bar Truss is used as example for this method. Appendix code 29 is the matlab code for solving FORM safety-index problem by Gradient Projection Method. Table 8-1 is a computation example for rod 3 with a starting point (u1=-1.8791, 1, 1). It is shown that it takes 15 steps for the iteration to converge. The final safety-index is 1.87571, corresponding to probability of failure 0.03035; the safety-index error is 0.006%.

Table 8-1: Searching iterations by gradient projection method

Itera tion	beta	Pf	error	u1	u2	u3	x1	x2	x3
1	2.35181	0.00934	/	-1.8791	1	1	6.242E-05	3.600E-05	1.200E-04
2	2.18312	0.01451	7.73E-02	-1.8783	0.7841	0.7893	6.243E-05	3.470E-05	1.158E-04
3	2.06393	0.01951	5.77E-02	-1.8777	0.6009	0.6107	6.245E-05	3.361E-05	1.122E-04
4	1.98592	0.02352	3.93E-02	-1.8772	0.4514	0.4652	6.246E-05	3.271E-05	1.093E-04
5	1.93813	0.02630	2.47E-02	-1.8767	0.3336	0.3506	6.247E-05	3.200E-05	1.070E-04
6	1.91028	0.02805	1.46E-02	-1.8764	0.2432	0.2629	6.247E-05	3.146E-05	1.053E-04
7	1.89457	0.02907	8.29E-03	-1.8762	0.1751	0.1968	6.248E-05	3.105E-05	1.039E-04
8	1.88589	0.02965	4.60E-03	-1.8760	0.1243	0.1477	6.248E-05	3.075E-05	1.030E-04
9	1.88116	0.02998	2.52E-03	-1.8758	0.0868	0.1115	6.248E-05	3.052E-05	1.022E-04
10	1.87859	0.03015	1.37E-03	-1.8757	0.0591	0.0848	6.249E-05	3.035E-05	1.017E-04
11	1.87720	0.03025	7.40E-04	-1.8757	0.0387	0.0652	6.249E-05	3.023E-05	1.013E-04
12	1.87645	0.03030	3.99E-04	-1.8756	0.0237	0.0508	6.249E-05	3.014E-05	1.010E-04
13	1.87605	0.03032	2.15E-04	-1.8756	0.0127	0.0402	6.249E-05	3.008E-05	1.008E-04
14	1.87583	0.03034	1.16E-04	-1.8755	0.0046	0.0325	6.249E-05	3.003E-05	1.006E-04
15	1.87571	0.03035	6.23E-05	-1.8755	-0.0013	0.0268	6.249E-05	2.999E-05	1.005E-04

Table 8-2 shows the results of Gradient Projection Method algorithm with different starting points for rod 3. Since the starting point are located on the constraint surface, a matlab function solve_u1.m is also provided to compute a third coordinate value when 2 are given, thus a starting point can be obtained. Table 18 provides cases of 11 different starting points, some of which are close to the minimum-distance-point, some are far. After certain number of iterations, all of these starting points converge to a same final point, and safety index β all converges to 1.8757, corresponding to probability of failure 0.0303. The errors are relatively very low. This table shows that Gradient Projection Method achieves very precise result for FORM problems.

Table 8-3 shows the results of different algorithms for computing Pf. It is shown that the FOSM Gradient Projection method, HL reliability index method and SORM method all achieve similar results.

Table 8-2: Converging from different starting points by gradient projection method

item	Starting Point			steps	initial beta	final beta	Pf	error
	u1	u2	u3					
1	-1.8757	0	0	2	1.8757	1.8756	0.0304	3.00E-05
2	-1.8706	0	1	13	2.1211	1.8758	0.0303	8.00E-05
3	-1.8847	1	0	13	2.1336	1.8758	0.0303	9.00E-05
4	-1.8791	1	1	15	2.3518	1.8757	0.0303	6.00E-05
5	-1.883	0	-1	13	2.1321	1.8758	0.0303	9.00E-05
6	-1.8668	0	2	16	2.7359	1.8758	0.0303	9.00E-05
7	-1.8937	0	-2	16	2.7543	1.8758	0.0303	9.00E-05
8	-1.8687	-2	-2	18	3.39	1.8758	0.0303	9.00E-05
9	-1.8823	2	2	18	3.3975	1.8758	0.0303	1.00E-04
10	-1.852	-2	2	18	3.3808	1.8758	0.0303	9.00E-05
11	-1.9118	2	-2	18	3.4139	1.8758	0.0303	1.00E-04

Table 8-3: Comparison between the results from gradient projection method and other methods

	Precise MCS	Kriging model	MVFOSM reliability index		Gradient Projection		HL reliability index		SORM Breitung
Sample size	1e4	1e6	/	/	/	/	/	/	/
Running time (s)	500	1600	/	/	/	/	/	/	/
Rod (Ai indept)	P_f	P_f	β	P_f	β	P_f	β_{HL}	P_f	P_f
3	0.0330	0.0343	2.9953	0.0014	1.8758	0.0303	1.8756	0.0304	0.0303
7	0.0322	0.0382	2.8889	0.0019	1.8389	0.0330	1.8388	0.0330	0.0329

8.1.2 Reduced Gradient Method

If the there's only an objective function without any equation constraint, then all the variables in the objective function are free variables and are independent of each other. However once there are constraints, equation constraints or active inequality constraints, the degree of freedom of the system decreases. The number of active constraints equals to the DOF decrease number. Suppose there are totally n variables and m constraints, then the DOF decrease will be m and there will be (n-m) free variables.

There will be m variables that are not free and can be represented by the (n-m) free variables. The (n-m) free variables are called Decision Variables and the m non-free variables are called State Variables.

Reduced Gradient Method splits the overall variables into 2 categories: Decision Variables and State Variables. The gradient based search steps are carried out based on the decision variables only. Since there are deterministic relations between state variables and decision variables, with each step on the decision variables, the steps on the state variables can be computed accordingly. More details can be found in [Papalambros, P.Y., 2000].

Some notations:

d_i : decision variables, i=1…(n-m);

s_i : state variables, i=1…m;

$\partial \mathbf{x} = \mathbf{x_{k+1}} - \mathbf{x_k}$: a perturbation on vector X;

$z(\mathbf{d}) = f(\mathbf{d}, \mathbf{s})$: Objective function;

$\mathbf{h}(\mathbf{d}, \mathbf{s}) = \mathbf{0}$: Equation constraints, mx1 matrix (column vector);

$$\frac{\partial f}{\partial \mathbf{d}} = \left[\frac{\partial f}{\partial d_1} \cdots\cdots \frac{\partial f}{\partial d_{n-m}} \right], \quad \frac{\partial z}{\partial \mathbf{d}} = \left[\frac{\partial z}{\partial d_1} \cdots\cdots \frac{\partial z}{\partial d_{n-m}} \right] \tag{8.14}$$

$\frac{\partial f}{\partial \mathbf{s}}, \frac{\partial z}{\partial \mathbf{s}}$: 1xm matrix (row vector);

$$\frac{\partial \mathbf{h}}{\partial \mathbf{d}} = \begin{bmatrix} \frac{\partial h_1}{\partial d_1} & \cdots\cdots & \frac{\partial h_1}{\partial d_{n-m}} \\ \cdots & \cdots\cdots & \cdots \\ \frac{\partial h_m}{\partial d_1} & \cdots\cdots & \frac{\partial h_m}{\partial d_{n-m}} \end{bmatrix} \tag{8.15}$$

$$\frac{\partial \mathbf{h}}{\partial \mathbf{s}} = \begin{bmatrix} \frac{\partial h_1}{\partial s_1} & \cdots\cdots & \frac{\partial h_1}{\partial s_m} \\ \cdots & \cdots\cdots & \cdots \\ \frac{\partial h_m}{\partial s_1} & \cdots\cdots & \frac{\partial h_m}{\partial s_m} \end{bmatrix} \tag{8.16}$$

A step on the decision variables toward the decreasing direction can be expressed as:

$$\mathbf{d}_{k+1} = \mathbf{d}_k - \alpha_k \cdot \left(\left(\frac{\partial z}{\partial \mathbf{d}} \right)_k \right)^T \text{ or } \partial \mathbf{d} = -\alpha_k \cdot \left(\left(\frac{\partial z}{\partial \mathbf{d}} \right)_k \right)^T \tag{8.17}$$

The corresponding step on the state variables is obtained:

$$\begin{aligned} &\mathbf{h}(\mathbf{x}) = \mathbf{0} \\ &\Rightarrow \frac{\partial \mathbf{h}}{\partial \mathbf{x}} \cdot \partial \mathbf{x} = \mathbf{0} \Rightarrow \cdots\cdots \\ &\Rightarrow \mathbf{s'}_{k+1} = \mathbf{s}_k - \left(\frac{\partial \mathbf{h}}{\partial \mathbf{s}} \right)^{-1} \left(\frac{\partial \mathbf{h}}{\partial \mathbf{d}} \right) \partial \mathbf{d} = \mathbf{s}_k + \alpha_k \left(\frac{\partial \mathbf{h}}{\partial \mathbf{s}} \right)_k^{-1} \left(\frac{\partial \mathbf{h}}{\partial \mathbf{d}} \right)_k \left(\frac{\partial z}{\partial \mathbf{d}} \right)_k^T \end{aligned} \tag{8.18}$$

Since $\mathbf{s'}_{k+1}$ is usually not on the constraint plane, it's necessary to return it to the constraint plane by this iteration:

$$\begin{aligned} &\frac{\partial \mathbf{h}}{\partial \mathbf{s}} \cdot (\mathbf{s'}_{k+1} - \mathbf{s}_{k+1}) = \partial \mathbf{h} = \mathbf{h}(\mathbf{d}_{\mathbf{k+1}}, \mathbf{s'}_{\mathbf{k+1}}) - \mathbf{h}(\mathbf{d}_{\mathbf{k+1}}, \mathbf{s}_{\mathbf{k+1}}) = \mathbf{h}(\mathbf{d}_{\mathbf{k+1}}, \mathbf{s'}_{\mathbf{k+1}}) - 0 \\ &\Rightarrow \mathbf{s}_{k+1} = \mathbf{s'}_{k+1} - \left(\frac{\partial \mathbf{h}}{\partial \mathbf{s}} \right)^{-1} \cdot \mathbf{h}(\mathbf{d}_{\mathbf{k+1}}, \mathbf{s}_{\mathbf{k+1}}) \end{aligned} \tag{8.19}$$

Usually one or two such returning operation will achieve very high precision. In the ten-bar-truss example, 3 returning steps are used.

The total process for find the minimum value of the objective function can be listed in the following steps:

1) Choose an initial point on the constraint surface and select an appropriate step size α;

2) Compute $\frac{\partial \mathbf{h}}{\partial \mathbf{d}}, \frac{\partial \mathbf{h}}{\partial \mathbf{s}}, \frac{\partial f}{\partial \mathbf{d}}, \frac{\partial f}{\partial \mathbf{s}}$ values at the given point;

3) Compute the next point $\mathbf{d}_{k+1}$ for decision variables;

4) Compute the next point $\mathbf{s'}_{k+1}$ for state variables;

5) Returning the state variable point to the constraint plane, in another word, find $\mathbf{s}_{k+1}$;

6) $(\mathbf{d}_{k+1}, \mathbf{s}_{k+1})$ gives a new point on the constraint plane, evaluate the objective function at this point and compare with the result of the previous step, if not converging, go to step 2); if it converges, terminate the process.

Ten bar truss case is used as an example for this method. Table 8-4 shows the iteration process for rod 3 given an initial point at (-1.8722, -1, -1). It takes 15 iterations to reach an error of 6e-5.

Table 8-4: Rod 3 searching iterations by reduced gradient algorithm

Iteration	beta	Pf	error	u1	u2	u3	x1	x2	x3
1	2.34633	0.00948	1.00E+00	-1.8722	-1.0000	-1.0000	6.26E-05	2.40E-05	8.00E-05
2	2.17909	0.01466	7.67E-02	-1.8729	-0.7914	-0.7838	6.25E-05	2.53E-05	8.43E-05
3	2.06118	0.01964	5.72E-02	-1.8735	-0.6144	-0.6008	6.25E-05	2.63E-05	8.80E-05
4	1.98418	0.02362	3.88E-02	-1.8740	-0.4701	-0.4519	6.25E-05	2.72E-05	9.10E-05
5	1.93709	0.02637	2.43E-02	-1.8743	-0.3565	-0.3349	6.25E-05	2.79E-05	9.33E-05
6	1.90968	0.02809	1.44E-02	-1.8746	-0.2693	-0.2453	6.25E-05	2.84E-05	9.51E-05
7	1.89424	0.02910	8.15E-03	-1.8748	-0.2036	-0.1779	6.25E-05	2.88E-05	9.64E-05
8	1.88571	0.02967	4.52E-03	-1.8750	-0.1547	-0.1279	6.25E-05	2.91E-05	9.74E-05
9	1.88106	0.02998	2.47E-03	-1.8751	-0.1185	-0.0909	6.25E-05	2.93E-05	9.82E-05
10	1.87853	0.03015	1.34E-03	-1.8752	-0.0918	-0.0637	6.25E-05	2.94E-05	9.87E-05
11	1.87717	0.03025	7.26E-04	-1.8753	-0.0721	-0.0437	6.25E-05	2.96E-05	9.91E-05
12	1.87644	0.03030	3.92E-04	-1.8753	-0.0577	-0.0290	6.25E-05	2.97E-05	9.94E-05
13	1.87604	0.03032	2.11E-04	-1.8754	-0.0471	-0.0182	6.25E-05	2.97E-05	9.96E-05
14	1.87583	0.03034	1.14E-04	-1.8754	-0.0393	-0.0104	6.25E-05	2.98E-05	9.98E-05
15	1.87571	**0.03035**	6.11E-05	-1.8754	-0.0335	-0.0046	6.25E-05	2.98E-05	9.99E-05

Table 8-5 shows the iteration process for rod 7 given an initial point at (1, 2, -1.8624). It takes 15 iterations to reach an error of 6e-5.

8.2 Correlated HL method

HL method transforms the Gaussian distributed random variables to standard Gaussian distributions, and take the first order Taylor expansion at the MPP point. HL makes great improvement compared with

MVFOSM. However, HL method does not consider the correlations between random variables. This can be seen from the form of HL safety index:

$$\beta = \frac{g(X^*) - \sum_{i=1}^{n} \frac{\partial g(X^*)}{\partial x_i} \sigma_{x_i} u_i^*}{\sqrt{\sum_{i=1}^{n} \left(\frac{\partial g(X^*)}{\partial x_i} \sigma_{x_i} \right)^2}} \qquad (8.20)$$

Table 8-5: Rod 7 searching iterations by reduced gradient algorithm

Itera tion	beta	Pf	error	u1	u2	u3	x1	x2	x3
1	2.91008	0.00181	1.00E+00	1.0000	2.0000	-1.8624	1.20E-04	4.20E-05	6.28E-05
2	2.62183	0.00437	1.10E-01	0.8288	1.6528	-1.8588	1.17E-04	3.99E-05	6.28E-05
3	2.38148	0.00862	1.01E-01	0.6715	1.3335	-1.8554	1.13E-04	3.80E-05	6.29E-05
4	2.19377	0.01413	8.56E-02	0.5314	1.0488	-1.8521	1.11E-04	3.63E-05	6.30E-05
5	2.05810	0.01979	6.59E-02	0.4113	0.8043	-1.8492	1.08E-04	3.48E-05	6.30E-05
6	1.96770	0.02455	4.59E-02	0.3124	0.6030	-1.8468	1.06E-04	3.36E-05	6.31E-05
7	1.91172	0.02796	2.93E-02	0.2342	0.4434	-1.8448	1.05E-04	3.27E-05	6.31E-05
8	1.87900	0.03012	1.74E-02	0.1742	0.3207	-1.8432	1.03E-04	3.19E-05	6.31E-05
9	1.86061	0.03140	9.88E-03	0.1291	0.2284	-1.8420	1.03E-04	3.14E-05	6.32E-05
10	1.85052	0.03212	5.45E-03	0.0957	0.1599	-1.8411	1.02E-04	3.10E-05	6.32E-05
11	1.84507	0.03251	2.95E-03	0.0712	0.1095	-1.8404	1.01E-04	3.07E-05	6.32E-05
12	1.84215	0.03273	1.59E-03	0.0532	0.0725	-1.8399	1.01E-04	3.04E-05	6.32E-05
13	1.84059	0.03284	8.48E-04	0.0401	0.0455	-1.8396	1.01E-04	3.03E-05	6.32E-05
14	1.83976	0.03290	4.52E-04	0.0305	0.0258	-1.8393	1.01E-04	3.02E-05	6.32E-05
15	1.83931	0.03293	2.40E-04	0.0236	0.0114	-1.8391	1.00E-04	3.01E-05	6.32E-05
16	1.83908	0.03295	1.28E-04	0.0185	0.0009	-1.8390	1.00E-04	3.00E-05	6.32E-05
17	1.83895	**0.03296**	6.79E-05	0.0148	-0.0068	-1.8389	1.00E-04	3.00E-05	6.32E-05

In which X* is the MPP point, U* is the mapping of X* to U space (Standar Gaussian).

The denominator is the standard variance of the limit-state function, the form of which implies uncorrelated random variables. The expression of the variance of the transformed limit state is:

$$
\begin{aligned}
&Var\left[\hat{g}(U)\right] = Var\left[\hat{g}(U^*) + \nabla^T \hat{g}(U^*) \cdot \left(U - U^*\right)\right] \\
&= Var\left[\nabla^T \hat{g}(U^*) \cdot U\right] \\
&= Var\left[\sum_{i=1}^{n} \frac{\partial g(U^*)}{\partial u_i} u_i\right] \\
&= Var\left[\sum_{i=1}^{n} \left(\frac{\partial g(X^*)}{\partial x_i} \sigma_{x_i} u_i\right)\right]
\end{aligned}
\tag{8.21}
$$

In case of uncorrelated random variables, as is the case for HL method, the variance of the limit-state function is:

$$
Var\left[\hat{g}(U)\right] = \sum_{i=1}^{n} \left(\frac{\partial g(X^*)}{\partial x_i} \sigma_{x_i}\right)^2 \tag{8.22}
$$

When the random variables are correlated, the variance becomes:

$$
Var\left[\hat{g}(U)\right] = \sum_{i=1}^{n} \left(\frac{\partial g(X^*)}{\partial x_i} \sigma_{x_i}\right)^2 + 2\sum_{i<j} COV\left(\frac{\partial g(X^*)}{\partial x_i} \sigma_{x_i} u_i, \frac{\partial g(X^*)}{\partial x_j} \sigma_{x_j} u_j\right) \tag{8.23}
$$

In this case since n=3, the variance becomes

$$
\begin{aligned}
&Var\left[\hat{g}(U)\right] = \sum_{i=1}^{n} \left(\frac{\partial g(X^*)}{\partial x_i} \sigma_{x_i}\right)^2 + 2\{\frac{\partial g(X^*)}{\partial x_1} \sigma_{x_1} \cdot \frac{\partial g(X^*)}{\partial x_2} \sigma_{x_2} \cdot COV(u_1, u_2) \\
&+ \frac{\partial g(X^*)}{\partial x_1} \sigma_{x_1} \cdot \frac{\partial g(X^*)}{\partial x_3} \sigma_{x_3} \cdot COV(u_1, u_3) + \frac{\partial g(X^*)}{\partial x_2} \sigma_{x_2} \cdot \frac{\partial g(X^*)}{\partial x_3} \sigma_{x_3} \cdot COV(u_2, u_3)\}
\end{aligned}
\tag{8.24}
$$

Since $X_i = \mu_i + \sigma_{x_i} u_i$, the correlation coefficient between x_i and x_j can be expressed as:

$$
\begin{aligned}
\rho_{ij}(X_i, X_j) &= \frac{COV(X_i, X_j)}{\sigma_{X_i} \cdot \sigma_{X_j}} \\
&= \frac{COV(\mu_i + \sigma_{x_i} u_i, \mu_j + \sigma_{x_j} u_j)}{\sigma_{X_i} \cdot \sigma_{X_j}} \\
&= \frac{\sigma_{x_i} \cdot \sigma_{x_j} \cdot COV(u_i, u_j)}{\sigma_{X_i} \cdot \sigma_{X_j}} \\
&= COV(u_i, u_j)
\end{aligned}
\tag{8.25}
$$

Thus plug $COV(u_i, u_j) = \rho_{ij}(X_i, X_j)$ into equation (8.24):

$$Var[\hat{g}(U)] = \sum_{i=1}^{3}\left(\frac{\partial g(X^*)}{\partial x_i}\sigma_{x_i}\right)^2 + 2\{\frac{\partial g(X^*)}{\partial x_1}\sigma_{x_1}\cdot\frac{\partial g(X^*)}{\partial x_2}\sigma_{x_2}\cdot\rho_{12}$$
$$+\frac{\partial g(X^*)}{\partial x_1}\sigma_{x_1}\cdot\frac{\partial g(X^*)}{\partial x_3}\sigma_{x_3}\cdot\rho_{13}+\frac{\partial g(X^*)}{\partial x_2}\sigma_{x_2}\cdot\frac{\partial g(X^*)}{\partial x_3}\sigma_{x_3}\cdot\rho_{23}\} \quad (8.26)$$

For any number of random variables, general form of $Var[\hat{g}(U)]$ becomes:

$$Var[\hat{g}(U)] = \sum_{i=1}^{n}\left(\frac{\partial g(X^*)}{\partial x_i}\sigma_{x_i}\right)^2 + 2\sum_{i<j}\left(\frac{\partial g(X^*)}{\partial x_i}\sigma_{x_i}\cdot\frac{\partial g(X^*)}{\partial x_j}\sigma_{x_j}\cdot\rho_{ij}\right) \quad (8.27)$$

Steps for computing correlated HL safety index:

1) Compute correlation coefficient of each pair of RV's:

$$\rho_{ij}(X_i, X_j) = \frac{COV(X_i, X_j)}{\sigma_{X_i}\cdot\sigma_{X_j}} \quad (8.28)$$

2) Compute correlated safety index β by MVFOSM. For the first step, the computing point X* is chosen to be the mean value point μ_X

$$\beta = \frac{g(\mu_X)}{\sqrt{\sum_{i=1}^{n}\left(\frac{\partial g(\mu_X)}{\partial x_i}\sigma_{x_i}\right)^2 + 2\sum_{i<j}\left(\frac{\partial g(\mu_X)}{\partial x_i}\sigma_{x_i}\cdot\frac{\partial g(\mu_X)}{\partial x_j}\sigma_{x_j}\cdot\rho_{ij}\right)}} \quad (8.29)$$

3) Given a point X* and corresponding U*, compute $\cos\theta_{u_i}$

$$\cos\theta_{u_i} = -\frac{\frac{\partial \hat{g}(U^*)}{\partial u_i}}{|\nabla \hat{g}(U^*)|} = -\frac{\frac{\partial g(X^*)}{\partial x_i}\sigma_{x_i}}{\sqrt{\sum_{i=1}^{n}\left(\frac{\partial g(X^*)}{\partial x_i}\sigma_{x_i}\right)^2}} \quad (8.30)$$

4) Compute the new design point U* and the corresponding X*

$$\begin{aligned} u_i^* &= \beta\cos\theta_{u_i} \\ x_i^* &= \mu_{x_i} + \sigma_{x_i}u_i^* \end{aligned} \quad (8.31)$$

5) Compute the new β at the new X* point and new U* point

$$\beta = \frac{g(X^*) - \sum_{i=1}^{n} \frac{\partial g(X^*)}{\partial x_i} \sigma_{x_i} u_i^*}{\sqrt{\sum_{i=1}^{n} \left(\frac{\partial g(X^*)}{\partial x_i} \sigma_{x_i} \right)^2 + 2\sum_{i<j} \left(\frac{\partial g(X^*)}{\partial x_i} \sigma_{x_i} \cdot \frac{\partial g(X^*)}{\partial x_j} \sigma_{x_j} \cdot \rho_{ij} \right)}} \qquad (8.32)$$

6) Repeat step 3)—5) until β converges, or if not converging then stop the loop at a given iteration number.

To verify the efficacy of correlated HL method, both MCS and correlated HL algorithm are implemented with coefficient of dependence (Pearson's rho) 0.8 for each pair of RV's. The result is shown in Table 8-6:

Table 8-6: Comparison between the results of correlated HL method and MCS

	MCS (1e5 samples 55 minutes)	HL reliability index	
Rod (Ai correlated)	P_f	$\beta_{HL\text{-}corr}$	P_f
3	0.0316	1.8705	0.0307
7	0.0343	1.8215	0.0343

For MCS simulation, the covariance matrix is selected as

$$C = \begin{bmatrix} \sigma_1^2 & \rho_{12}\sigma_1\sigma_2 & \rho_{13}\sigma_1\sigma_3 \\ \rho_{12}\sigma_1\sigma_2 & \sigma_2^2 & \rho_{23}\sigma_2\sigma_3 \\ \rho_{13}\sigma_1\sigma_3 & \rho_{23}\sigma_2\sigma_3 & \sigma_3^2 \end{bmatrix} \qquad (8.33)$$
$$\rho_{12} = \rho_{13} = \rho_{23} = \rho = 0.8$$

8.3 Summary

In this chapter, two algorithms are introduced: 1) solving reliability index by nonlinear programming (NLP) algorithms and 2) correlated HL method.

Reliability index can be interpreted as a nonlinear optimization problem with the objective to minimize the distance under an equality constraint. Two NLP algorithms (gradient projection method and reduced gradient method) are applied to compute reliability index. Results are compared with that of other RBDO algorithms. Compared with other RBDO algorithms, NLP based algorithm is more intuitive and easy to understand.

Correlated HL method is an algorithm based on HL method that considers the correlation between random variables. For this algorithm, more cases should be studied to validate it.

CHAPTER 9: ILLUSTRATED EXAMPLE OF FLOOR GRID STRUCTURAL OPTIMIZATION

In this chapter, the previously explained methods and algorithms will be applied to find the optimal solution of the aircraft floor grid optimization case. The optimization model is:

$$\begin{aligned} &\text{minimize} && W_{total}(\mathbf{X}) = W_{grid}(\mathbf{X}) + W_{board}(\mathbf{X}) \\ &\text{subject to} && \boldsymbol{\beta}(\mathbf{X}, \mathbf{R}) \geq \boldsymbol{\beta}_0 \end{aligned} \tag{9.1}$$

In which **β** is a vector of reliability indices; $\boldsymbol{\beta}_0$ is the minimum permissible **β** value. **X** is a vector of design variables, **R** represents random variable distributions and correlations.

The objective function is to minimize the total weight of floor board and floor grid; the constraint is to guarantee the minimum permissible reliability.

This problem will be studied in Case I and Case II. Case I considers only one random variable; Case II considers 3 random variables. In Case I the reliability constraints all have analytical form and the optimization problem is easy to solve, this provide a good example to show the process of doing reliability-based floor grid layout optimization; In Case II the reliability constraints do not have analytical form thus response surface will be applied.

9.1 Case I

9.1.1 Hypothesis and Simplifications

In engineering practice the real world situations are usually very complicated. The deck in the figure looks very intact and the strength members are very continuous. However in practice there may be many openings and holes on the floor board for cables or pipes going through, or ladder opening, etc. Longitudinal stiffeners may not be so uniformly distributed due to the uneven arrangement of cabins, local strengthening may be required on special places. If all these factors are taken into consideration in the initial design stage, the design would be impossible to move forward. Reasonable hypothesis and simplifications must be made for optimization work.

Some key hypothesis and simplifications are listed below:

1) All longitudinal tracks are regarded as simply supported beams (one side hinged the other side roller supported).

2) Loads on longitudinal tracks are evenly distributed.

3) Plate inside any grid area is regarded as an intact plate.

4) The shape design of the cross section of longitudinal tracks are related to stability criteria and buckling verification that are far beyond the scope of this thesis. For simplicity, the cross section of longitudinal members is regarded as rectangular with the ratio of width and height 1/3.

5) Load on the floor is evenly distributed inside each grid and follows normal distribution:

$q \sim N(2200{,}400)$ Pa.

6) Plates are considered fixed on 4 sides. Plate vertical deflection equation:

$$w = k\frac{qd^4}{Et^3}$$

k: deflection coefficient, set as 0.05;

q: evenly distributed load (N/m^2);

d: smaller width of rectangular plate (m);

E: Young's elastic modulus (Pa);

t: Plate thickness (m);

7) Material is selected as Aluminum alloy 6061-T6, the material properties are listed below:

ρ=2.7e3kg/m^3;

σ_{yield}=241 MPa;

E=69 GPa;

8) Since the number of track columns can only be selected as discrete positive integers, the track space d can only be selected as discrete values. However most optimization methods are based on continuous variables, as such track spacing will be taken as continuous design variable in the initial optimization. Discrete effect will be considered in the further steps.

9) Optimization of shape and spacing of transverse beams is not included in this study. Only plate and tracks will be considered.

9.1.2 Optimization Model

Minimize: (total mass (or material volume) of floor plate and longitudinal tracks)

Subject to:

1) $P\{g_1(X) < 0\} < 1e-5$	track stress reliability requirement
2) $P\{g_2(X) < 0\} < 1e-5$	plate deflection reliability requirement
3) $t \geq 3mm$	plate corrosion margin requirement
4) $0.3m \leq d \leq 0.75m$	track spacing limit

There are 3 design variables selected for this problem:

d: track spacing;

t: plate thickness;

b: track section size;

For this study the shape and size of the transverse beams are not included. The total weight only includes the weight of all longitudinal tracks and sum of all floor board plate.

$$\begin{aligned} W &= W_{track} + W_{plate} \\ &= bh \cdot D \cdot \frac{W_{fuselage}}{d} + W_{fuselage} \cdot D \cdot t \end{aligned}$$

In which:

h: track cross section height, set as 3b;

D: Transverse beam space, 0.75m;

$W_{fuselage}$: fuselage width, 6m;

The limit-state function for track stress:

$$\begin{aligned} g_1(X) &= \sigma_{yield} - \sigma \\ &= \sigma_{yield} - \frac{6M}{bh^2} \\ &= \sigma_{yield} - \frac{3qdD^2}{4bh^2} \end{aligned} \tag{9.2}$$

The limit-state function for plate deflection:

$$\begin{aligned} g_2(X) &= \delta_{allow} - \delta \\ &= \delta_{allow} - k\frac{qd^4}{Et^3} \end{aligned} \tag{9.3}$$

In which

δ_{allow}= 5mm.

Since there is only one random variable in this example, the relationship between Pf and design variables becomes easy.

$$P\{g_1(X)<0\}=P\left\{\sigma_{yield}-\frac{3qdD^2}{4bh^2}<0\right\}$$
$$=P\left\{q>\sigma_{yield}\cdot\frac{4bh^2}{3dD^2}\right\}$$
$$=P\left\{\frac{q-\mu_q}{\sigma_q}>\frac{\sigma_{yield}\cdot\frac{4bh^2}{3dD^2}-\mu_q}{\sigma_q}\right\}$$

$P\{g_1(X)<0\}<1e-5$ thus can be substituted by

$$\frac{\sigma_{yield}\cdot\frac{4bh^2}{3dD^2}-\mu_q}{\sigma_q}>-\Phi^{-1}(1e-5)$$

Or more concisely

$$\sigma_{yield}\cdot\frac{4bh^2}{3dD^2}>4.2649\sigma_q+\mu_q$$

For constraint 2,

$$P\{g_2(X)<0\}=P\left\{\delta_{allow}-k\frac{qd^4}{Et^3}<0\right\}$$
$$=P\left\{q>\delta_{allow}\cdot\frac{Et^3}{kd^4}\right\}$$
$$=P\left\{\frac{q-\mu_q}{\sigma_q}>\frac{\delta_{allow}\cdot\frac{Et^3}{kd^4}-\mu_q}{\sigma_q}\right\}$$

$P\{g_2(X)<0\}<1e-5$ can be substituted by:

$$\frac{\delta_{allow} \cdot \frac{Et^3}{kd^4} - \mu_q}{\sigma_q} > -\Phi^{-1}(1e-5)$$

Or

$$\delta_{allow} \cdot \frac{Et^3}{kd^4} > 4.2649\sigma_q + \mu_q$$

The final optimization model becomes:

Minimize:

$$W = 3D \cdot W_{fuselage} \cdot \left(\frac{b^2}{d}\right) + W_{fuselage} \cdot D \cdot (t) \tag{9.4}$$

by determining design variables b, d, t

Subject to :

$$\begin{aligned} &1) \quad \sigma_{yield} \cdot \frac{4bh^2}{3dD^2} > 4.2649\sigma_q + \mu_q \\ &2) \quad \delta_{allow} \cdot \frac{Et^3}{kd^4} > 4.2649\sigma_q + \mu_q \\ &3) \quad t \geq 0.003m \\ &4) \quad 0.3m \leq d \leq 0.75m \end{aligned} \tag{9.5}$$

The constraints of this optimization problem are all inequality constraints. As a tentative step, first suppose that the first two constraints are active while the rest constraints are redundant. The problem can be solved by Gradient Projection method. Select the starting point as $b = 0.007; \;\; d = 0.5; \;\; t = 0.0035$, step size α as 1.5, the algorithm converges after 15 steps to point

$b = 0.0034; \;\; d = 0.0533; \;\; t = 1.66e-4$:

The searching process is shown in Table 9-1:

Table 9-1: Tentative searching iterations

step	b(m)	d(m)	t(m)	W(m3)
1	0.007	0.5	3.50E-03	0.017073
2	0.006931	0.43833	2.75E-03	0.01387466
3	0.006629	0.383468	2.30E-03	0.01191789
4	0.006314	0.331401	1.90E-03	0.01016134
5	0.005986	0.282335	1.53E-03	0.00860821
6	0.005643	0.236532	1.21E-03	0.00726285
7	0.005285	0.194336	9.31E-04	0.0061308
8	0.004914	0.156209	6.96E-04	0.00521876
9	0.004535	0.122799	5.05E-04	0.00453348
10	0.004164	0.095006	3.59E-04	0.00407707
11	0.003831	0.074004	2.57E-04	0.0038338
12	0.003589	0.060839	1.98E-04	0.0037486
13	0.003471	0.055047	1.73E-04	0.00373427
14	0.00344	0.053565	1.67E-04	0.00373342
15	**0.003435**	**0.053334**	**1.66E-04**	0.0037334

This point violates constraint 3 and 4. Thus constraint 3 or 4 or both should be also active. However, it's difficult to determine which constraints are active before the optimal point is finally found. This is a paradox. To solve this problem, heuristic searching algorithm is applied in the next section.

9.1.3 Searching Algorithm

For aircraft floor grid problem, first write all constraint in the standard form of $h_i(X) > 0$, five standard inequality constraints are obtained:

1) $\sigma_{yield} \cdot \frac{4bh^2}{3dD^2} - \left(4.2649\sigma_q + \mu_q\right) > 0$
2) $\delta_{allow} \cdot \frac{Et^3}{kd^4} - \left(4.2649\sigma_q + \mu_q\right) > 0$
3) $t - 0.003 > 0$
4) $d - 0.3 > 0$
5) $0.75 - d > 0$

Heuristic gradient descent searching algorithms is used in this problem, the searching iterations are shown in Table 9-2. Starting point is selected as: $b = 0.01; d = 0.4; t = 0.0035$, step size set as fixed value 0.001, it takes 9 steps to converge to the optimal point: $b = 0.00708; d = 0.46733; t = 0.0030$. Flag1 to flag5 denote the activation of constraint 1 to constraint 5 at each step. It's shown that in the beginning, constraint 1 is not active, however in the end steps constraint 1 becomes active.

Table 9-2: Heuristic gradient based searching iterations

Step	b(m)	d(m)	t(m)	Objective Function(m3)	error	flag 1	flag 2	flag 3	flag4	flag5
1	0.0100000	0.4000000	0.0035	/	/	/	/	/	/	/
2	0.0093250	0.4668622	0.003	0.019125	/	0	1	1	0	0
3	0.0087857	0.4673274	0.003	0.016014449	0.1942341	0	1	1	0	0
4	0.0082781	0.4673274	0.003	0.015729801	0.0180961	0	1	1	0	0
5	0.0077998	0.4673274	0.003	0.015479589	0.016164	0	1	1	0	0
6	0.0073492	0.4673274	0.003	0.015257454	0.0145591	0	1	1	0	0
7	0.0070809	0.4673274	0.003	0.015060245	0.0130946	1	1	1	0	0
8	0.0070809	0.4673274	0.003	0.014948421	0.0074807	1	1	1	0	0
9	0.0070809	0.4673274	0.003	0.014948421	6.72E-13	1	1	1	0	0

To prevent the convergence to local optimal, 8 different starting points are chosen that covers most of the design space. All of these starting point converge to the same optimal point: $b = 0.00708; d = 0.46733; t = 0.0030$. Difference lies in the iteration number. The result is shown in Table 9-3:

Table 9-3: Convergence from different starting points

Starting Point			Convergence Point				step numbers
b(m)	d(m)	t(m)	b(m)	d(m)	t(m)	f (m3)	
0.01	0.4	0.0035	0.0071	0.4673	0.003	0.0149	9
0.02	0.4	0.0035	0.0071	0.4673	0.003	0.0149	21
0.01	0.6	0.0035	0.0071	0.4673	0.003	0.0149	10
0.02	0.6	0.0035	0.0071	0.4673	0.003	0.0149	21
0.01	0.4	0.004	0.0071	0.4673	0.003	0.0149	9
0.02	0.4	0.004	0.0071	0.4673	0.003	0.0149	21
0.01	0.6	0.004	0.0071	0.4673	0.003	0.0149	10
0.02	0.6	0.004	0.0071	0.4673	0.003	0.0149	21

Drawback of this algorithm: Since the nature of the terminating criteria, searching would stop at any local minimum it first encounters. To avoid converging to local minimum, lots of starting points are required.

The optimal track spacing d=0.46733 correspond to the track number: $n = W_{fuselage} / d = 12.84$, round it to n=12 and n=13. The corresponding track space becomes $d_1 = 0.5; d_2 = 0.462$. Under both cases the track spacing d will be taken as known and the optimization problem need be reevaluated to determine the other two design variables b and t.

For d=0.5m, the new optimization problem becomes:

Minimize:

$$W = 3D \cdot \frac{W_{fuselage}}{d} \cdot \left(b^2\right) + W_{fuselage} \cdot D \cdot (t) \tag{9.6}$$

by determining design variables b, t

Subject to :

$$\begin{aligned} &1) \quad \sigma_{yield} \cdot \frac{4bh^2}{3dD^2} - \left(4.2649\sigma_q + \mu_q\right) > 0 \\ &2) \quad \delta_{allow} \cdot \frac{Et^3}{kd^4} - \left(4.2649\sigma_q + \mu_q\right) > 0 \\ &3) \quad t - 0.003 > 0 \end{aligned} \tag{9.7}$$

The solution of this problem is shown in Table 9-4:

Table 9-4: Optimal solutions

d	0.5 (Case A)	0.462 (Case B)
b	0.00724	0.00705
t	0.00328	0.003
W	0.0162	0.0150

9.1.4 Probability of Failure Evaluation

Once the optimal point is found, the probability of failure at this point for tracks and plates can be evaluated. P_{f1} and P_{f2} are the probability of failure for longitudinal tracks and floor board respectively. System failure happens when any of the 2 components failure happens. System is safe only when both subsystems are safe:

$$
\begin{aligned}
&P_{safety} = P_{safety1} \cdot P_{safety2} \\
&\Rightarrow (1 - P_f) = (1 - P_{f1}) \cdot (1 - P_{f2}) \\
&\Rightarrow P_f = P_{f1} + P_{f2} - P_{f1} \cdot P_{f2}
\end{aligned}
$$

P_{f1} , P_{f2} and total P_f are evaluated as (Table 9-5):

Table 9-5: Probability of failure evaluation for optimal solutions

d	0.5 (Case A)	0.462 (Case B)
Pf1	1.04E-05	1.08E-05
Pf2	1.12E-05	1.16E-06
Pf	2.14E-05	**1.20E-05**

From Table 9-5 it can be seen that Case B is safer than Case A. Besides that, table1 shows that Case B is smaller in weight than case A.

9.2 Case II

In Case II, the rectangular section tracks of Case I would be replaced by more realistic I beam; there will be more than one random variable in the meantime the correlation of the random variables will be considered; floor board will be replaced by that of lighter material and structure. Due to the introduction of multi-random-variables, the analytical expression of reliability constraints will not be available any more, to impose the reliability constraints in an analytical form, a surrogate model based on multivariate interpolation that could provide approximate gradient information over the whole design space would be utilized.

9.2.1 Hypothesis and Simplification

Some key hypothesis and simplifications are listed below:

1) All longitudinal tracks are regarded as simply supported beams (one side hinged the other side roller supported).

2) Loads on longitudinal tracks are evenly distributed.

3) Plate inside any grid area is regarded as an intact plate.

4) The track is designed as I beam as shown in Figure 9-1. t1 and t2 are the thickness of the web and the flange plate respectively, $t_1, t_2 \sim N(0.003, 0.0003)mm$.

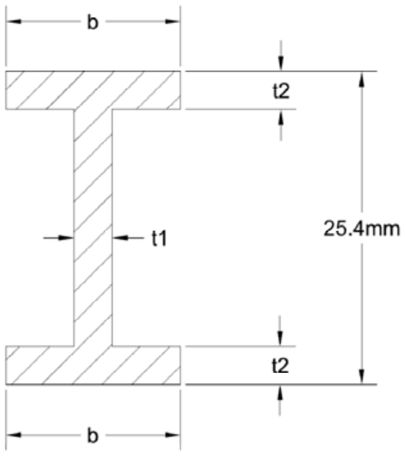

Figure 9-1: Track section view

5) Load on the floor is evenly distributed inside each grid and follows normal distribution: $q \sim N(2200, 400)$ Pa.

6) Floor board is made of honeycomb plate with density ρ=20kg/m^3. Plates are considered fixed on 4 sides. Plate vertical deflection equation:

$$w = k\frac{qd^4}{t^3} \tag{9.8}$$

k: deflection coefficient, k=1e-10;

q: evenly distributed load (N/m^2), $q \sim N(2200,400)$ Pa;

d: smaller width of rectangular plate (m);

t: Plate thickness (m);

7) Track material is selected as Aluminum alloy 6061-T6, the material properties are listed below:

ρ=2.7e3kg/m^3;

σ_{yield}=241 MPa;

E=69 GPa;

8) Since the number of track columns can only be selected as discrete positive integers, the track space d can only be selected as discrete values. However most optimization methods are based on continuous variables, as such track spacing will be taken as continuous design variable in the initial optimization. Discrete effect will be considered in the further steps.

9) Optimization of shape and spacing of transverse beams is not included in this study. Only plate and tracks will be considered.

9.2.2 Optimization Model

Minimize: (total mass of floor board and longitudinal tracks)

Subject to:

1) $P\{g_1(X) < 0\} < 1e-5$	track stress reliability requirement
2) $P\{g_2(X) < 0\} < 1e-5$	floor board deflection reliability requirement
3) $0.3m \le d \le 0.75m$	tracks spacing limit

There are 3 design variables selected for this problem:

d: track spacing;

t: floor board thickness;

b: I beam flange width;

There are three random variables in this problem:

$q \sim N(2200{,}400)$ Pa: load on the floor board;

$t_1 \sim N(0.003{,}0.0003)mm$: thickness of the web of I beam;

$t_2 \sim N(0.003{,}0.0003)mm$: thickness of the flanges of I beam;

For this study the shape and size of the transverse beams are not included. The total mass only includes the mass of all longitudinal tracks and sum of all floor board.

$$\begin{aligned} M &= M_{track} + M_{board} \\ &= A \cdot D \cdot \frac{W_{fuselage}}{d} \cdot \rho_{\text{aluminum}} + W_{fuselage} \cdot D \cdot t \cdot \rho_{\text{honeycomb}} \end{aligned} \tag{9.9}$$

In which:

D: Transverse beam space, 0.75m;

W_{fuselage}: fuselage width, 6m;

ρ: material density;

A: section area of longitudinal track: $A = t_1(0.0254 - 2t_2) + 2bt_2$

The limit-state function for track stress:

$$
\begin{aligned}
g_1(X) &= \sigma_{yield} - \sigma \\
&= \sigma_{yield} - \frac{M \cdot h/2}{I} = \sigma_{yield} - \frac{qd \cdot D^2 \cdot h}{16I} \\
&= \sigma_{yield} - \frac{qd \cdot D^2 \cdot h}{16\left\{\frac{1}{12} t_1 (0.0254 - 2t_2)^3 + \left[\frac{1}{6} bt_2^3 + \frac{1}{2} bt_2 (0.0254 - t_2)^2\right]\right\}}
\end{aligned}
\tag{9.10}
$$

The limit-state function for plate deflection:

$$
\begin{aligned}
g_2(X) &= \delta_{allow} - \delta \\
&= \delta_{allow} - k \frac{qd^4}{t^3}
\end{aligned}
\tag{9.11}
$$

In which

δ_{allow}= 5mm.

Reliability constraint $P_f \{g(X) \le 0\} \le 1e-5$ is equivalent to the constraint on reliability index: $\beta \ge -\Phi^{-1}(1e-5) = 4.2649$. Although β is a function of design variables (b,d,t), however the process of calculating β is iterative, thus it's difficult to find the analytical expression of β with respect to (b,d,t).

The optimization problem can be expressed as:

Minimize:

$$
\begin{aligned}
M &= M_{track} + M_{board} \\
&= \left[t_1(0.0254 - 2t_2) + 2bt_2\right] \cdot D \cdot \frac{W_{fuselage}}{d} \cdot \rho_{\text{aluminum}} + W_{fuselage} \cdot D \cdot t \cdot \rho_{\text{honeycomb}}
\end{aligned}
\tag{9.12}
$$

By selecting the values of (b,d,t),

Subject to:

$$
\begin{aligned}
&1)\beta_1 + \Phi^{-1}(1e-5) \ge 0; \quad \text{(track stress reliability index constraint)} \\
&2)\beta_2 + \Phi^{-1}(1e-5) \ge 0; \quad \text{(floor board deflection reliability index constraint)} \\
&3)d - 0.3 \ge 0; \\
&4)0.75 - d \ge 0;
\end{aligned}
\tag{9.13}
$$

9.2.3 Searching Algorithm

The constraints of the optimization model are all inequality constraints and the reliability index constraints obtained by MCS do not have analytical form thus no gradient information as well. This problem can be solved by Heuristic Gradient Based Searching Algorithm as described in Chapter 7. The reliability constraints that do not have analytical form can replaced by response surface obtained from regression method. Multiquadric or kriging will be used for interpolation.

9.2.3.1 Evaluating β at sampling points

For each dimension of the three design variables, equal interval distributed four points will be used as sampling points. These sampling points on each axis are listed in Table 9-6:

Table 9-6: Sampling points on each dimension

b(m)	0.004	0.006	0.008	0.01
d(m)	0.3	0.45	0.6	0.75
t(m)	0.012	0.015	0.018	0.021

These sampling points on each axis will obtain totally 64 sampling points. The reliability indices are evaluated by correlated HL method for the limit-state functions as listed in Table 9-7 and Table 9-8:

Table 9-7: Reliability indices of stress limit-state function at sampling points

	t=0.012			
	d=0.3	d=0.45	d=0.6	d=0.75
b=0.004	4.6688	1.6748	-0.0403	-1.1103
b=0.006	6.5319	3.5638	1.5505	0.2192
b=0.008	7.6323	5.0589	2.9779	1.474
b=0.01	8.2907	6.1502	4.1895	2.6198
	t=0.015			
	d=0.3	d=0.45	d=0.6	d=0.75
b=0.004	4.6688	1.6748	-0.0403	-1.1103
b=0.006	6.532	3.5638	1.5505	0.2192
b=0.008	7.6355	5.0587	2.9779	1.474
b=0.01	8.2906	6.1502	4.1895	2.6198
	t=0.018			
	d=0.3	d=0.45	d=0.6	d=0.75
b=0.004	4.6688	1.6748	-0.0403	-1.1103
b=0.006	6.532	3.5638	1.5505	0.2192
b=0.008	7.6355	5.0587	2.9779	1.474
b=0.01	8.2906	6.1502	4.1895	2.6198
	t=0.021			
	d=0.3	d=0.45	d=0.6	d=0.75
b=0.004	4.6688	1.6748	-0.0403	-1.1103
b=0.006	6.532	3.5638	1.5505	0.2192
b=0.008	7.6355	5.0587	2.9779	1.474
b=0.01	8.2906	6.1502	4.1895	2.6198

Table 9-8: Reliability indices of deflection limit-state function at sampling points

	t=0.012			
	d=0.3	d=0.45	d=0.6	d=0.75
b=0.004	21.1667	-0.2325	-3.8333	-4.8173
b=0.006	21.1667	-0.2325	-3.8333	-4.8173
b=0.008	21.1667	-0.2325	-3.8333	-4.8173
b=0.01	21.1667	-0.2325	-3.8333	-4.8173
	t=0.015			
	d=0.3	d=0.45	d=0.6	d=0.75
b=0.004	46.5833	4.7881	-2.2448	-4.1667
b=0.006	46.5833	4.7881	-2.2448	-4.1667
b=0.008	46.5833	4.7881	-2.2448	-4.1667
b=0.01	46.5833	4.7881	-2.2448	-4.1667
	t=0.018			
	d=0.3	d=0.45	d=0.6	d=0.75
b=0.004	84.5	12.2778	0.125	-3.196
b=0.006	84.5	12.2778	0.125	-3.196
b=0.008	84.5	12.2778	0.125	-3.196
b=0.01	84.5	12.2778	0.125	-3.196
	t=0.021			
	d=0.3	d=0.45	d=0.6	d=0.75
b=0.004	137.4167	22.7305	3.4323	-1.8413
b=0.006	137.4167	22.7305	3.4323	-1.8413
b=0.008	137.4167	22.7305	3.4323	-1.8413
b=0.01	137.4167	22.7305	3.4323	-1.8413

9.2.3.2 Regression by multiquadric interpolation or Kriging

Multiquadric interpolation can be used to get response surface for the reliability constraints. However it was found that multiquadric interpolation may cause big error under some circumstances. One of the interpolation error examples is illustrated in Figure 9-2:

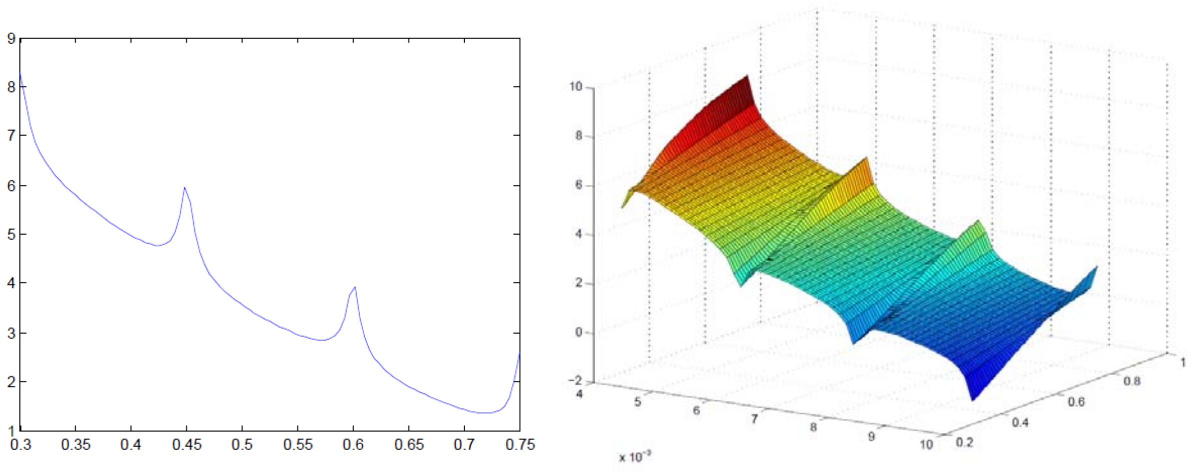

Figure 9-2: Ill conditioned interpolation example by multiquadric interpolation

To avoid big interpolation errors, in this case Kriging interpolation will be used to build the response surface.

Kriging interpolation relies on the distance between sampling points and the distance between interpolation points and sampling points, however the nature of the design space with respect to each axis is very uneven, different dimensions have different scales. Some design variables has very small scales compared to others, if these small scale dimensions has very big impact on the objective function, i.e. with big gradient on these dimensions, original kriging interpolation will underplay their importance of these small scale dimensions by equally evaluate their contribution only by distance. Because of this, it's necessary to normalize each dimensions of the design space first. Suppose the range of x is between $[x_l, x_h]$, the linear normalization of x can be expressed as:

$$\bar{x} = \frac{x - x_l}{x_h - x_l} \tag{9.14}$$

Kriging interpolation will be based on normalized design space on $[0 \quad 1]^3$. Each interpolation point will first be normalized before evaluation by kriging interpolation.

Plot the same curve at (b=0.01; t=0.021) and surface at (t=0.021), the interpolation is accurate and smooth.

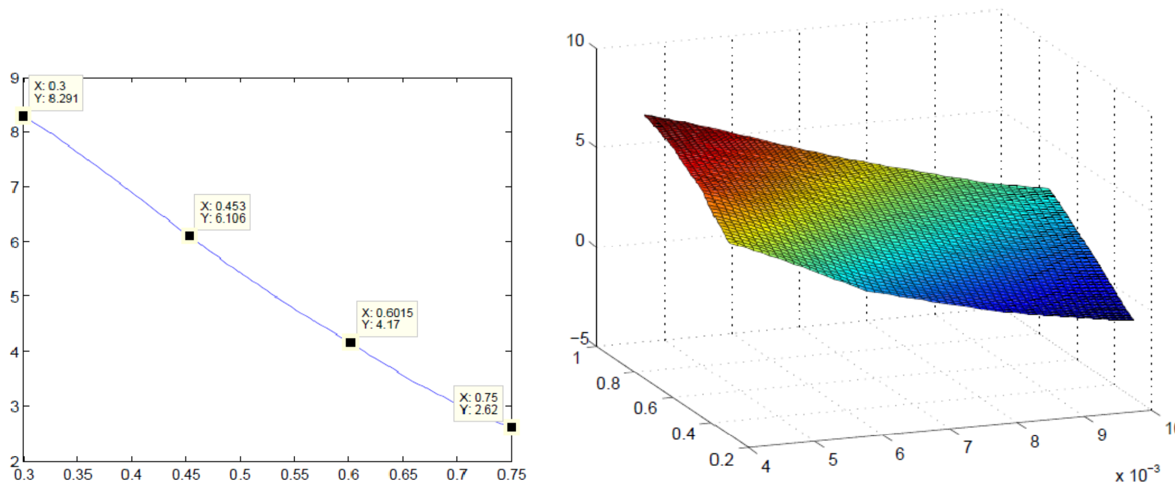

Figure 9-3: Kriging interpolation example on 1 and 2 dimensions

Gradient Projection method requires the value of active constraint gradient at current point, the constraint gradient will be calculated based on Kriging model. The interpolation function of Kriging is:

$$Z(u) = \sum_{i=1}^{n} \lambda_i(u) \cdot Z_i \tag{9.15}$$

Take derivative of Z(u) with respect to x:

$$\begin{aligned} \frac{\partial Z(u)}{\partial \overline{x}} &= \sum_{i=1}^{n} \frac{\partial \lambda_i(u)}{\partial \overline{x}} \cdot Z_i = \mathbf{Z}^T \cdot \frac{\partial \boldsymbol{\lambda}(u)}{\partial \overline{x}} \\ &= \mathbf{Z}^T \cdot \left(\mathbf{K}^{-1} \cdot \frac{\partial \mathbf{k}(u)}{\partial \overline{x}} \right)_{1:n} \end{aligned} \tag{9.16}$$

Since covariance vector **k**(u) is a function of distance between current point and sample points, the covariance function is chosen as:

$$k(d) = 1 - 1.5\left(\frac{d}{d_{\max}}\right) + 0.5\left(\frac{d}{d_{\max}}\right)^3 \tag{9.17}$$

the derivative of covariance vector can be written as:

$$\frac{\partial \mathbf{k}(u)}{\partial \overline{x}} = \frac{\partial}{\partial \overline{x}} \begin{bmatrix} k_1(u) \\ k_2(u) \\ \vdots \\ k_n(u) \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{\partial k_1(u)}{\partial \overline{x}} \\ \frac{\partial k_2(u)}{\partial \overline{x}} \\ \vdots \\ \frac{\partial k_n(u)}{\partial \overline{x}} \\ 0 \end{bmatrix} \tag{9.18}$$

In which

$$\frac{\partial k_i(u)}{\partial \overline{x}} = \frac{\partial k_i(u)}{\partial d_i} \cdot \frac{\partial d_i}{\partial \overline{x}} = \frac{1.5}{d_{\max}} \left[\left(\frac{d_i}{d_{\max}} \right)^2 - 1 \right] \cdot \frac{\overline{x} - \overline{x}_i}{d_i} \tag{9.19}$$

Gradient calculated in this method is based on normalized space, transform is needed to return to the original space:

$$\frac{\partial Z(u)}{\partial x} = \frac{\partial Z(u)}{\partial \overline{x}} \cdot \frac{1}{x_h - x_l} \tag{9.20}$$

Table 9-9 is the searching process of the algorithm

Table 9-9: Searching iterations table

b	d	t
4.00E-02	6.000E-01	1.000E-02
3.95E-02	6.38E-01	2.44E-02
3.04E-02	6.57E-01	2.59E-02
2.81E-02	7.09E-01	3.34E-02
2.57E-02	6.73E-01	2.78E-02
2.23E-02	5.94E-01	1.76E-02
2.10E-02	5.79E-01	1.56E-02
1.81E-02	5.58E-01	1.37E-02
1.50E-02	5.34E-01	1.39E-02
1.38E-02	5.23E-01	1.39E-02
1.38E-02	5.25E-01	1.38E-02

For this problem, the optimal design point is b=1.38cm, d=0.525m, t=13.8mm.

The sensitivity factor can be evaluated at the optimal point by

$$\cos\theta_{u_i} = -\frac{\frac{\partial g(X^*)}{\partial x_i}\sigma_{x_i}}{\sqrt{\sum_{i=1}^{n}\left(\frac{\partial g(X^*)}{\partial x_i}\sigma_{x_i}\right)^2}} \tag{9.21}$$

In which

$$g(X) = \sigma_{yield} - \frac{qd\cdot D^2\cdot h}{16\left\{\frac{1}{12}t_1(0.0254-2t_2)^3 + \left[\frac{1}{6}bt_2^3 + \frac{1}{2}bt_2(0.0254-t_2)^2\right]\right\}} \tag{9.22}$$

$g_2(X)$ will not be considered since it has only one random variable. MPP point can be computed by HL method as X*=(2956pa, 0.002835m, 0.002612m). At the optimal design point, the sensitivity factor can be computed as: $\cos\theta_1 = 0.5100$ $\cos\theta_2 = 0.2164$ and $\cos\theta_3 = 0.8325$ .These sensitivity factors correspond to q, t1 and t2 respectively. This result shows that value of t2 has greatest importance for the reliability of the structure.

To evaluate the accuracy of the response surface model based on Kriging model, a quantity σ similar to mean square error is defined as:

$$\sigma = \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}e_i^2}}{\frac{1}{n}\sum_{i=1}^{n}\beta_i} \tag{9.23}$$

In which:

$e_i = \beta_i - \beta_i'$ is estimation error;

β_i, β_i' are respectively the accurate reliability index value and the approximation value at a sample point;

n is test points number. Test points are selected randomly.

σ reflects the estimation error of the response surface, it can be used to evaluate the goodness of fit of the response surface model. In this example, 100 random points are selected. The σ value is 3%. This shows that the accuracy of the response surface model is acceptable.

9.3 Summary

In this chapter, the RBDO based method and algorithms introduced in the previous chapters are applied in the aircraft floor grid layout optimization problem. First the optimization model including the objective function and the reliability constraints is set up. The optimal point is found by heuristic searching algorithm. In each step of heuristic searching algorithm, the values and the gradients of the reliability constraints are evaluated based on the response surface model which is obtained by kriging. The values of the sample points for kriging are computed by various methods including HL, SORM, NLP, or MCS. The goodness of fit of the response surface is measured by σ, and the three random variables are analyzed for their sensitivities to the reliability index.

CHAPTER 10: ILLUSTRATED EXAMPLE OF FLOOR GRID AND FLOOR BOARD BILEVEL OPTIMIZATION

When there are two different player in an optimization process, both of these players have their own objectives which are usually not the same or even conflictive. For this kind of problems bilevel optimization method can be applied. The two players are identified respectively as leader and follower according to their importance and priority. The leader will first decide an optimal solution then the follower will make its decision to maximize its own utility based on the leader's decision. Follower's decision would be feedback to the leader, the leader would adjust its decision accordingly to maximize its own utility. This process continues until a balance between leader and follower is reached. Details about bilevel optimization can be found in [Bard, J.F., 2010], [Dempe, S. 2010]. Some good examples can be found in [Rao, J.R.J, et al, 1997].

Generally bilevel optimization problem can be expressed as:

$$\begin{aligned}
&\min_{x} f_L(x,y) \\
&\text{s.t. } G(x) \le 0, \\
&\min_{y} f_F(x,y) \\
&\text{s.t. } H(x,y) \le 0,
\end{aligned} \tag{10.1}$$

In which f_L is the upper level objective function, f_F is the lower level objective function.

$x = (x_1, x_2, \cdots, x_n) \in R^n$ is the decision variables of the upper level, here $a_i \le x_i \le b_i$, $a_i, b_i \in R$.

$y = (y_1, y_2, \cdots, y_n) \in R^m$ is the decision variables of the lower level. $G(x) = (g_1(x), g_2(x), \cdots, g_s(x))$ is the upper level constraints vector, in which the decision variable boundaries $a_i \le x_i \le b_i$ are included.

$H(x,y) = (h_1(x,y), h_2(x,y), \cdots h_t(x,y))$ is the lower level constraints vector.

Let $S = \{(x, y) \mid G(x) \le 0, H(x, y) \le 0\}$. For any x that satisfies $G(x) \le 0$, the solution $Y(x)$ of the lower level optimization problem

$$\begin{aligned} &\min_{y} f_F(x, y) \\ &\text{s.t. } H(x, y) \le 0, \end{aligned} \tag{10.2}$$

Is the optimal solution set. $F(s) = \{(x, y) \mid (x, y) \in S, y \in Y(x)\}$ is the feasible region of the bilevel optimization problem; if $(x, y) \in F(S)$, (x, y) is a feasible solution of the bilevel optimization problem. If for any feasible solution (x, y), $f_L(x^*, y^*) \le f_L(x, y)$ always exist, then (x^*, y^*) is an optimal solution of the bilevel optimization problem.

The objective of the optimization problem is to minimize the weight of the sum of tracks and board. This can also be modeled as a bilevel optimization problem. Set the leader level of optimization as the total weight, the follower level as the weight of the board. The corresponding objective functions are minimizing the respective weights W_{total} and W_{board}. Set W_{total} as the Leader:

$$\begin{aligned} &\text{minimize} \\ &f^{l}(b,d,t) = \left[t_1(H - 2t_2) + 2bt_2\right] \cdot D \cdot \frac{W_{fuselage}}{d} \cdot \rho_{\text{aluminum}} + W_{fuselage} \cdot D \cdot t \cdot \rho_{\text{honeycomb}} \\ &\text{subject to:} \\ &\qquad h_1^{l} : (d,t) = (d^*(b), t^*(b)); \\ &\qquad b > 0.003; \end{aligned} \tag{10.3}$$

Where $(d^*(b), t^*(b))$ is the solution to the follower's problem given by:

$$\begin{aligned} &\text{minimize} \quad f^{f}(t) = W_{fuselage} \cdot D \cdot t \cdot \rho_{\text{honeycomb}} \\ &\text{subject to:} \\ &\qquad \beta_1(b,d) + \Phi^{-1}(1e-5) \ge 0; \\ &\qquad \beta_2(d,t) + \Phi^{-1}(1e-5) \ge 0; \\ &\qquad 0.3 < d < 0.75 \end{aligned} \tag{10.4}$$

For the follower problem, since the objective function $f^f(t)$ is monotone with respect to t, the constraint $\beta_2(d,t)+\Phi^{-1}(1e-5)\geq 0$ must be active to avoid t from decreasing to zero. From this constraint $t(d)$ can be obtained. From constraint $\beta_1(b,d)+\Phi^{-1}(1e-5)\geq 0$, $d(b)$ can be calculated, thus $d(b)$ and $t(b)$ are all known. Bring them back to the leader problem it becomes a one-degree-of-freedom problem, b can be solved.

Since β_1 and β_2 are computed by iterative numerical method, constraints specified by β_1 and β_2 does not have an analytical form. The response surface models derived from by Kriging regression method can be used as substitutes for these constraints. t1 and t2 are random variables, the mean and variance of t1 and t2 will determine the value of β1 and β2. However, in the leader's objective function they are set as their respective mean values. The rest of the steps are the same with single level optimization problem.

The flow chat of this process is shown in Figure 10-1:

Set the starting point at (0.05, 0.9, 0.05)m, the searching steps in each iteration are listed in Table 10-1.

Table 10-1: Searching iterations table

b(m)	d(m)	t(m)
5.00E-02	9.00E-01	5.00E-02
4.09E-02	7.15E-01	4.62E-02
3.34E-02	6.27E-01	4.05E-02
3.08E-02	6.03E-01	2.93E-02
2.58E-02	5.89E-01	2.32E-02
2.21E-02	5.67E-01	1.93E-02
1.94E-02	5.31E-01	1.72E-02
1.74E-02	5.33E-01	1.51E-02
1.61E-02	5.10E-01	1.37E-02
1.52E-02	4.99E-01	1.37E-02
1.52E-02	5.01E-01	1.37E-02

For this problem, the optimal design point is b=1.52cm, d=0.501m, t=13.7mm.

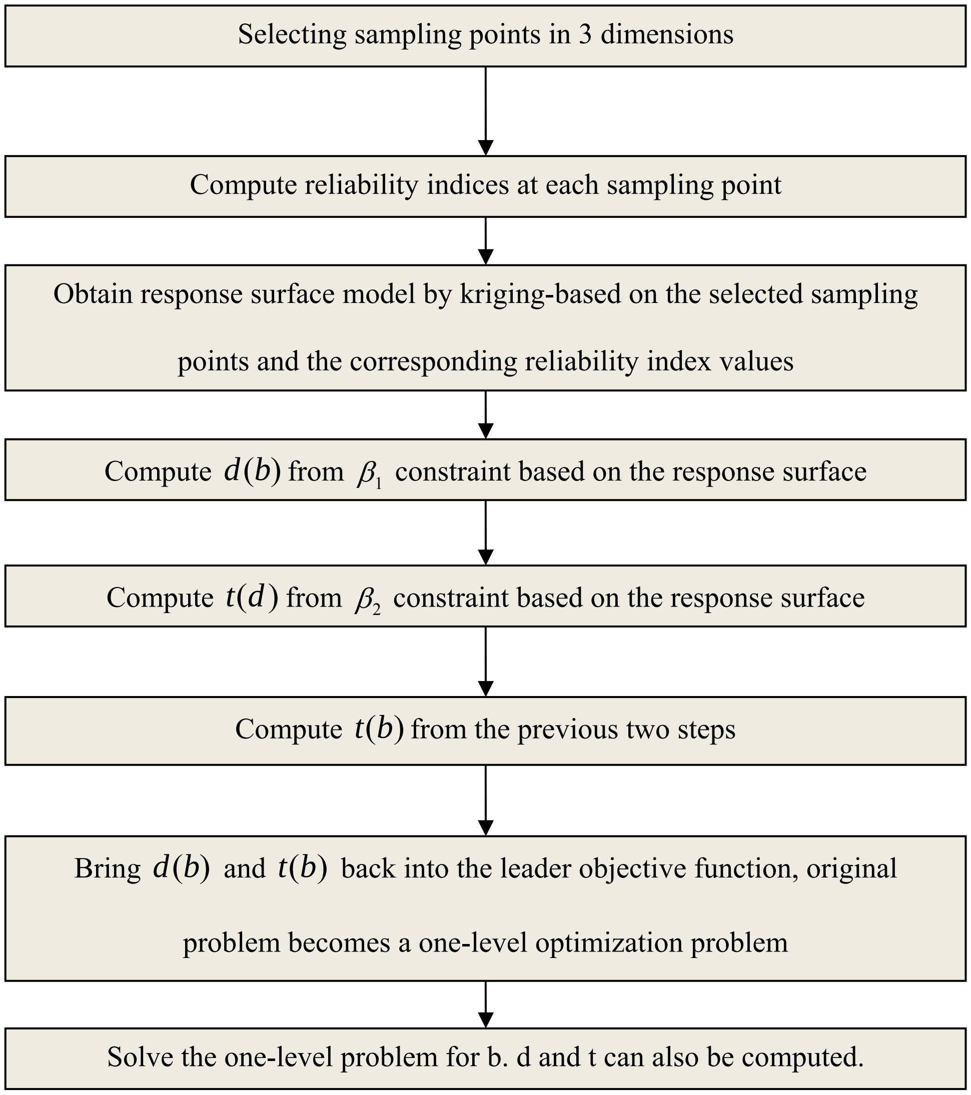
Selecting sampling points in 3 dimensions
Compute reliability indices at each sampling point
Obtain response surface model by kriging-based on the selected sampling points and the corresponding reliability index values
Compute $d(b)$ from β_1 constraint based on the response surface
Compute $t(d)$ from β_2 constraint based on the response surface
Compute $t(b)$ from the previous two steps
Bring $d(b)$ and $t(b)$ back into the leader objective function, original problem becomes a one-level optimization problem
Solve the one-level problem for b. d and t can also be computed.

Figure 10-1: Flow chat of bilevel optimization searching algorithm

The sensitivity factor can be evaluated at the optimal point by

$$\cos\theta_{u_i} = -\frac{\dfrac{\partial g(X^*)}{\partial x_i}\sigma_{x_i}}{\sqrt{\sum_{i=1}^{n}\left(\dfrac{\partial g(X^*)}{\partial x_i}\sigma_{x_i}\right)^2}}$$

In which

$$g(X) = \sigma_{yield} - \frac{qd \cdot D^2 \cdot h}{16\left\{\frac{1}{12} t_1 (0.0254 - 2t_2)^3 + \left[\frac{1}{6} bt_2^3 + \frac{1}{2} bt_2 (0.0254 - t_2)^2\right]\right\}}$$

$g_2(X)$ will not be considered since it has only one random variable. MPP point can be computed by HL method as X*=(2873pa, 0.002873m, 0.002679m). At the optimal design point, the sensitivity factor can be computed as: $\cos\theta_1 = 0.4743$ $\cos\theta_2 = 0.2146$ and $\cos\theta_3 = 0.8538$ .These sensitivity factors correspond to q, t1 and t2 respectively. This result shows that value of t2 has greater importance for the reliability of the structure.

CHAPTER 11: CONCLUSION, LIMITATIONS AND FUTURE WORK

11.1 Conclusion

In this thesis, several RBDO method and algorithms for airplane floor grid layout optimization have been proposed. Copula has been introduced to model random variable correlations as well as the method to apply copula in Monte Carlo simulations. The influence of random variable correlations to the probability of failure has been analyzed. Based on HL method, a correlated HL method is proposed to evaluate reliability index under correlation. As an intuitive and alternative method for computing reliability index, reliability index is interpreted as an optimization problem and a nonlinear programming algorithm is introduced to evaluate reliability index. To evaluate the reliability index by Monte Carlo simulation in a time efficient manner, a kriging-based surrogate model is proposed, comparison has been made between the original model and surrogate model in terms of simulation time. Kriging has been compared with other interpolative methods by quoting literatures. Since in RBDO optimization models, the reliability constraints obtained by MCS usually do not have analytical model, which is necessary for most optimization algorithms, a heuristic gradient based direct searching algorithm and a regression based response surface method is proposed. A framework of the process of building an optimization model based on RBDO is proposed. These methods, algorithms and framework have been applied to ten bar truss example and the layout optimization of aircraft floor grid structural design.

By studying several examples, this thesis shows that for highly nonlinear limit-state functions, MVFOSM based on first order Tylor series expansion at mean value point has greater error than HL and SORM methods. This thesis also shows that Kriging-based surrogate model can effectively increase the simulation efficiency by reducing computation expenses. By examine several families of copula, this thesis shows that copula can be applied in simulation based method to discover random variable correlations and to produce sample points for MCS. It also shows that different families of copula with different dependence patterns may result in different simulation results. By applying heuristic searching algorithm to two examples, this thesis shows that heuristic gradient based searching algorithm can be applied to solve optimization problems with inequality constraints that are segment functions.

The contributions of this thesis mainly include:

1) Proposing a general process of reliability-based structural design optimization;

2) Using copula in RBDO to discover correlations between random variables and for generating data samples for MCS;

3) Proposing a surrogate model building method and response surface building method based on Kriging

4) Proposing a heuristic gradient-based search algorithm for optimization problems with segment-function inequality constraints;

5) Proposing two alternative methods to evaluate reliability index by NLP;

6) Proposing a correlated HL method.

7) General process of reliability-based structural design and optimization is applied to aircraft floor grid layout optimization problem.

11.2 Limitations

Copula can be used to discover correlation and for simulation, however, as have been shown in the ten-bar-truss example, with a same measure of concordance, say, Kendall's tau, data samples from different families of copula produce different simulation results. This property is undesirable in engineering. Kriging can be used to effectively reduce the simulation time. It can be used to build a computationally inexpensive surrogate model for many simulation problems. However, if the computation point is far from the sample points, the interpolation algorithm will show big error. This requires that the computation point be within the coverage of sample point set. For highly nonlinear problems, to achieve satisfactory results, it's necessary to increase the number of sampling points, which increases the computation expenses. To overcome this problem, the acting radius can be set smaller so that less number of sampling points are included in computation. However this could compromise the estimation accuracy.

Heuristic searching algorithm introduced in this thesis is based on gradient projection method, it checks the constraints violation in each step. Heuristic searching algorithm can be used to find the optimal point for RBDO problems. However, as an iterative numerical searching algorithm that include nested iterative gradient projection algorithm, heuristic searching algorithm is usually time-consuming, especially when there are nested searching steps. Another problem of heuristic searching algorithm is that it is easy to converge to local minimum. To avoid converging to local minimum, large number of starting points need be applied, which again increases computation expenses.

11.3 Future Work

Based on the limitations and drawbacks of the methods and algorithms proposed in this thesis, future works should be focused on:

1) Since different families of copula with an identical measure of concordance would usually obtain different simulation results, it's necessary to study how to choose a family of copula that resembles the true data sample relations best.

2) Due to the drawbacks of Kriging-based surrogate model, other methods such as Artificial Neural Network (ANN) should also be applied to build surrogate model for computationally expensive models.

3) Finite Element Model should be used to replace the current inaccurate analytical model in the limit state functions.

4) Due to the low probability of failure in simulation, Importance Sampling should also be used to eliminate unnecessary computations.

5) Correlated HL method proposed in this thesis need be proved or disproved.

APPENDIX

```
%% CODE COPULAFUNC01: for copula to generate a random variable.
% input u0 and w are RV's within [0,1], t is Gumbel-Hougaard coefficient
function y = copulafunc01(u0, w, t)

options = optimset('Display', 'off'); % Turn off Display
y = fzero(@nestfunc01, 0.5, options);

 function y = nestfunc01(x) % Compute the polynomial.
 %y = x^3 + b*x + c;
 y = exp(-((-log(u0))^(t)+(-log(x))^(t))^(1/t))*(((-log(u0))^t+(-
log(x))^t)^((1-t)/t))*((-log(u0))^(t-1)/u0 ) - w ;
 end
end
```

```
%% CODE COPULAFUNC02:for copula to generate a random variable.
% input u0 and w are RV's within [0,1], t is Clayton coefficient
function y = copulafunc02(u0, w, t)

options = optimset('Display', 'off'); % Turn off Display
y = fzero(@nestfunc01, 0.5, options);

 function y = nestfunc01(x) % Compute the polynomial.
 %y = x^3 + b*x + c;
 y = ((u0)^(-t)+(x)^(-t)-1)^((-1-t)/t)*(u0)^(-t-1) - w;
 end
end
```

```
%% CODE KRIGING01: Kriging model for rod 3
function Y = kriging01(X)
% X is a matrix of (3*n), the input points
% L is the coefficients lambda (126*n)
load kriging_rod3;
[m,n]=size(X);
[m1,n1]=size(points);
L=zeros(n1+1,n);
% covariance between the given point x and the set points
k=zeros(n1+1,n);
for i=1:n
    for j=1:n1
        d=((X(1,i)-points(1,j))^2+(X(2,i)-points(2,j))^2+(X(3,i)-
points(3,j))^2)^0.5;
        if d>=dmax
            d=dmax;
        end
        k(j,i)=1-1.5*(d/dmax)+0.5*(d/dmax)^3;
    end
    k(n1+1,i)=1;
end
L=C1inv*k;
Y=G'*L;
```

```
%% CODE KRIGING02: Kriging model for rod 3 and 7
function Y = kriging02(X)
% X is a matrix of (3*n), the input points
% L is the coefficients lambda (126*n)
load kriging_rod7;
[m,n]=size(X);
[m1,n1]=size(points);
L=zeros(n1+1,n);
% covariance between the given point x and the set points
k=zeros(n1+1,n);
for i=1:n
    for j=1:n1
        d=((X(1,i)-points(1,j))^2+(X(2,i)-points(2,j))^2+(X(3,i)-
points(3,j))^2)^0.5;
        if d>=dmax
            d=dmax;
        end
        k(j,i)=1-1.5*(d/dmax)+0.5*(d/dmax)^3;
    end
    k(n1+1,i)=1;
end
L=C1inv*k;
Y=G'*L;
```

```
%% CODE SOLVE_U1: FOR ROD3, GIVEN U2 AND U3, COMPUTE U1
function y=solve_u1(u2,u3)
% this function is for rod 3
options = optimset('Display', 'off'); % Turn off Display
y = fzero(@nestfunc01, 0.0, options);

 function y = nestfunc01(u1) % Compute the polynomial.
 %y = x^3 + b*x + c;
 y = 100000000 - (3000*((((1770887431076117*u2)/295147905179352825856 +
3/100000)*(u1/50000 + 1/10000)*(u3/50000 + 1/10000)*(u3/50000 +
2*2^(1/2)*(u1/50000 + 1/10000) +
1/10000))/(4*((1770887431076117*u2)/295147905179352825856 +
3/100000)^2*(8*(u1/50000 + 1/10000)^2 + (u3/50000 + 1/10000)^2) + (u1/50000 +
1/10000)*(u3/50000 + 1/10000)^2*(u1/50000 +
(5312662293228351*u2)/147573952589676412928 + 7/25000) +
4*2^(1/2)*((1770887431076117*u2)/295147905179352825856 + 3/100000)*(u1/50000
+ 1/10000)*(u3/50000 + 1/10000)*((3*u1)/50000 +
(1770887431076117*u2)/73786976294838206464 + 21/50000)) + 2))/(u1/50000 +
1/10000);
 end
end
```

```
%% CODE SOLVE_U3: FOR ROD7, GIVEN U1 AND U2, COMPUTE U3
function y=solve_u3(u1,u2)
% this function is for rod 7
options = optimset('Display', 'off'); % Turn off Display
y = fzero(@nestfunc01, 0.0, options);

 function y = nestfunc01(u3) % Compute the polynomial.
 %y = x^3 + b*x + c;
 y = 100000000 - (3000*((2^(1/2)*((1770887431076117*u2)/295147905179352825856
+ 3/100000)*(u1/50000 + 1/10000)*(u3/50000 + 1/10000)*(u3/50000 +
2*2^(1/2)*(u1/50000 + 1/10000) +
1/10000))/(4*((1770887431076117*u2)/295147905179352825856 +
3/100000)^2*(8*(u1/50000 + 1/10000)^2 + (u3/50000 + 1/10000)^2) + (u1/50000 +
1/10000)*(u3/50000 + 1/10000)^2*(u1/50000 +
(5312662293228351*u2)/147573952589676412928 + 7/25000) +
4*2^(1/2)*((1770887431076117*u2)/295147905179352825856 + 3/100000)*(u1/50000
+ 1/10000)*(u3/50000 + 1/10000)*((3*u1)/50000 +
(1770887431076117*u2)/73786976294838206464 + 21/50000)) + 2))/(u3/50000 +
1/10000);
 end
end
```

```
%% CODE STRESS: COMPUTE ROD STRESSES GIVEN SECTION AREA AND EXT LOADS
function y = stress(A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,P)

A=zeros(10,10);
A(1,8)=-2^(-0.5);
A(2,10)=-2^(-0.5);
A(3,8)=-2^(-0.5);
A(4,10)=-2^(-0.5);
A(5,8)=-2^(-0.5);
A(5,10)=-2^(-0.5);
A(6,10)=-2^(-0.5);
A(7,8)=1;
A(9,10)=1;
```

```
a11 = 1/A1+1/A3+1/A5+2^1.5/A7+2^1.5/A8;
a12 = 1/A5;
a21 = a12;
a22 = 1/A2+1/A4+1/A5+1/A6+2^1.5/A9+2^1.5/A10;

temp_den = a11*a22 - a12*a21;
temp_1 = (-a22*2^0.5*(1/A3+2^1.5/A7))/temp_den;
temp_2 = (a22*2^0.5*(1/A1-2/A3-1/A5-
2^1.5/A7)+a12*(2^0.5/A4+2^0.5/A5+4/A9))/temp_den;
temp_3 = (a21*2^0.5*(1/A3+2^1.5/A7))/temp_den;
temp_4 = -(a11*(2^0.5/A4+2^0.5/A5+4/A9)+a21*2^0.5*(1/A1-2/A3-1/A5-
2^1.5/A7))/temp_den;

% coefficient matrix for external force P
B=[ 0 1
    0 0
    -1 -2
    0 -1
    0 -1
    0 0
    2^0.5 2^0.5
    temp_1 temp_2
    0 2^0.5
    temp_3 temp_4];

% internal force N calculated:
N = (eye(10)-A)^(-1)*B*P;

% stresses
sigma=[N(1)/A1;N(2)/A2;N(3)/A3;N(4)/A4;N(5)/A5;
    N(6)/A6;N(7)/A7;N(8)/A8;N(9)/A9;N(10)/A10];
y=sigma;
```

```
% Compute next point by Gradient Projection Method, with 3 variables only
% This function computes one step only
function y = gradproj(f,h,SP,step)
% y: next point, column vector;
% f: objective function of x1,x2,x3;
% h: equality constraints, column vector;
% SP: starting point, column vector;
% step: step size;
syms ph % partial derivatives of h
syms x1 x2 x3 % design variables
[n,m]=size(h); % constraint number n
ph=[diff(h,x1) diff(h,x2) diff(h,x3)];
pf=[diff(f,x1) diff(f,x2) diff(f,x3)];
x1=SP(1);x2=SP(2);x3=SP(3);
ph1=eval(ph);
pf1=eval(pf);
```

```
% projection matrix P:
P=eye(3)-ph1'*(ph1*ph1')^(-1)*ph1;
% EP: end point
EP=SP-step*P*pf1';
for i=1:3
    SP=EP;
    x1=SP(1);x2=SP(2);x3=SP(3);
    ph1=eval(ph);
    h1=eval(h);
    EP=SP-ph1'*(ph1*ph1')^(-1)*h1;
end
y=EP;
```

```
%% code 01 File for ten-bar truss model
% -------------------------------------
%First assign value for
%A1...A10: bar section area;
%P:         external force;
A_hor=1e-4;
A_ver=0.3e-4;
A_dia=0.6e-4;
A1=A_hor; A2=A_hor; A3=A_hor; A4=A_hor;
A5=A_ver; A6=A_ver;
A7=A_dia; A8=A_dia; A9=A_dia; A10=A_dia;
P = [3000; 3000];

%A matrix: coefficient matrix for N
A=zeros(10,10);
A(1,8)=-2^(-0.5);
A(2,10)=-2^(-0.5);
A(3,8)=-2^(-0.5);
A(4,10)=-2^(-0.5);
A(5,8)=-2^(-0.5);
A(5,10)=-2^(-0.5);
A(6,10)=-2^(-0.5);
A(7,8)=1;
A(9,10)=1;

a11 = 1/A1+1/A3+1/A5+2^1.5/A7+2^1.5/A8;
a12 = 1/A5;
a21 = a12;
a22 = 1/A2+1/A4+1/A5+1/A6+2^1.5/A9+2^1.5/A10;

temp_den = a11*a22 - a12*a21;
temp_1 = (-a22*2^0.5*(1/A3+2^1.5/A7))/temp_den;
temp_2 = (a22*2^0.5*(1/A1-2/A3-1/A5-
2^1.5/A7)+a12*(2^0.5/A4+2^0.5/A5+4/A9))/temp_den;
temp_3 = (a21*2^0.5*(1/A3+2^1.5/A7))/temp_den;
```

```
temp_4 = -(a11*(2^0.5/A4+2^0.5/A5+4/A9)+a21*2^0.5*(1/A1-2/A3-1/A5-
2^1.5/A7))/temp_den;

% coefficient matrix for external force P
B=[ 0 1
    0 0
    -1 -2
    0 -1
    0 -1
    0 0
    2^0.5 2^0.5
    temp_1 temp_2
    0 2^0.5
    temp_3 temp_4];

% internal force N calculated:
N = (eye(10)-A)^(-1)*B*P;

% stresses
sigma=[N(1)/A1;N(2)/A2;N(3)/A3;N(4)/A4;N(5)/A5;
    N(6)/A6;N(7)/A7;N(8)/A8;N(9)/A9;N(10)/A10]




%% code 02 prepare input data

A_hor=1e-4;
A_ver=0.3e-4;
A_dia=0.6e-4;
A1=A_hor; A2=A_hor; A3=A_hor; A4=A_hor;
A5=A_ver; A6=A_ver;
A7=A_dia; A8=A_dia; A9=A_dia; A10=A_dia;
P = [3000; 3000];
stress(A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,P)


%% code 03 Experiment1: MCS with P uncorrelated, both P1 and P2 ~N(3000, 500)

n = 1e6;
rho = 0;
Z = mvnrnd([3000 3000], 500^2*[1 rho; rho 1], n);
%plot(Z(:,1),Z(:,2),'.'); xlim([0 5000]); ylim([0 5000]);

% rods section area
A_hor=1e-4;
A_ver=0.3e-4;
A_dia=0.6e-4;
A1=A_hor; A2=A_hor; A3=A_hor; A4=A_hor;
A5=A_ver; A6=A_ver;
A7=A_dia; A8=A_dia; A9=A_dia; A10=A_dia;

% count of failure
i_failurecount=0;
% the maximum permissible stress is 100MPa
```

```
s_safe=100e6;

% Evaluating failure rate by Monte Carlo Simulation
for i=1:n
    P=Z(i,:).';
    s_max=max(stress(A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,P));
    if (s_max>=s_safe)
        i_failurecount=i_failurecount+1;
    end
end
failurerate=i_failurecount/n
% result: 0.003869


%% code 04 Experiment2: MCS with P correlated using bivariate Gaussian
distribution.

n = 1e6;
rho = 0.6;
Z = mvnrnd([3000 3000], 500^2*[1 rho; rho 1], n);
%plot(Z(:,1),Z(:,2),'.'); xlim([0 5000]); ylim([0 5000]);

% rods section area
A_hor=1e-4;
A_ver=0.3e-4;
A_dia=0.6e-4;
A1=A_hor; A2=A_hor; A3=A_hor; A4=A_hor;
A5=A_ver; A6=A_ver;
A7=A_dia; A8=A_dia; A9=A_dia; A10=A_dia;

% count of failure
i_failurecount=0;
% the maximum permissible stress is 100MPa
s_safe=100e6;

% Evaluating failure rate by Monte Carlo Simulation
for i=1:n
    P=Z(i,:).';
    s_max=max(stress(A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,P));
    if (s_max>=s_safe)
        i_failurecount=i_failurecount+1;
    end
end
failurerate=i_failurecount/n
% result: 0.017161



%% code 05 KL expansion example .

rho=0.6;
K=500^2*[1 rho;rho 1];
[V,D] = eig(K);
A=V*(D.^0.5);
X=mvnrnd([0 0], [1 0; 0 1], 1000);
```

```
X=X';
Y=A*X;
Y=3000+Y;
% Y are correlated RV's generated by KL expansion

figure;
subplot(1,2,1);
plot(X(1,:),X(2,:),'.'); axis equal;
subplot(1,2,2);
plot(Y(1,:),Y(2,:),'.'); axis equal;xlim([0 5000]);ylim([0 5000]);



%% code 06 Experiment3: MCS with P correlated using KL expansion.

n = 1e5;
rho=0.6;
K=500^2*[1 rho;rho 1];
[V,D] = eig(K);
A=V*(D.^0.5);
X=mvnrnd([0 0], [1 0; 0 1], n);
X=X';
Y=A*X;
Y=3000+Y;
Z=Y';

% rods section area
A_hor=1e-4;
A_ver=0.3e-4;
A_dia=0.6e-4;
A1=A_hor; A2=A_hor; A3=A_hor; A4=A_hor;
A5=A_ver; A6=A_ver;
A7=A_dia; A8=A_dia; A9=A_dia; A10=A_dia;

% count of failure
i_failurecount=0;
% the maximum permissible stress is 100MPa
s_safe=100e6;

% Evaluating failure rate by Monte Carlo Simulation
for i=1:n
    P=Z(i,:).';
    s_max=max(stress(A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,P));
    if (s_max>=s_safe)
        i_failurecount=i_failurecount+1;
    end
end
failurerate=i_failurecount/n
% result: 0.01746




%% code 07 Copula example .
```

```
n = 1e3;
rho=0.6;
tau = copulastat('Gaussian',rho ,'type','kendall');
alpha = copulaparam('Frank',tau,'type','kendall');
U = copularnd('Frank',alpha,n);
Z = norminv(U,3000,500);
%plot(Z(:,1),Z(:,2),'.'); axis equal;xlim([0 5000]);ylim([0 5000]);
hist(Z,20);

subplot(3,1,1);
plot(U(:,1),U(:,2),'.');
title(['Clayton Copula, {\it\alpha} = ',sprintf('%0.2f',alpha)]);
xlabel('U1');
ylabel('U2');




%% code 08 Experiment4: MCS with P correlated using
(Clayton,Gumbel,Frank)copula
n = 1e5;
rho=-0.6;
tau = copulastat('Gaussian',rho ,'type','kendall');
alpha = copulaparam('Frank',tau,'type','kendall');
U = copularnd('Frank',alpha,n);
Z = norminv(U,3000,500);
%Z = Z*500 + 3000;
%plot(Z(:,1),Z(:,2),'.'); axis equal;xlim([0 5000]);ylim([0 5000]);

% rods section area
A_hor=1e-4;
A_ver=0.3e-4;
A_dia=0.6e-4;
A1=A_hor; A2=A_hor; A3=A_hor; A4=A_hor;
A5=A_ver; A6=A_ver;
A7=A_dia; A8=A_dia; A9=A_dia; A10=A_dia;

% count of failure
i_failurecount=0;
% the maximum permissible stress is 100MPa
s_safe=100e6;

% Evaluating failure rate by Monte Carlo Simulation
for i=1:n
    P=Z(i,:).';
    s_max=max(stress(A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,P));
    if (s_max>=s_safe)
        i_failurecount=i_failurecount+1;
    end
end
failurerate=i_failurecount/n
% result: for rho=0.6
% Clayton: 0.00756
% Gumbel:  0.02166
% Frank:   0.01154
```

```
%% code 09 inverse normal CDF code
P=[0.2 0.3 0.6]
norminv(P,0,1)
```

```
%% code 10 generating random variables by copula
% copulafunc01 is Gumbel-Hougaard copula
% copulafunc02 is Clayton copula
clear;
n=1e3;
U=rand([n 2]);
for i=1:n
    A(i)=copulafunc01(U(i,1),U(i,2),5);
    B(i)=copulafunc02(U(i,1),U(i,2),5);
end
subplot(1,2,1);
plot(U(:,1),A(:),'.'); axis equal;xlim([0 1]);ylim([0 1]);
subplot(1,2,2);
plot(U(:,1),B(:),'.'); axis equal;xlim([0 1]);ylim([0 1]);
```

```
%% code 11 Illustrating how positive and negative Kendall's tau influences
failure
% rate
n = 5e2;
rho=0.8;
tau1 = copulastat('Gaussian',rho ,'type','kendall');
tau2=-tau1;
alpha1 = copulaparam('Frank',tau1,'type','kendall');
alpha2 = copulaparam('Frank',tau2,'type','kendall');
U1 = copularnd('Frank',alpha1,n);
U2 = copularnd('Frank',alpha2,n);
Z1 = norminv(U1,1500,500);
Z2 = norminv(U2,1500,500);
subplot(1,2,1);
plot(Z1(:,1),Z1(:,2),'.'); axis equal;xlim([0 5000]);ylim([0 5000]);
subplot(1,2,2);
plot(Z2(:,1),Z2(:,2),'.'); axis equal;xlim([0 5000]);ylim([0 5000]);
```

```
%% ********************************************************
%  Here below starts FOSM and SORM
% ********************************************************

%% code 12 Compute initial beta at mean value of X; the Pf of each beta
% external load; mean rod area; coeff. of variance;
clear;
```

```
P=3000;
s_allow=100e6;
mu1=1e-4;mu2=0.3e-4;mu3=1e-4;
cv=0.2;
% standard variance of rod section area
var1=mu1*cv;var2=mu2*cv;var3=mu3*cv;
% xi:rod section area; pi:differential of limit-state func gi;
syms x1 x2 x3 p1 p2 p3;
DT=4*x2^2*(8*x1^2+x3^2)+4*2^0.5*x1*x2*x3*(3*x1+4*x2)+x1*x3^2*(x1+6*x2);
% rod 3
g1=s_allow-P/x1*(2+(x1*x2*x3*(2*2^0.5*x1+x3))/DT);
% rod 7
g2=s_allow-P/x3*(2+2^0.5*x1*x2*x3*((2*2^0.5*x1+x3))/DT);

% partial diff. of g wrt three variables x1 x2 x3
p1=diff(g1,x1);
p2=diff(g1,x2);
p3=diff(g1,x3);
den=((p1*var1)^2+(p2*var2)^2+(p3*var3)^2)^0.5;
x1=mu1;
x2=mu2;
x3=mu3;
beta(1)=eval(g1/den);

syms x1 x2 x3;
p1=diff(g2,x1);
p2=diff(g2,x2);
p3=diff(g2,x3);
den=((p1*var1)^2+(p2*var2)^2+(p3*var3)^2)^0.5;
x1=mu1;
x2=mu2;
x3=mu3;
beta(2)=eval(g2/den);

Pf=normcdf(-beta)



%% code 13 MCS for uncorrelated Ai, rod 3 and rod 7
clear;
P=3000;
s_allow=100e6;
mu1=1e-4;mu2=0.3e-4;mu3=1e-4;
cv=0.2;

% xi:rod section area; pi:differential of limit-state func gi;
syms x1 x2 x3 p1 p2 p3;
DT=4*x2^2*(8*x1^2+x3^2)+4*2^0.5*x1*x2*x3*(3*x1+4*x2)+x1*x3^2*(x1+6*x2);

% rod 3
g1=s_allow-P/x1*(2+(x1*x2*x3*(2*2^0.5*x1+x3))/DT);
% rod 7
g2=s_allow-P/x3*(2+2^0.5*x1*x2*x3*((2*2^0.5*x1+x3))/DT);

n = 1e2;
```

```
rho = 0;
Z = mvnrnd([mu1 mu2 mu3], [(cv*mu1)^2 rho rho; rho (cv*mu2)^2 rho; rho rho
(cv*mu3)^2], n);
%plot(Z(:,1),Z(:,3),'.');
counter1=0;
counter2=0;

for i=1:n
    x1=Z(i,1);
    x2=Z(i,2);
    x3=Z(i,3);
    temp(i,1)=eval(g1);
    temp(i,2)=eval(g2);
    if temp(i,1)<=0
        counter1=counter1+1;
    end
    if temp(i,2)<=0
        counter2=counter2+1;
    end
end

Pf1=counter1/n
Pf2=counter2/n




%% code 14 Compute beta by HL method with RV's uncorrelated:
% external load; mean rod area; coeff. of variance;
clear;
P=3000;
s_allow=100e6;
mu1=1e-4;mu2=0.3e-4;mu3=1e-4;
cv=0.2;
% standard variance of rod section area
var1=mu1*cv;var2=mu2*cv;var3=mu3*cv;
% xi:rod section area; pi:differential of limit-state func gi;
syms x1 x2 x3 p1 p2 p3;
DT=4*x2^2*(8*x1^2+x3^2)+4*2^0.5*x1*x2*x3*(3*x1+4*x2)+x1*x3^2*(x1+6*x2);
% rod 3
g=s_allow-P/x1*(2+(x1*x2*x3*(2*2^0.5*x1+x3))/DT);
% rod 7
%g=s_allow-P/x3*(2+2^0.5*x1*x2*x3*((2*2^0.5*x1+x3))/DT);

% partial diff. of g wrt three variables x1 x2 x3
p1=diff(g,x1);
p2=diff(g,x2);
p3=diff(g,x3);
den=((p1*var1)^2+(p2*var2)^2+(p3*var3)^2)^0.5;
x1=mu1;
x2=mu2;
x3=mu3;

i_max=10;
cosine=zeros(i_max,3);
```

```
beta=zeros(i_max,1);
X=zeros(i_max,3);
U=zeros(i_max,3);
temp_g=zeros(i_max,1);
temp_p=zeros(i_max,3);

i=1;
beta_pre=0;
beta(i)=eval(g/den);
X(i,:)=[mu1 mu2 mu3];
U(i,:)=[0 0 0];

% stop criteria: beta converges or more than 10 iterations
while [abs((beta(i)-beta_pre)/beta(i))>0.0001 i<=i_max]
    i=i+1;
    x1 = X(i-1,1); x2 = X(i-1,2); x3 = X(i-1,3);
    u1 = U(i-1,1); u2 = U(i-1,2); u3 = U(i-1,3);
    % compute the gradient and limit state value at each point.
    temp_p(i-1,1)=eval(p1);
    temp_p(i-1,2)=eval(p2);
    temp_p(i-1,3)=eval(p3);
    temp_g(i-1)=eval(g);
    % compute cosine values based on previous Xi
    cosine(i-1,1)=eval(-(p1*var1)/den);
    cosine(i-1,2)=eval(-(p2*var2)/den);
    cosine(i-1,3)=eval(-(p3*var3)/den);
    % compute new Ui
    U(i,1) = beta(i-1)*cosine(i-1,1);
    U(i,2) = beta(i-1)*cosine(i-1,2);
    U(i,3) = beta(i-1)*cosine(i-1,3);
    % compute new Xi
    X(i,1) = mu1+var1*U(i,1);
    X(i,2) = mu2+var2*U(i,2);
    X(i,3) = mu3+var3*U(i,3);
    % compute new beta
    x1=X(i,1);x2=X(i,2);x3=X(i,3);
    beta(i)=eval((g-(p1*var1*U(i,1)+p2*var2*U(i,2)+p3*var3*U(i,3)))/den);
    beta_pre=beta(i-1);
end
normcdf(-beta)




%% code 15 test of example in Text P94. get the same results
%
clear;
P=3000;
s_allow=100e6;
mu1=10;mu2=10;

% standard variance of rod section area
var1=5;var2=5;var3=5;
% xi:rod section area; pi:differential of limit-state func gi;
syms x1 x2 x3 p1 p2 p3;
```

```
% rod 3
g=x1^3+x2^3-18;
% rod 7
%g=s_allow-P/x3*(2+2^0.5*x1*x2*x3*((2*2^0.5*x1+x3))/DT);

% partial diff. of g wrt three variables x1 x2 x3
p1=diff(g,x1);
p2=diff(g,x2);
den=((p1*var1)^2+(p2*var2)^2)^0.5;
x1=mu1;
x2=mu2;



i_max=10;
cosine=zeros(i_max,2);
beta=zeros(i_max,1);
X=zeros(i_max,2);
U=zeros(i_max,2);
temp_g=zeros(i_max,1);
temp_p=zeros(i_max,2);

i=1;
beta_pre=0;
beta(i)=eval(g/den);
X(i,:)=[mu1 mu2];
U(i,:)=[0 0];

% stop criteria: beta converges or more than 10 iterations
while [abs((beta(i)-beta_pre)/beta(i))>0.001 i<=i_max]
    i=i+1;
    x1 = X(i-1,1); x2 = X(i-1,2);
    u1 = U(i-1,1); u2 = U(i-1,2);
    % compute cosine values based on previous Xi
    temp_p(i-1,1)=eval(p1);
    temp_p(i-1,2)=eval(p2);
    temp_g(i-1)=eval(g);
    cosine(i-1,1)=eval(-(p1*var1)/den);
    cosine(i-1,2)=eval(-(p2*var2)/den);
    % compute new Ui
    U(i,1) = beta(i-1)*cosine(i-1,1);
    U(i,2) = beta(i-1)*cosine(i-1,2);
    % compute new Xi
    X(i,1) = mu1+var1*U(i,1);
    X(i,2) = mu2+var2*U(i,2);
    % compute new beta
    x1=X(i,1);x2=X(i,2);
    beta(i)=eval((g-(p1*var1*U(i,1)+p2*var2*U(i,2)))/den);
    beta_pre=beta(i-1);
end




%% code 16 MCS for rod 3,7 with three RV's correlated to each other by
rho=0.8
```

```
clear;
P=3000;
s_allow=100e6;
mu1=1e-4;mu2=0.3e-4;mu3=1e-4;
cv=0.2; % coefficient of variance

% xi:rod section area; pi:differential of limit-state func gi;
syms x1 x2 x3 p1 p2 p3;
DT=4*x2^2*(8*x1^2+x3^2)+4*2^0.5*x1*x2*x3*(3*x1+4*x2)+x1*x3^2*(x1+6*x2);

% rod 3
g1=s_allow-P/x1*(2+(x1*x2*x3*(2*2^0.5*x1+x3))/DT);
% rod 7
g2=s_allow-P/x3*(2+2^0.5*x1*x2*x3*((2*2^0.5*x1+x3))/DT);

n = 5e3;
rho = 0.8;
Z = mvnrnd([mu1 mu2 mu3], [(cv*mu1)^2 rho*(cv*mu1)*(cv*mu2)
rho*(cv*mu1)*(cv*mu3); rho*(cv*mu2)*(cv*mu1) (cv*mu2)^2 rho*(cv*mu2)*(cv*mu3);
rho*(cv*mu3)*(cv*mu1) rho*(cv*mu3)*(cv*mu2) (cv*mu3)^2], n);
%plot(Z(:,1),Z(:,3),'.');
counter1=0;
counter2=0;

for i=1:n
    x1=Z(i,1);
    x2=Z(i,2);
    x3=Z(i,3);
    temp(i,1)=eval(g1);
    temp(i,2)=eval(g2);
    if temp(i,1)<=0
        counter1=counter1+1;
    end
    if temp(i,2)<=0
        counter2=counter2+1;
    end
end

Pf1=counter1/n
Pf2=counter2/n




%% code 17 Compute beta by correlated HL method with RV's correlation coeff
rho=0.8:
% external load; mean rod area; coeff. of variance;
clear;
P=3000;
s_allow=100e6;
mu1=1e-4;mu2=0.3e-4;mu3=1e-4;
cv=0.2;%coefficient of variance for each RV
rho=0.8; %coeff. of correlation for each couple of 2 RV's
% standard variance of rod section area
var1=mu1*cv;var2=mu2*cv;var3=mu3*cv;
```

```
% xi:rod section area; pi:differential of limit-state func gi;
syms x1 x2 x3 p1 p2 p3;
DT=4*x2^2*(8*x1^2+x3^2)+4*2^0.5*x1*x2*x3*(3*x1+4*x2)+x1*x3^2*(x1+6*x2);
% rod 3
g=s_allow-P/x1*(2+(x1*x2*x3*(2*2^0.5*x1+x3))/DT);
% rod 7
%g=s_allow-P/x3*(2+2^0.5*x1*x2*x3*((2*2^0.5*x1+x3))/DT);

% partial diff. of g wrt three variables x1 x2 x3
p1=diff(g,x1);
p2=diff(g,x2);
p3=diff(g,x3);
corr=2*(p1*p2*var1*var2*rho+p1*p3*var1*var3*rho+p2*p3*var2*var3*rho);
den=((p1*var1)^2+(p2*var2)^2+(p3*var3)^2+corr)^0.5;
x1=mu1;
x2=mu2;
x3=mu3;

i_max=10;
cosine=zeros(i_max,3);
beta=zeros(i_max,1);
X=zeros(i_max,3);
U=zeros(i_max,3);
temp_g=zeros(i_max,1);
temp_p=zeros(i_max,3);

i=1;
beta_pre=0;
beta(i)=eval(g/den);
X(i,:)=[mu1 mu2 mu3];
U(i,:)=[0 0 0];

% stop criteria: beta converges or more than 10 iterations
while [abs((beta(i)-beta_pre)/beta(i))>0.0001 i<=i_max]
    i=i+1;
    x1 = X(i-1,1); x2 = X(i-1,2); x3 = X(i-1,3);
    u1 = U(i-1,1); u2 = U(i-1,2); u3 = U(i-1,3);
    % compute the gradient and limit state value at each point.
    temp_p(i-1,1)=eval(p1);
    temp_p(i-1,2)=eval(p2);
    temp_p(i-1,3)=eval(p3);
    temp_g(i-1)=eval(g);
    % compute cosine values based on previous Xi
    cosine(i-1,1)=eval(-(p1*var1)/den);
    cosine(i-1,2)=eval(-(p2*var2)/den);
    cosine(i-1,3)=eval(-(p3*var3)/den);
    % compute new Ui
    U(i,1) = beta(i-1)*cosine(i-1,1);
    U(i,2) = beta(i-1)*cosine(i-1,2);
    U(i,3) = beta(i-1)*cosine(i-1,3);
    % compute new Xi
    X(i,1) = mu1+var1*U(i,1);
    X(i,2) = mu2+var2*U(i,2);
    X(i,3) = mu3+var3*U(i,3);
    % compute new beta
    x1=X(i,1);x2=X(i,2);x3=X(i,3);
    beta(i)=eval((g-(p1*var1*U(i,1)+p2*var2*U(i,2)+p3*var3*U(i,3)))/den);
    beta_pre=beta(i-1);
end
normcdf(-beta)





%% code 18 Compute beta by correlated HL method with RV's correlation coeff.
% rho=0.8:, Cosine modified
% external load; mean rod area; coeff. of variance;
clear;
P=3000;
s_allow=100e6;
mu1=1e-4;mu2=0.3e-4;mu3=1e-4;
cv=0.2;%coefficient of variance for each RV
rho=0.8; %coeff. of correlation for each couple of 2 RV's
% standard variance of rod section area
var1=mu1*cv;var2=mu2*cv;var3=mu3*cv;
% xi:rod section area; pi:differential of limit-state func gi;
syms x1 x2 x3 p1 p2 p3;
DT=4*x2^2*(8*x1^2+x3^2)+4*2^0.5*x1*x2*x3*(3*x1+4*x2)+x1*x3^2*(x1+6*x2);
% rod 3
g=s_allow-P/x1*(2+(x1*x2*x3*(2*2^0.5*x1+x3))/DT);
% rod 7
%g=s_allow-P/x3*(2+2^0.5*x1*x2*x3*((2*2^0.5*x1+x3))/DT);

% partial diff. of g wrt three variables x1 x2 x3
p1=diff(g,x1);
p2=diff(g,x2);
p3=diff(g,x3);
corr=2*(p1*p2*var1*var2*rho+p1*p3*var1*var3*rho+p2*p3*var2*var3*rho);
den=((p1*var1)^2+(p2*var2)^2+(p3*var3)^2+corr)^0.5;
x1=mu1;
x2=mu2;
x3=mu3;

i_max=10;
cosine=zeros(i_max,3);
beta=zeros(i_max,1);
X=zeros(i_max,3);
U=zeros(i_max,3);
temp_g=zeros(i_max,1);
temp_p=zeros(i_max,3);

i=1;
beta_pre=0;
beta(i)=eval(g/den);
X(i,:)=[mu1 mu2 mu3];
U(i,:)=[0 0 0];

% stop criteria: beta converges or more than 10 iterations
while [abs((beta(i)-beta_pre)/beta(i))>0.0001 i<=i_max]
    i=i+1;
    x1 = X(i-1,1); x2 = X(i-1,2); x3 = X(i-1,3);
    u1 = U(i-1,1); u2 = U(i-1,2); u3 = U(i-1,3);
    % compute the gradient and limit state value at each point.
    temp_p(i-1,1)=eval(p1);
    temp_p(i-1,2)=eval(p2);
    temp_p(i-1,3)=eval(p3);
    temp_g(i-1)=eval(g);
    % compute cosine values based on previous Xi
    cosine(i-1,1)=eval(-(p1*var1)/((p1*var1)^2+(p2*var2)^2+(p3*var3)^2)^0.5);
    cosine(i-1,2)=eval(-(p2*var2)/((p1*var1)^2+(p2*var2)^2+(p3*var3)^2)^0.5);
    cosine(i-1,3)=eval(-(p3*var3)/((p1*var1)^2+(p2*var2)^2+(p3*var3)^2)^0.5);
    % compute new Ui
    U(i,1) = beta(i-1)*cosine(i-1,1);
    U(i,2) = beta(i-1)*cosine(i-1,2);
    U(i,3) = beta(i-1)*cosine(i-1,3);
    % compute new Xi
    X(i,1) = mu1+var1*U(i,1);
    X(i,2) = mu2+var2*U(i,2);
    X(i,3) = mu3+var3*U(i,3);
    % compute new beta
    x1=X(i,1);x2=X(i,2);x3=X(i,3);
    beta(i)=eval((g-(p1*var1*U(i,1)+p2*var2*U(i,2)+p3*var3*U(i,3)))/den);
    beta_pre=beta(i-1);
end
normcdf(-beta)




%% code 19 Kriging model for rod 3: computing Kriging matrix and parameters
% need store these data in the file kriging.mat:
% C1inv: the inverse matrix of covariance matrix;
% points: sampling points coordinates;
% dmax: the maximum sampling points distance, for covariance function use;
% G: the function value at eath sampling point.
clear;
P=3000;
s_allow=100e6;
mu1=1e-4;mu2=0.3e-4;mu3=1e-4;
cv=0.2; % coefficient of variance
% standard variance
sv1=cv*mu1;sv2=cv*mu2;sv3=cv*mu3;
% xi:rod section area; pi:differential of limit-state func gi;
syms x1 x2 x3 p1 p2 p3;
DT=4*x2^2*(8*x1^2+x3^2)+4*2^0.5*x1*x2*x3*(3*x1+4*x2)+x1*x3^2*(x1+6*x2);

% rod 3
g1=s_allow-P/x1*(2+(x1*x2*x3*(2*2^0.5*x1+x3))/DT);
% rod 7
g2=s_allow-P/x3*(2+2^0.5*x1*x2*x3*((2*2^0.5*x1+x3))/DT);

x=(-2*sv1: sv1 :2*sv1); x=x+mu1;
y=(-2*sv2: sv2 :2*sv2); y=y+mu2;
z=(-2*sv3: sv3 :2*sv3); z=z+mu3;

[X,Y,Z]=meshgrid(x,y,z);
```

```
x1=X; x2=Y; x3=Z;
A=eval(g2);
% holding all interpolation points, totally 125 points
points=zeros(3,5^3);
G=zeros(5^3+1,1);
counter=1;
for i=1:5
    for j=1:5
        for k=1:5
            points(1,counter)=x(i);
            points(2,counter)=y(j);
            points(3,counter)=z(k);
            % Attention for the sequence of 'A(j,i,k)', not 'A(i,j,k)'
            G(counter)=A(j,i,k);
            counter=counter+1;
        end
    end
end
% distance matrix of each pair of points
d=dist(points);
dmax=1.156546583584077e-04;
% Covariance matrix between each pair of points:
C=1-1.5*(d/dmax)+0.5*(d/dmax).^3;
% Covariance matrix for Ordinary Kriging
C1=[C ones(125,1);ones(1,125) 0];
C1inv=C1^-1;
```

```
%% code 20 Kriging model for rod 3 and 7: invoking the kriging.m function:

x=[1e-4 0.3e-4 1e-4;
1e-4 0.3e-4 0.6e-4;
1e-4 0.3e-4 0.8e-4;
1e-4 0.3e-4 1.2e-4;
1e-4 0.3e-4 1.4e-4;
0.6e-4 0.3e-4 1e-4;
0.8e-4 0.3e-4 1e-4;
1.2e-4 0.3e-4 1e-4;
1.4e-4 0.3e-4 1e-4;]'
kriging01(x)
```

```
%% code 21 Kriging model: MCS with Kriging uncorrelated
clear;
P=3000;
s_allow=100e6;
mu1=1e-4;mu2=0.3e-4;mu3=1e-4;
cv=0.2;

% xi:rod section area; pi:differential of limit-state func gi;
```

```
syms x1 x2 x3 p1 p2 p3;
DT=4*x2^2*(8*x1^2+x3^2)+4*2^0.5*x1*x2*x3*(3*x1+4*x2)+x1*x3^2*(x1+6*x2);

% rod 3
g1=s_allow-P/x1*(2+(x1*x2*x3*(2*2^0.5*x1+x3))/DT);
% rod 7
g2=s_allow-P/x3*(2+2^0.5*x1*x2*x3*((2*2^0.5*x1+x3))/DT);

n = 1e4;
rho = 0;
X = mvnrnd([mu1 mu2 mu3], [(cv*mu1)^2 rho rho; rho (cv*mu2)^2 rho; rho rho
(cv*mu3)^2], n);
%plot(X(:,1),X(:,3),'.');
X=X';
Y1=kriging01(X);
Y2=kriging02(X);
counter1=0;
counter2=0;
for i=1:n
    if Y1(i)<0
        counter1=counter1+1;
    end
    if Y2(i)<0
        counter2=counter2+1;
    end
end
Pf1=counter1/n
Pf2=counter2/n



%% code 22 Kriging model: MCS with Kriging uncorrelated, comparing with real
value
clear;
P=3000;
s_allow=100e6;
mu1=1e-4;mu2=0.3e-4;mu3=1e-4;
cv=0.2;

% xi:rod section area; pi:differential of limit-state func gi;
syms x1 x2 x3 p1 p2 p3;
DT=4*x2^2*(8*x1^2+x3^2)+4*2^0.5*x1*x2*x3*(3*x1+4*x2)+x1*x3^2*(x1+6*x2);

% rod 3
g1=s_allow-P/x1*(2+(x1*x2*x3*(2*2^0.5*x1+x3))/DT);
% rod 7
g2=s_allow-P/x3*(2+2^0.5*x1*x2*x3*((2*2^0.5*x1+x3))/DT);

n = 1e4;
rho = 0;
X = mvnrnd([mu1 mu2 mu3], [(cv*mu1)^2 rho rho; rho (cv*mu2)^2 rho; rho rho
(cv*mu3)^2], n);
temp1=zeros(n,1);
temp2=zeros(n,1);
%plot(X(:,1),X(:,3),'.');
```

```
% Evaluate the real value of limit state function by the RV's
% for i=1:n
%     x1=X(i,1);x2=X(i,2);x3=X(i,3);
%     temp1(i)=eval(g1);
%     temp2(i)=eval(g2);
% end
%Evaluate by surrogate model: Kriging
X=X';
Y1=kriging01(X);
Y2=kriging02(X);
counter1=0;
counter2=0;
for i=1:n
    if Y1(i)<0
        counter1=counter1+1;
    end
    if Y2(i)<0
        counter2=counter2+1;
    end
end
Pf1=counter1/n
Pf2=counter2/n
```

```
%% code 23 Plotting the covariance function of Kriging
x=(0:1:100);
y=1-1.5*(x./100)+0.5*(x./100).^3;
plot(x,y,'-')
```

```
%% code 24 MCS for Example P132
clear;
mu1=10;mu2=10;
var1=5;var2=5;
% xi:rod section area; pi:differential of limit-state func gi;
syms x1 x2 ;

g1=x1^4+2*x2^4-20;
n = 1e4;
rho = 0;
Z = mvnrnd([mu1 mu2], [(var1)^2 rho ; rho (var2)^2], n);
%plot(Z(:,1),Z(:,2),'.');
counter1=0;
temp=zeros(n,1);
for i=1:n
    x1=Z(i,1);
    x2=Z(i,2);
    temp(i)=eval(g1);
    if temp(i)<=0
        counter1=counter1+1;
    end
end
```

```
Pf1=counter1/n




%% code 25 Test Gram-Schmidt orthogonaizing:
A=[1/2^0.5 -1/2^0.5; 0 1]';
Gram_Schmidt_Process(A);




%% code 26 Ploting the limit state function

clear;
P=3000;
s_allow=100e6;
mu1=1e-4;mu2=0.3e-4;mu3=1e-4;
cv=0.2;

% xi:rod section area; pi:differential of limit-state func gi;
syms x1 x2 x3 p1 p2 p3;
DT=4*x2^2*(8*x1^2+x3^2)+4*2^0.5*x1*x2*x3*(3*x1+4*x2)+x1*x3^2*(x1+6*x2);

% rod 3
g1=s_allow-P/x1*(2+(x1*x2*x3*(2*2^0.5*x1+x3))/DT);
% rod 7
g2=s_allow-P/x3*(2+2^0.5*x1*x2*x3*((2*2^0.5*x1+x3))/DT);

x01=(mu1-2*cv*mu1: cv*mu1/20 : mu1+2*cv*mu1);
x02=mu2;
x03=(mu3-2*cv*mu3: cv*mu3/20 : mu3+2*cv*mu3);

x1=x01; x2=x02; x3=x03;
[X,Y]=meshgrid(x1,x3);
x1=X; x3=Y;
y=eval(g1);
surf(x1,x3,y);




%% code 27 SORM Breitung

clear;
P=3000;
s_allow=100e6;
mu1=1e-4;mu2=0.3e-4;mu3=1e-4;
cv=0.2;
var1=cv*mu1;var2=cv*mu2;var3=cv*mu3;
% xi:rod section area; pi:differential of limit-state func gi;
syms x1 x2 x3 p1 p2 p3;
DT=4*x2^2*(8*x1^2+x3^2)+4*2^0.5*x1*x2*x3*(3*x1+4*x2)+x1*x3^2*(x1+6*x2);

% rod 3
g1=s_allow-P./x1.*(2+(x1.*x2.*x3.*(2*2^0.5*x1+x3))./DT);
% rod 7
```

```
g2=s_allow-P/x3*(2+2^0.5*x1*x2*x3*((2*2^0.5*x1+x3))/DT);

% g=g1;
g=g2;

px01=diff(g,x1);
px02=diff(g,x2);
px03=diff(g,x3);

pu01=px01*var1;
pu02=px02*var2;
pu03=px03*var3;

px11=diff((diff(g,x1)),x1);
px12=diff((diff(g,x1)),x2);
px13=diff((diff(g,x1)),x3);
px22=diff((diff(g,x2)),x2);
px23=diff((diff(g,x2)),x3);
px33=diff((diff(g,x3)),x3);

pu11=px11*var1*var1;
pu12=px12*var1*var2;
pu13=px13*var1*var3;
pu22=px22*var2*var2;
pu23=px23*var2*var3;
pu33=px33*var3*var3;

%rod 3
% beta=1.8765;x1=6.2491e-5;x2=2.9894e-5;x3=1.0022e-4;
%rod7
beta=1.8388;x1=1.001e-4;x2=2.9836e-5;x3=6.3228e-5;

d=eval((pu01^2+pu02^2+pu03^2).^0.5);
B=eval([pu11 pu12 pu13; pu12 pu22 pu23; pu13 pu23 pu33]./d);
H0=eval([-pu01./d -pu02./d -pu03./d; 0 1 0; 0 0 1].');

H1=Gram_Schmidt_Process(H0);
H1=H1';
H=[H1(2,:);H1(3,:);H1(1,:)];

temp=H*B*H';
temp=[temp(1,1) temp(1,2);temp(2,1) temp(2,2)];
[vector,value]=eig(temp);

k1=value(1,1);
k2=value(2,2);

Pf=normcdf(-beta)*(1+k1*beta)^0.5*(1+k2*beta)^0.5;




%% code 28 Preparing P_ini[] for the next program:
solve_u1(0,0)
```

```
solve_u3(0,0)

%% code 29 Gradient Projection Method for safety-index
clear;
P=3000;
s_allow=100e6;%max permissible stress
mu1=1e-4;mu2=0.3e-4;mu3=1e-4;
cv=0.2;% coeff of variance
var1=cv*mu1;var2=cv*mu2;var3=cv*mu3;%standard variance
alpha=0.5; %step size important
i_max=20;

U=zeros(i_max,3);%U at each step
Ul=zeros(10,3);% local loops to approach the constraint surface
X=zeros(i_max,3);% X at each step
Y=zeros(i_max,1);% Y at each step
error=zeros(i_max,1);% Y errors at each step
Grad_O=zeros(i_max,3);% gradient of objective func
Grad_C=zeros(i_max,3);% gradient of constraint func

% initial point, must be on the constraint plane
P_ini=[-1.8757 0 0];% for rod3
% P_ini=[-1.8706 0 1];
% P_ini=[-1.8847 1 0];
% P_ini=[-1.8791 1 1];
% P_ini=[-1.8830 0 -1];
% P_ini=[-1.8668 0 2];
% P_ini=[-1.8937 0 -2];
% P_ini=[-1.8687 -2 -2];
% P_ini=[-1.8823 2 2];
% P_ini=[-1.8520 -2 2];
% P_ini=[-1.9118 2 -2];

% P_ini=[0 0 -1.8390];% for rod7

% xi:rod section area; pi:differential of limit-state func gi;
syms x1 x2 x3 u1 u2 u3 pc1 pc2 pc3 po1 po2 po3 y;

%Objective function, need be minimized
y=(u1^2+u2^2+u3^2)^0.5;

x1=mu1+u1*var1; x2=mu2+u2*var2; x3=mu3+u3*var3;
DT=4*x2^2*(8*x1^2+x3^2)+4*2^0.5*x1*x2*x3*(3*x1+4*x2)+x1*x3^2*(x1+6*x2);
% rod 3
g=s_allow-P/x1*(2+(x1*x2*x3*(2*2^0.5*x1+x3))/DT);
% rod 7
% g=s_allow-P/x3*(2+2^0.5*x1*x2*x3*((2*2^0.5*x1+x3))/DT);

% first order partial direvatives of objective  function wrt ui
po1=diff(y,u1);
po2=diff(y,u2);
po3=diff(y,u3);
% first order partial direvatives of constraint function wrt ui
pc1=diff(g,u1);
pc2=diff(g,u2);
```

```
pc3=diff(g,u3);

i=1;
U(i,:)= P_ini;
u1=U(i,1); u2=U(i,2); u3=U(i,3);
X(i,:)=eval([x1 x2 x3]);
Y(i)=eval(y);
error(i)=1;

while [error(i)>0.0001 i<i_max]
    Grad_O(i,1)=eval(po1);
    Grad_O(i,2)=eval(po2);
    Grad_O(i,3)=eval(po3);

    Grad_C(i,1)=eval(pc1);
    Grad_C(i,2)=eval(pc2);
    Grad_C(i,3)=eval(pc3);

    A=Grad_C(i,:);
    P=eye(3)-A.'*(A*A.')^(-1)*A;
    Ul(1,:)=(U(i,:).'-alpha*P*Grad_O(i,:).').'; %point on tangent plane
    %---------
    % iteratively compute new point on the constraint plane
    %---------
    j=1;
    while (j<3) % return from the tangent plane to the constraint plane
        u1=Ul(j,1);u2=Ul(j,2);u3=Ul(j,3);
        temp0=eval(g);% value of constraint function at the new point
        temp1=[eval(pc1) eval(pc2) eval(pc3)]; %gradient of the constraint
function at the new point
        temp2=Ul(j,:).'-(temp1).'*(temp1*temp1.')^(-1)*(temp0); % next new
point
        Ul(j+1,:)=temp2.';
        j=j+1;
    end
    U(i+1,:)=Ul(j,:);
    i=i+1;
    u1=U(i,1); u2=U(i,2); u3=U(i,3);
    %---------
    % check if the objective function value at the new
    % point is converging, if yes then stop iteration
    %---------
    X(i,:)=eval([x1 x2 x3]);
    Y(i)=eval(y);
    error(i)=abs(Y(i)-Y(i-1))/(Y(i));
end
Pf=normcdf(-Y);




%% code 30 Reduced Gradient Method for safety-index rod3
clear;
P=3000;
```

```
s_allow=100e6;%max permissible stress
mu1=1e-4;mu2=0.3e-4;mu3=1e-4;
cv=0.2;% coeff of variance
var1=cv*mu1;var2=cv*mu2;var3=cv*mu3;%standard variance
alpha=0.5; %step size important
i_max=20;

% xi:rod section area; pi:differential of limit-state func gi;
syms x1 x2 x3 u1 u2 u3 pc1 pc2 pc3 po1 po2 po3 y;
%Objective function, need be minimized
y=(u1^2+u2^2+u3^2)^0.5;

x1=mu1+u1*var1; x2=mu2+u2*var2; x3=mu3+u3*var3;
DT=4*x2^2*(8*x1^2+x3^2)+4*2^0.5*x1*x2*x3*(3*x1+4*x2)+x1*x3^2*(x1+6*x2);
% rod 3
g=s_allow-P/x1*(2+(x1*x2*x3*(2*2^0.5*x1+x3))/DT);
% rod 7
% g=s_allow-P/x3*(2+2^0.5*x1*x2*x3*((2*2^0.5*x1+x3))/DT);

% first order partial direvatives of objective  function wrt ui
po1=diff(y,u1);
po2=diff(y,u2);
po3=diff(y,u3);
% first order partial direvatives of constraint function wrt ui
pc1=diff(g,u1);
pc2=diff(g,u2);
pc3=diff(g,u3);

% set u2, u3 at decision variables, u1 as state variable
% the following are the partial derivatives:
pfpd=[po2 po3];
pfps=[po1];
phpd=[pc2 pc3];
phps=[pc1];

u_ini=[0 -1 -1]; %initial value for decision varialbes
u=zeros(i_max,3);% each step point u
x=zeros(i_max,3);% each step point x
Y=zeros(i_max,1);% objective function value
s=zeros(i_max,4);% inner loop for u1 to return to constraint surface
error=zeros(i_max,1);% Y errors at each step

i=1;
u(i,2)=u_ini(2);
u(i,3)=u_ini(3);
alpha=0.5; % step size
u2=u(i,2);u3=u(i,3);
u1=solve_u1(u2,u3);
u(i,1)=u1; %compute s_k, store it in u()
x(i,1)=mu1+var1*u(i,1);
x(i,2)=mu2+var2*u(i,2);
x(i,3)=mu3+var3*u(i,3);
Y(i)=eval(y);
error(i)=1;
```

```
while [error(i)>0.0001 i<i_max]
    ev_fd=eval(pfpd);
    ev_fs=eval(pfps);
    ev_hd=eval(phpd);
    ev_hs=eval(phps);

    ev_zd=ev_fd-ev_fs*ev_hs^(-1)*ev_hd;

    %******************
    % compute d_k+1 or u2 u3
    %******************
    d_new=[u(i,2) u(i,3)]'-alpha*ev_zd';
    %******************
    % compute s_k+1'
    %******************
    s_new0=u1+alpha*ev_hs^(-1)*ev_hd*ev_zd';
    %******************
    % s_k+1' return s to constraint plane as s_k+1 or u1
    %******************
    s(i,1)=s_new0;
    for j=1:2
        u2=d_new(1);u3=d_new(2);u1=s_new0;
        s_new1=s_new0-ev_hs^(-1)*eval(g);
        s_new0=s_new1;
        s(i,j+1)=s_new0;
    end

    %******************
    % update u() and x() and Y()
    %******************
    u(i+1,2)=d_new(1);u(i+1,3)=d_new(2);u(i+1,1)=s(i,j+1);
    x(i+1,1)=mu1+var1*u(i+1,1);
    x(i+1,2)=mu2+var2*u(i+1,2);
    x(i+1,3)=mu3+var3*u(i+1,3);
    i=i+1;
    u1=u(i,1);u2=u(i,2);u3=u(i,3);
    Y(i)=eval(y);
    error(i)=abs(Y(i)-Y(i-1))/(Y(i));
end
Pf=normcdf(-Y);
```

```
%% code 31 Reduced Gradient Method for safety-index rod7
clear;
P=3000;
s_allow=100e6;%max permissible stress
mu1=1e-4;mu2=0.3e-4;mu3=1e-4;
cv=0.2;% coeff of variance
var1=cv*mu1;var2=cv*mu2;var3=cv*mu3;%standard variance
alpha=0.5; %step size important
i_max=20;
```

```
% xi:rod section area; pi:differential of limit-state func gi;
syms x1 x2 x3 u1 u2 u3 pc1 pc2 pc3 po1 po2 po3 y;
%Objective function, need be minimized
y=(u1^2+u2^2+u3^2)^0.5;

x1=mu1+u1*var1; x2=mu2+u2*var2; x3=mu3+u3*var3;
DT=4*x2^2*(8*x1^2+x3^2)+4*2^0.5*x1*x2*x3*(3*x1+4*x2)+x1*x3^2*(x1+6*x2);
% rod 3
% g=s_allow-P/x1*(2+(x1*x2*x3*(2*2^0.5*x1+x3))/DT);
% rod 7
g=s_allow-P/x3*(2+2^0.5*x1*x2*x3*((2*2^0.5*x1+x3))/DT);

% first order partial direvatives of objective  function wrt ui
po1=diff(y,u1);
po2=diff(y,u2);
po3=diff(y,u3);
% first order partial direvatives of constraint function wrt ui
pc1=diff(g,u1);
pc2=diff(g,u2);
pc3=diff(g,u3);

% set u2, u3 at decision variables, u1 as state variable
% the following are the partial derivatives:
pfpd=[po1 po2];
pfps=[po3];
phpd=[pc1 pc2];
phps=[pc3];

u_ini=[1 2 0]; %initial value for decision varialbes
u=zeros(i_max,3);% each step point u
x=zeros(i_max,3);% each step point x
Y=zeros(i_max,1);% objective function value
s=zeros(i_max,4);% inner loop for u1 to return to constraint surface
error=zeros(i_max,1);% Y errors at each step

i=1;
u(i,1)=u_ini(1);
u(i,2)=u_ini(2);
alpha=0.5; % step size
u1=u(i,1);u2=u(i,2);
u3=solve_u3(u1,u2);
u(i,3)=u3; %compute s_k, store it in u()
x(i,1)=mu1+var1*u(i,1);
x(i,2)=mu2+var2*u(i,2);
x(i,3)=mu3+var3*u(i,3);
Y(i)=eval(y);
error(i)=1;

while [error(i)>0.0001 i<i_max]
    ev_fd=eval(pfpd);
    ev_fs=eval(pfps);
    ev_hd=eval(phpd);
    ev_hs=eval(phps);

    ev_zd=ev_fd-ev_fs*ev_hs^(-1)*ev_hd;
    %******************
    % compute d_k+1 or u1 u2
    %******************
    d_new=[u(i,1) u(i,2)]'-alpha*ev_zd';
    %******************
    % compute s_k+1'
    %******************
    s_new0=u3+alpha*ev_hs^(-1)*ev_hd*ev_zd';
    %******************
    % s_k+1' return s to constraint plane as s_k+1 or u3
    %******************
    s(i,1)=s_new0;
    for j=1:2
        u1=d_new(1);u2=d_new(2);u3=s_new0;
        s_new1=s_new0-ev_hs^(-1)*eval(g);
        s_new0=s_new1;
        s(i,j+1)=s_new0;
    end

    %******************
    % update u() and x() and Y()
    %******************
    u(i+1,1)=d_new(1);u(i+1,2)=d_new(2);u(i+1,3)=s(i,j+1);
    x(i+1,1)=mu1+var1*u(i+1,1);
    x(i+1,2)=mu2+var2*u(i+1,2);
    x(i+1,3)=mu3+var3*u(i+1,3);
    i=i+1;
    u1=u(i,1);u2=u(i,2);u3=u(i,3);
    Y(i)=eval(y);
    error(i)=abs(Y(i)-Y(i-1))/(Y(i));
end
Pf=normcdf(-Y);
```

```
%% Code 32
n = 500;
figure;
rho = 0.0;
Z = mvnrnd([0 0], [1 rho; rho 1], n);
X = exp(Z);
U = normcdf(Z);
subplot(2,2,1)
plot(Z(:,1),Z(:,2),'.'); xlim([-4 4]); ylim([-4 4]);
subplot(2,2,3)
plot(X(:,1),X(:,2),'.'); xlim([0 10]); ylim([0 10]);
rho = 0.8;
Z = mvnrnd([0 0], [1 rho; rho 1], n);
X = exp(Z);
U = normcdf(Z);
subplot(2,2,2)
plot(Z(:,1),Z(:,2),'.'); xlim([-4 4]); ylim([-4 4]);
subplot(2,2,4)
```

```
plot(X(:,1),X(:,2),'.'); xlim([0 10]); ylim([0 10]);




%% Code 33
n = 500;
figure;
rho = 0.8;
Z = mvnrnd([0 0], [1 rho; rho 1], n);
U = normcdf(Z);
subplot(2,2,1);
plot(Z(:,1),Z(:,2),'.'); xlim([-4 4]); ylim([-4 4]);
title('X: correlated bivariate Gaussian');
subplot(2,2,2);
plot(U(:,1),U(:,2),'.');
title('U: after taking CDF transform');
subplot(2,2,3);
hist(U(:,1),10);
title('histogram of U1');
subplot(2,2,4);
hist(U(:,2),10);
title('histogram of U2');




%% Code 34
n = 500;
figure;
rho = 0.8;
Z = mvnrnd([0 0], [1 rho; rho 1], n);
U = normcdf(Z);
X = [gaminv(U(:,1),2,1) tinv(U(:,2),5)];
subplot(1,3,1)
plot(Z(:,1),Z(:,2),'.'); xlim([-4 4]); ylim([-4 4]);
subplot(1,3,2)
plot(U(:,1),U(:,2),'.');
subplot(1,3,3)
plot(X(:,1),X(:,2),'.');




%% Code 35
n = 500;
figure;
rho = 0.8;
Z = mvnrnd([0 0], [1 rho; rho 1], n);
U = normcdf(Z);
X = [gaminv(U(:,1),2,1) gaminv(U(:,2),2,1)];
subplot(1,3,1)
plot(Z(:,1),Z(:,2),'.'); xlim([-4 4]); ylim([-4 4]);
subplot(1,3,2)
plot(U(:,1),U(:,2),'.');
subplot(1,3,3)
```

```
plot(X(:,1),X(:,2),'.');




%% Code 36
n = 500;
figure;
rho = 0.8;
Z = mvnrnd([0 0], [1 rho; rho 1], n);
U = normcdf(Z);
X = [gaminv(U(:,1),2,1) tinv(U(:,2),5)];
subplot(2,3,1)
plot(Z(:,1),Z(:,2),'.'); xlim([-8 8]); ylim([-8 8]);
subplot(2,3,2)
plot(U(:,1),U(:,2),'.');
subplot(2,3,3)
plot(X(:,1),X(:,2),'.'); xlim([0 10]); ylim([-6 6]);
Z = mvtrnd([1 rho; rho 1], 1, n);
U = tcdf(Z,1);
X = [gaminv(U(:,1),2,1) tinv(U(:,2),5)];
subplot(2,3,4)
plot(Z(:,1),Z(:,2),'.'); xlim([-8 8]); ylim([-8 8]);
subplot(2,3,5)
plot(U(:,1),U(:,2),'.');
subplot(2,3,6)
plot(X(:,1),X(:,2),'.'); xlim([0 10]); ylim([-6 6]);




%% Code 37
n = 1000;
tau = 0.7

alpha = copulaparam('Clayton',tau,'type','kendall')
U = copularnd('Clayton',alpha,n);
subplot(1,3,1)
plot(U(:,1),U(:,2),'.');
title(['Clayton Copula, {\it\theta} = ',sprintf('%0.2f',alpha)])
xlabel('U1')
ylabel('U2')

alpha = copulaparam('Frank',tau,'type','kendall');
U = copularnd('Frank',alpha,n);
subplot(1,3,2)
plot(U(:,1),U(:,2),'.')
title(['Frank Copula, {\it\theta} = ',sprintf('%0.2f',alpha)])
xlabel('U1')
ylabel('U2')

alpha = copulaparam('Gumbel',tau,'type','kendall');
U = copularnd('Gumbel',alpha,n);
subplot(1,3,3)
plot(U(:,1),U(:,2),'.')
```

```
title(['Gumbel Copula, {\it\theta} = ',sprintf('%0.2f',alpha)])
xlabel('U1')
ylabel('U2')




%% Code 38: exponential covariance function model:
syms x y;
y=exp(-abs(x));
x=[-5:0.1:5];
y=eval(y);
plot(x,y);


%% Code 38: Gaussian covariance function model:
syms x y;
y=exp(-(x)^2);
x=[-3:0.1:3];
y=eval(y);
plot(x,y);



%% Code 39: KL to reduce data dimension example
rho=0.9;
K=[1 rho;rho 1];
[V,D] = eig(K);
A=V*(D.^0.5);
X=mvnrnd([0 0], [1 0; 0 1], 100);
X=X';
Y=A*X;
figure;

% plot the uncorrelated data X
subplot(2,3,1);
plot(X(1,:),X(2,:),'.'); axis equal; xlim([-4 4]); ylim([-4 4]);
title('uncorrelated random variables');

% plot the correlated data Y
subplot(2,3,2);
plot(Y(1,:),Y(2,:),'.'); axis equal; xlim([-4 4]); ylim([-4 4]);
title('correlated random variables');

% Find the Kendall's tau
tau=copulastat('Gaussian', rho)

% Plot the coordinates of Y points projected on eigenvectors
C=V^(-1)*Y;
subplot(2,3,3);
plot(C(1,:),C(2,:),'.'); axis equal; xlim([-4 4]); ylim([-4 4]);
title('projections on eigenvectors');

% Reducing dimension
```

```
C1=C;
C1(1,:)=0;
Y1=V*C1;
X1=A^(-1)*Y1;
% subplot(2,3,6);
% plot(X1(1,:),X1(2,:),'.'); axis equal; xlim([-4 4]); ylim([-4 4]);
% title('');
subplot(2,3,5);
plot(Y1(1,:),Y1(2,:),'.'); axis equal; xlim([-4 4]); ylim([-4 4]);
title('back to the original coordinate system');
subplot(2,3,4);
plot(C1(1,:),C1(2,:),'.'); axis equal; xlim([-4 4]); ylim([-4 4]);
title('minor component discarded');




%% Code 40: Bivariate Gaussian Plot
syms x y z;
s1=1;s2=1;u1=0;u2=0;rho=0.8;
z=(1/(2*pi*s1*s2*(1-rho^2)^0.5))*exp(-(1/(2*(1-rho^2)))*(((x-u1)^2/s1^2)+((y-
u2)^2/s2^2)-((2*rho*(x-u1)*(y-u2))/(s1*s2))));
x=[-3:0.1:3];
y=[-3:0.1:3];
[X,Y]=meshgrid(x,y);
x=X;y=Y;
Z=eval(z);
mesh(X,Y,Z);
figure;
surf(X,Y,Z);
figure;
surfl(X,Y,Z);
shading interp;
colormap pink;



%% Code 41: Gumbel-Hougaard Copula CDF and PDF plotting
clear;
syms u v C c;
t=2;
C=exp(-((-log(u))^(t)+(-log(v))^(t))^(1/t));
c=diff(diff(C,u),v);
x=[0:0.05:1];
y=[0:0.05:1];
[u,v]=meshgrid(x,y);
C1=eval(C);
c1=eval(c);
figure;
surf(u,v,C1);
figure;
surfl(u,v,C1);
%shading interp;
%colormap gray;
figure;
```

```
surf(u,v,c1);
figure;
surfl(u,v,c1);
%shading interp;
%colormap gray;




%% Code 42: Finding Bilinear Interpolation function coefficients C&D
clear;
X1=1e-4 * [1 1 1.2 1.2]';
X3=1e-4 * [1 1.2 1 1.2]';
Pf3=[0.0303 0.0307 0.0083 0.0084]';
Pf7=[0.033 0.00899 0.0331 0.00904]';
A=[[1 1 1 1]' X1 X3 [X1(1)*X3(1) X1(2)*X3(2) X1(3)*X3(3) X1(4)*X3(4)]'];
C=A^(-1)*Pf3;
D=A^(-1)*Pf7;




%% Code 43: Solve the system of nonlinear functions specified by myfunc01.m
x0 = [1.0e-4; 1.0e-4];           % Make a starting guess at the solution
options=optimset('Display','iter');   % Option to display output
[x,fval] = fsolve(@myfun01,x0,options)  % Call optimizer




%% Code 44: Bilinear Interpolation illustration example
clear;
syms x y f
P11=1;P12=0;P21=0;P22=1;
f=P11*(1-x)*(1-y)+P21*(x)*(1-y)+P12*(1-x)*(y)+P22*(x)*(y);
X=[0:0.1:1];
Y=[0:0.1:1];
[x,y]=meshgrid(X,Y);
f=eval(f);
mesh(x,y,f);
figure;
surf(x,y,f);



%% Code 45: Copula boundaries illustration
clear;
syms x y f1 f2;
X=[0:0.05:1];
Y=[0:0.05:1];
[x,y]=meshgrid(X,Y);
f1=max(x+y-1, 0);
f2=min(x,y);
figure;
subplot(1,2,1);
surfl(x,y,f2);
% axis equal;
subplot(1,2,2);
surfl(x,y,f1);
```

```
% axis equal;



%% Code 46: FloorGrid g1 g2, compute Pf and material volume
clear;

%constants
stress_allow=241e6;
delta_allow=5e-3;
D=0.75; %transverse beam spacing
W_fl=6; %fuselage width
k=0.05;  %defection coefficient
E=69e9; %Young's modulus
u=2200; %mean value of plate load
var=400;%variance of plate load
c=4.2649;%norminv(1e-5)

% Design variables
b=0.0070809;
d=0.467327;
t=0.003;
h=3*b;

% stress limit-state function:
q11=stress_allow*4*b*h^2/(3*d*D^2);
q10=(q11-u)/var;
Pf1=normcdf(-q10)
W1=b*h*D*W_fl/d
% deflection limit-state function
q21=delta_allow*E*t^3/(k*d^4);
q20=(q21-u)/var;
Pf2=normcdf(-q20)
W2=W_fl*D*t



%% Code 47: Gradient Projection method to solve floor board problem
%   with only 2 limit-state Pf constraints as equality constraints
clear;
% x1, x2, x3: b, d, t
syms x1 x2 x3 f h1 h2
alpha=1.5; %step size important
i_max=30;

%constants
stress_allow=241e6;
delta_allow=5e-3;
D=0.75; %transverse beam spacing
W_fl=6; %fuselage width
k=0.05;  %defection coefficient
E=69e9; %Young's modulus
u=2200; %mean value of plate load
var=400;%variance of plate load
c=4.2649;%norminv(1e-5)
```

```
f=3*D*W_fl*(x1^2/x2)+W_fl*D*(x3);
h1=stress_allow*4*x1*(3*x1)^2/(3*x2*D^2)-(c*var+u);
h2=delta_allow*E*x3^3/(k*x2^4)-(c*var+u);

pfx1=diff(f,x1);
pfx2=diff(f,x2);
pfx3=diff(f,x3);

p1x1=diff(h1,x1);
p1x2=diff(h1,x2);
p1x3=diff(h1,x3);

p2x1=diff(h2,x1);
p2x2=diff(h2,x2);
p2x3=diff(h2,x3);

A0=[p1x1 p1x2 p1x3
    p2x1 p2x2 p2x3];
pf0=[pfx1 pfx2 pfx3];

x_ini=[0.007 0.5 0.0035]; %initial value for decision varialbes
x=zeros(i_max,3);% each step point x
Y=zeros(i_max,1);% objective function value
s=zeros(i_max,4);% inner loop for u1 to return to constraint surface
error=zeros(i_max,1);% Y errors at each step

i=1;
error(i)=1;
x(i,1)=x_ini(1);x(i,2)=x_ini(2);x(i,3)=x_ini(3);
x1=x(i,1);x2=x(i,2);x3=x(i,3);
Y(i)=eval(f);

while [error(i)>1e-5 i<i_max]
    x1=x(i,1);x2=x(i,2);x3=x(i,3);
    A=eval(A0);
    pf=eval(pf0);
    P=eye(3)-A'*(A*A')^(-1)*A;
    temp=[x1 x2 x3]'-alpha*P*pf';
    %---------------------------------
    % Move back to constraint plane
    %---------------------------------
    for j=1:2
        x1=temp(1);x2=temp(2);x3=temp(3);
        A=eval(A0);
        h=eval([h1 h2]');
        temp=temp-A'*(A*A')^(-1)*h;
    end
    x1=temp(1);x2=temp(2);x3=temp(3);
    x(i+1,1)=x1;x(i+1,2)=x2;x(i+1,3)=x3;
    Y(i+1)=eval(f);
    error(i+1)=abs(Y(i+1)-Y(i))/(Y(i+1));
    i=i+1;
end
```

```
%% Code 48: Optimization under all inequal constraints (Floor Grid Problem)
clear;
syms f h a1 a2 a3 a4 a5 a6 a7 a8 a9 ;
syms x1 x2 x3;

%model related constants
stress_allow=241e6;
delta_allow=5e-3;
D=0.75; %transverse beam spacing
W_fl=6; %fuselage width
k=0.05;  %defection coefficient
E=69e9; %Young's modulus
u=2200; %mean value of plate load
var=400;%variance of plate load
c=4.2649;%norminv(1e-5)

%algorithm related constants
SP=[0.01 0.4 0.0035].'; %starting point
% SP=[0.02 0.4 0.0035].'; %starting point
% SP=[0.01 0.6 0.0035].'; %starting point
% SP=[0.02 0.6 0.0035].'; %starting point
% SP=[0.01 0.4 0.004].'; %starting point
% SP=[0.02 0.4 0.004].'; %starting point
% SP=[0.01 0.6 0.004].'; %starting point
% SP=[0.02 0.6 0.004].'; %starting point

alpha=0.001; %step size
i_max=50;

%objective function and constraints definition
f=3*D*W_fl*(x1^2/x2)+W_fl*D*x3;
h=[ stress_allow*4*x1*(3*x1)^2/(3*x2*D^2)-(c*var+u);
    delta_allow*E*x3^3/(k*x2^4)-(c*var+u);
    x3-0.003;
    x2-0.3;
    0.75-x2];
pf=[diff(f,x1) diff(f,x2) diff(f,x3)];
ph=[diff(h,x1) diff(h,x2) diff(h,x3)];
[m,n]=size(h);

x=zeros(3,i_max);% each step point x, column vectors
xt=zeros(3,i_max);% testing point on negtive gradient direction, column
vectors
y=zeros(i_max,1);% objective function value
flag=zeros(m,i_max);% recording which constraint becomes active (+1);, column
vectors
error=zeros(i_max,1);% Y errors at each step
h1=[a1 a2 a3 a4 a5 a6 a7 a8 a9].'; % maximum constraint number:9

i=1;
error(i)=1;error(i+1)=1;
```

```
x(:,1)=SP;
while [error(i)>1e-5 i<i_max]
    x1=SP(1);x2=SP(2);x3=SP(3);
    y(i)=eval(f);
    temp=(eval(pf))';
    % normalizing gradient vector:
    temp=temp/(temp(1)^2+temp(2)^2)^0.5;
    EP=SP-alpha*temp; % Gradient Descent method
    xt(:,i+1)=EP; % testing point on -gradient direction
    x1=EP(1);x2=EP(2);x3=EP(3);
    flag0=eval(h).'; % find any constraint violation
    flag0=sign(flag0);
    flag0=flag0-1;
    flag0=sign(abs(flag0));% +1 represent constraint violation, column vector
    flag(:,i)=flag0;
    temp=sum(flag0); % test if any constraint violation
    if temp==0 % no constraint violation
        xt(:,i)=EP;
        SP=EP;
        x(:,i+1)=SP;
    elseif temp~=0 % constraint violated at least once
        counter=1;
        for j=1:m
            if flag0(j)==1
                h1(counter)=h(j);
                counter=counter+1;
            end
        end
        h1=h1(1:(counter-1)); % # of active constraint: counter-1
        EP=gradproj(f,h1,SP,alpha);
        x(:,i+1)=EP;
        SP=EP;
    end
    if i>1
        error(i+1)=abs((y(i)-y(i-1))/y(i));
    end
    i=i+1;
end
```

```
%% Code 49: when d is known, compute b and t and Weight
clear;

%model related constants
stress_allow=241e6;
delta_allow=5e-3;
D=0.75; %transverse beam spacing
W_fl=6; %fuselage width
k=0.05;  %defection coefficient
E=69e9; %Young's modulus
u=2200; %mean value of plate load
var=400;%variance of plate load
c=4.2649;%norminv(1e-5)
```

```
x2=0.462;

x1=((c*var+u)*((3*x2*D^2))/(stress_allow*36))^(1/3)
x3=((c*var+u)*k*x2^4/(delta_allow*E))^(1/3)
x3=max(x3,0.003);
W1=3*x1^2*D*W_fl/x2;
W2=W_fl*D*x3;
W=W1+W2



%% Code 50: Floor Grid: compute beta value of Limit-state function at sample
points
clear;
syms g g1 g2 q t1 t2 b d t;
%model related constants
stress_allow=241e6;
delta_allow=5e-3;
D=0.75; %transverse beam spacing
W_fl=6; %fuselage width
k=1e-10;  %defection coefficient
E=69e9; %Young's modulus of Aluminum
h=0.0254; %track height
u_q=2200; var_q=400;%mean and variance of plate load
u_t1=0.003; var_t1=0.0003; %web of I beam
u_t2=0.003; var_t2=0.0003; %flange of I beam
c=4.2649; %norminv(1e-5)
rho=0.8;
% q=2200;t1=0.003;t2=0.003;b=0.004;d=0.3;t=0.012;

b0=[0.004:0.002:0.01];
d0=[0.3:0.15:0.75];
t0=[0.012:0.003:0.021];
beta0=zeros(4,4,4);
I=zeros(4,4,4);

g1=stress_allow-q*d*D^2*h/(16*(t1*(0.0254-2*t2)^3/12+b*t2^3/6+b*t2*(0.0254-
t2)^2/2));
g2=delta_allow-k*q*d^4/t^3;
g=g1;

% partial diff. of g wrt three variables x1 x2 x3
p1=diff(g,q);
p2=diff(g,t1);
p3=diff(g,t2);
corr=2*(p1*p2*var_q*var_t1*0+p1*p3*var_q*var_t2*0+p2*p3*var_t1*var_t2*rho);
den=((p1*var_q)^2+(p2*var_t1)^2+(p3*var_t2)^2+corr)^0.5;
den0=((p1*var_q)^2+(p2*var_t1)^2+(p3*var_t2)^2)^0.5;
q=u_q;
t1=u_t1;
t2=u_t2;

i_max=15;
```

```
for i1=1:4
    for i2=1:4
        for i3=1:4
            b=b0(i1); d=d0(i2); t=t0(i3);
            cosine=zeros(i_max,3);
            beta=zeros(i_max,1);
            X=zeros(i_max,3);
            U=zeros(i_max,3);
            temp_g=zeros(i_max,1);
            temp_p=zeros(i_max,3);

            i=1;
            beta_pre=0;
            temp1=eval(g); temp2=eval(den);
            beta(i)=eval(g/den);
            X(i,:)=[u_q u_t1 u_t2];
            U(i,:)=[0 0 0];
            while [abs((beta(i)-beta_pre)/beta(i))>0.0001 i<=i_max]
                i=i+1;
                x1 = X(i-1,1); x2 = X(i-1,2); x3 = X(i-1,3);
                q=x1; t1=x2; t2=x3;
                u1 = U(i-1,1); u2 = U(i-1,2); u3 = U(i-1,3);
                % compute the gradient and limit state value at each point.
                temp_p(i-1,1)=eval(p1);
                temp_p(i-1,2)=eval(p2);
                temp_p(i-1,3)=eval(p3);
                temp_g(i-1)=eval(g);
                % compute cosine values based on previous Xi
                cosine(i-1,1)=eval(-(p1*var_q)/den0);
                cosine(i-1,2)=eval(-(p2*var_t1)/den0);
                cosine(i-1,3)=eval(-(p3*var_t2)/den0);
                % compute new Ui
                U(i,1) = beta(i-1)*cosine(i-1,1);
                U(i,2) = beta(i-1)*cosine(i-1,2);
                U(i,3) = beta(i-1)*cosine(i-1,3);
                % compute new Xi
                X(i,1) = u_q+var_q*U(i,1);
                X(i,2) = u_t1+var_t1*U(i,2);
                X(i,3) = u_t2+var_t2*U(i,3);
                % compute new beta
                x1=X(i,1);x2=X(i,2);x3=X(i,3);
                q=x1; t1=x2; t2=x3;
                beta(i)=eval((g-
(p1*var_q*U(i,1)+p2*var_t1*U(i,2)+p3*var_t2*U(i,3)))/den);
                beta_pre=beta(i-1);
            end
            % beta evaluated:
            beta0(i1,i2,i3)=beta(i);
            I(i1,i2,i3)=i;
        end
    end
end
```

```
%% Code 51A: Floor Grid: Computing multiquadric interpolation coefficients
% plot the interpolation effect on 3D space
clear;
load beta_for_interpolation;
v1=zeros(4^3,1); %beta1 values
v2=zeros(4^3,1); %beta2 values
points=zeros(3,4^3);
A=zeros(64,64);
x=(0.004:0.002:0.01);
y=(0.3:0.15:0.75);
z=(0.012:0.003:0.021);
alpha=0.0001;

counter=1;
for i=1:4
    for j=1:4
        for k=1:4
            points(1,counter)=x(i);
            points(2,counter)=y(j);
            points(3,counter)=z(k);
            v1(counter)=beta1(i,j,k);
            v2(counter)=beta2(i,j,k);
            counter=counter+1;
        end
    end
end
distance=dist(points); %d is a 64x64 matrix
for i=1:63
    for j=i+1:64
        A(i,j)=(distance(i,j)^2+alpha^2)^0.5;
        A(j,i)=A(i,j);
    end
end
b1=A^(-1)*v1;% multiquadric interpolation coefficients
b2=A^(-1)*v2;% multiquadric interpolation coefficients

x=(0.004:0.0006:0.01);y=(0.3:0.045:0.75);z=0.012;
[x,y,z]=meshgrid(x,y,z);
f1=0;f2=0;
for i=1:64
    d1=((x-points(1,i)).^2+(y-points(2,i)).^2+(z-points(3,i)).^2).^0.5;
    f1=f1+b1(i)*(d1.^2+alpha.^2).^0.5;
    f2=f2+b2(i)*(d1^2+alpha^2)^0.5;
end
surf(x,y,f1);
```

```
%% Code 51B: Floor Grid: Computing multiquadric interpolation coefficients
% plot the interpolation effect on 4D space
clear;
load beta_for_interpolation;
v1=zeros(4^3,1); %beta1 values
```

```
v2=zeros(4^3,1); %beta2 values
points=zeros(3,4^3);
A=zeros(64,64);
x=(0.004:0.002:0.01);
y=(0.3:0.15:0.75);
z=(0.012:0.003:0.021);
alpha=0.0001;% determined by several test

counter=1;
for i=1:4
    for j=1:4
        for k=1:4
            points(1,counter)=x(i);
            points(2,counter)=y(j);
            points(3,counter)=z(k);
            v1(counter)=beta1(i,j,k);
            v2(counter)=beta2(i,j,k);
            counter=counter+1;
        end
    end
end
distance=dist(points); %d is a 64x64 matrix
for i=1:63
    for j=i+1:64
        A(i,j)=(distance(i,j)^2+alpha^2)^0.5;
        A(j,i)=A(i,j);
    end
end
b1=A^(-1)*v1;% multiquadric interpolation coefficients
b2=A^(-1)*v2;% multiquadric interpolation coefficients

x=(0.004:0.0006:0.01);y=(0.3:0.045:0.75);z=(0.012:0.0009:0.021);
[x,y,z]=meshgrid(x,y,z);
f1=0;f2=0;
for i=1:64
    d1=((x-points(1,i)).^2+(y-points(2,i)).^2+(z-points(3,i)).^2).^0.5;
    f1=f1+b1(i).*(d1.^2+alpha.^2).^0.5;
%     f2=f2+b2(i).*(d1.^2+alpha.^2).^0.5;
end
% surf(x,y,f1);
% slice(x,y,z,f1,[0.005 0.008],[0.45 0.6],[0.015 0.018]);
slice(x,y,z,f1,[],[],[0.012 0.015 0.018 0.021]);
xlabel('b axis');
ylabel('d axis');
zlabel('t axis');




%% Code 52: Floorgrid compute the analytical form of gradients at any given
% point based on multiquadric interpolation.
clear;
syms g1 g2 g3 x1 x2 x3;
index=1;% select which reliability index:stress=1;deflection=2
load b1b2;
if index==1
```

```
    b=b1;
elseif index==2
    b=b2;
end
points=zeros(3,4^3);
x=(0.004:0.002:0.01);
y=(0.3:0.15:0.75);
z=(0.012:0.003:0.021);
alpha=0.0001; % determined by several test

counter=1;
for i=1:4
    for j=1:4
        for k=1:4
            points(1,counter)=x(i);
            points(2,counter)=y(j);
            points(3,counter)=z(k);
            counter=counter+1;
        end
    end
end
f=0;
for i=1:64
    d1=((x1-points(1,i)).^2+(x2-points(2,i)).^2+(x3-points(3,i)).^2).^0.5;
    f=f+b(i).*(d1.^2+alpha.^2).^0.5;
end
g1=diff(f,x1);
g2=diff(f,x2);
g3=diff(f,x3);
% g1,g2,g3 are analytical form of gradient based on multiquadric
% interpolation, they are stored in data file b1b2.mat together with
% interpolation coefficients b1 and b2.

% in b1b2.mat file:
% g11,g12,g13 are for stress limit-state function;
% g21,g22,g23 are for deflection limit-state function;
% b1,b2 are interpolation coefficients for stress and deflection
% limit-state functions




% Compute next point by Gradient Projection Method, with 3 variables only
% This function computes one step only
% This function is for optimization problem with reliability index
% constraints that do not have a analytical form.
function y = gradproja(f,h,index,SP,step)
% y: next point, column vector;
% f: objective function of x1,x2,x3;
% h: equality constraints, not including beta constraints, column vector;
% index: specify which beta constraint is active: 0:none; 1:beta1; 2:beta2;
3:both
% SP: starting point, column vector;
% step: step size;
global b1 b2 g11 g12 g13 g21 g22 g23;
syms ph % partial derivatives of h
syms x1 x2 x3 % design variables
[n,m]=size(h); % constraint number n
c=4.2649;
% ph=[diff(h,x1) diff(h,x2) diff(h,x3)];

if index==0 %no beta constraint
    ph=[];
elseif index==1 % beta1 stress constraint
    ph=beta_gradient(1,SP);
elseif index==2 % beta2 deflection constraint
    ph=beta_gradient(2,SP);
elseif index==3 % both beta constraints
    ph=[beta_gradient(1,SP);beta_gradient(2,SP)];
end

pf=[diff(f,x1) diff(f,x2) diff(f,x3)];
ph0=[diff(h,x1) diff(h,x2) diff(h,x3)];%all non-beta constraints
x1=SP(1);x2=SP(2);x3=SP(3);
% ph1=eval(ph);
pf1=eval(pf);
temp1=eval(ph0);
ph1=[ph;temp1];
% projection matrix P:
P=eye(3)-ph1'*(ph1*ph1')^(-1)*ph1;
% EP: end point
EP=SP-step*P*pf1';
for i=1:2
    SP=EP;
    x1=SP(1);x2=SP(2);x3=SP(3);
    % ph1=eval(ph);
    if index==0 %no beta constraint
        ph=[]; %numerical value of beta constraint gradients
        h1=[]; % beta-4.2649 values
    elseif index==1 % beta1 stress constraint
        ph=beta_gradient(1,SP);
        h1=beta_interp(1,SP)-c;
    elseif index==2 % beta2 deflection constraint
        ph=beta_gradient(2,SP);
        h1=beta_interp(2,SP)-c;
    elseif index==3 % both beta constraints
        ph=[beta_gradient(1,SP);beta_gradient(2,SP)];
        h1=[beta_interp(1,SP)-c; beta_interp(2,SP)-c];
    end
    temp1=eval(ph0);%evaluation of all non-beta constraints
    ph1=[ph;temp1]; %numerical value of gradient of h at the new point
    % h1=eval(h);
    temp=[];
    if n~=0
        temp=eval(h);
    end
    h1=[h1;temp]; %numerical value of h at the new point
    EP=SP-ph1'*(ph1*ph1')^(-1)*h1;
end
y=EP;
```

```
% This function is for floorgrid problem, evaluate the reliability index
% values at any given point by multiquadric interpolation.
function y=beta_interp(index,P)
%index: indicate which limit-state function: stress=1,deflection=2;
%P: interpolation point, row vector (x,y,z)
global b1 b2 g11 g12 g13 g21 g22 g23;
% load b1b2;
if index==1
    b=b1;
elseif index==2
    b=b2;
end
points=zeros(3,4^3);
x=(0.004:0.002:0.01);
y=(0.3:0.15:0.75);
z=(0.012:0.003:0.021);
alpha=0.0001; % determined by several test

counter=1;
for i=1:4
    for j=1:4
        for k=1:4
            points(1,counter)=x(i);
            points(2,counter)=y(j);
            points(3,counter)=z(k);
            counter=counter+1;
        end
    end
end

f=0;
for i=1:64
    d1=((P(1)-points(1,i)).^2+(P(2)-points(2,i)).^2+(P(3)-
points(3,i)).^2).^0.5;
    f=f+b(i).*(d1.^2+alpha.^2).^0.5;
%     f2=f2+b2(i).*(d1.^2+alpha.^2).^0.5;
end
y=f;
```

```
% This function is for floorgrid problem, given a point P it will return
% the gradient (row vector) at the point based on interpolation model
function y=beta_gradient(index,P)
%index: indicate which limit-state function: stress=1,deflection=2;
%P: interpolation point, column vector (x,y,z)
global b1 b2 g11 g12 g13 g21 g22 g23;
syms g1 g2 g3 x1 x2 x3;
% load b1b2;
if index==1
    g1=g11;g2=g12;g3=g13;
elseif index==2
```

```
    g1=g21;g2=g22;g3=g23;
end

x1=P(1);x2=P(2);x3=P(3);
y=eval([g1 g2 g3]);





%% Code 53: Floorgrid:Optimization under all inequal constraints based on
% multiquadric interpolation surrogate constraints
clear;
syms f h a1 a2 a3 a4 a5 a6 a7 a8 a9 ;
syms x1 x2 x3;
global b1 b2 g11 g12 g13 g21 g22 g23;
load b1b2;
%model related constants
stress_allow=241e6;
delta_allow=5e-3;
D=0.75; %transverse beam spacing
W_fl=6; %fuselage width
k=1e-10;  %defection coefficient
E=69e9; %Young's modulus of Aluminum
h=0.0254; %track height
u_q=2200; var_q=400;%mean and variance of plate load
u_t1=0.003; var_t1=0.0003; t1=u_t1; %web of I beam
u_t2=0.003; var_t2=0.0003; t2=u_t2; %flange of I beam
c=4.2649; %norminv(1e-5)
rho=0.8;% correlation coefficient
density_al=2.7e3; %density of aluminum
density_hc=20;    %density of honeycomb

%algorithm related constants
SP=[0.004 0.7 0.015].'; %starting point
% SP=[0.0074424 0.4544 0.01437].'; %starting point
% SP=[0.0076 0.4555 0.0142].'; %starting point
% SP=[0.02 0.4 0.03].'; %starting point
alpha=0.001; %step size
i_max=20;

%objective function and constraints definition
f=(t1*(0.0254-2*t2)+2*x1*t2)*D*W_fl/x2*density_al+W_fl*D*x3*density_hc;

h=[ x2-0.3;
    0.75-x2;
%     x1-0.003;
%     x3-0.01
    ];
pf=[diff(f,x1) diff(f,x2) diff(f,x3)];
ph=[diff(h,x1) diff(h,x2) diff(h,x3)];
[m,n]=size(h);

x=zeros(3,i_max);% each step point x, column vectors
xt=zeros(3,i_max);% testing point on negtive gradient direction, column
vectors
```

```
y=zeros(i_max,1);% objective function value
flag=zeros(m+2,i_max);% recording which constraint becomes active (+1);,
column vectors
error=zeros(i_max,1);% Y errors at each step
h1=[a1 a2 a3 a4 a5 a6 a7 a8 a9].'; % maximum constraint number:9
beta=zeros(i_max,2);

i=1;
error(i)=1;error(i+1)=1;
x(:,1)=SP;
while [error(i)>1e-2 i<i_max]
    beta(i,1)=beta_interp(1,SP);beta(i,2)=beta_interp(2,SP);
    x1=SP(1);x2=SP(2);x3=SP(3);
    y(i)=eval(f);
    temp=(eval(pf))';
    EP=SP-alpha*temp; % Gradient Descent method
    xt(:,i+1)=EP; % testing point on -gradient direction
    x1=EP(1);x2=EP(2);x3=EP(3);
    flag0=[beta_interp(1,EP)-c beta_interp(2,EP)-c eval(h).']; % all
constraint values
    flag0=sign(flag0);
    flag0=flag0-1;
    flag0=sign(abs(flag0));% +1 represent constraint violation, column vector
    flag(:,i)=flag0;
    temp=sum(flag0); % test if any constraint violated
    if temp==0 % no constraint violation
        xt(:,i)=EP;
        SP=EP;
        x(:,i+1)=SP;
    elseif temp~=0 % constraint violated at least once
        counter=1;
        for j=3:m+2 % search all violated non-beta constraints
            if flag0(j)==1
                h1(counter)=h(j-2);
                counter=counter+1;
            end
        end
        h1=h1(1:(counter-1)); % # of active non-beta constraint: counter-1
        temp=[flag0(1)==1 flag0(2)==1];
        if temp==[0 0]  index=0;
        elseif temp==[1 0] index=1;
        elseif temp==[0 1] index=2;
        elseif temp==[1 1] index=3;
        end
        EP=gradproja(f,h1,index,SP,alpha);
        h1=[a1 a2 a3 a4 a5 a6 a7 a8 a9].'; %reset h1
        x(:,i+1)=EP;
        SP=EP;
    end
    if i>1
        error(i+1)=abs((y(i)-y(i-1))/y(i));
    end
    i=i+1;
end
```

REFERENCES

- Abbas, A.E. 2009 “Multiattribute Utility Copulas” Operations Research, Vol.57, No.6. Nov—Dec 2009, pp.1367-1383

- Bailey, R.A. (2008). “Design of Comparative Experiments”, Cambridge University Press

- Bendsoe, M. and N. Kikuchi, “Generating Optimal Topologies in Structural Design Using a Homogenization Method” Computer Methods in Applied Mechanics and Engineering, 1988. 71: p. 197-224.

- Bonabeau, E., Dorigo, M., Theraulaz, G., “Swarm Intelligence: From Natural to Artificial Systems”, Oxford University Press, 1999

- Brujic,D., Ristic,M., Mattone,M., Maggiore,P., Poli,G.P.D., “CAD based shape optimization for gas turbine component design”, Struct. Multidisc. Optim. (2010) 41:647-659

- Choi, S.K., “Reliability-based Structural Design”, Springer, 2006

- Choi,S.K. 2006 “Estimation of Structural reliability for Gaussian Random Fields” Structure and Infrastructure Engineering, Vol.2 Nos.3-4

- Chowdhury, M., Alouani, A., Hossain, F., “Comparison of Ordinary Kriging and Artificial Neural Network for Spatial Mapping of Arsenic Contamination of Groundwater”, Stoch Environ Res Risk Assess (2010) 24: 1-7

- Dennis G. Zill, “Advanced Engineering Mathematics”, Jones & Bartlett Publishers; 4 edition (December 21, 2009)

- Desktop Aeronautics, Inc., copy right 1997-2006, “Aircraft Design: Synthesis and Analysis”, CD version.

- Doucet,A., N de Freitas and N Gordon, “Sequential Monte Carlo Methods in Practice”, Springer, 2001. ISBN 978-0387951461

- Embrechts,P., Lindskog,F., McNeil,A., “Modelling Dependence with Copulas and Applications to Risk Management”, http://www.risklab.ch/ftp/papers/DependenceWithCopulas.pdf

- Evans,A.G., Hutchinson,J.W., Fleck,N.A., Ashby,M.F., Wadley,HNG, “The Topological Design of Multifunctional Cellular Metals”, PROGRESS IN MATERIALS SCIENCE 46 (3-4): 309-327 2001

- Farshi,B., Jahed,H., Mehrabian,A., “Optimum Design of Inhomogeneious Non-uniform Rotating Discs”, Computers and Structures 82 (2004) 773-779

- Frees, E.W., Valdez,E.A.(1998) “Understanding relationships using copulas” North Am. Act. J., 2(1),1-25

- Genest, C. (1987) “Frank’s family of bivariate distributions” Biometrika, 74(3), 549-555
- Genest, C. and Favre,A.C. (2007) “Everything you always wanted to know about copula modeling but were afraid to ask” J. Hydrologic Engrg
- Genest, C. and MacKay,J (1986) “The joy of Copulas: Bivariate distributions with uniform marginals”
- Genest, C. and Rivest, L.P. (1993) “Statistical inference procedures for bivariate Archimedean copulas” J.Am.Stat.Assoc., 88(3), 1034-1043
- Ghanem,R.G.; Spanos,P.D., ” Stochastic Finite Elements: A Spectral Approach”, Dover Publications, 2003
- Hohenbichler M, Gollwitzer S, Kruse W, Rackwitz R. “New light on first-and second-order reliability methods”. Structural Safety 1987;4:267–84.
- Hohenbichler R, Rackwitz R. “Improvement of second-order reliability estimates by importance sampling”, J Eng. Mech ASCE 1988;114(12):2195–9.
- Isaaks, E. H., and Srivastava, R. M. (1989), An Introduction to Applied Geostatistics, Oxford University Press, New York
- Jonathan F. Bard, “Practical Bilevel Optimization Algorithms and Applications”, Kluwer Academic Publishers, 2010
- Kim S.R., Park J.Y., Lee W.G., Yu J.S., Han S.Y., “Reliability-Based Topology Optimization Based on Evolutionary Structural Optimization”, World Academy of Science, Engineering and Technology 32, 2007
- Kim,H., Garcia,M.J., Querin,O.M., Steven,G.P., Xie,Y.M., “Introduction of Fixed Grid in Evolutionary Structural Optimisation”, Engineering Computations 17 (4): 427-439 2000
- Kodiyalam,S., Kumar,V., Finnigan,P.M., “Constructive Solid Geometry Approach to 3-dimensional Structural Shape Optimization”, AIAA JOURNAL 30 (5): 1408-1415 MAY 1992
- Kuczera,R.,Mourelatos,Z.P.,Nikolaidis,E., 2010 “System RBDO with Correlated Variables using Probabilistic Re-Analysis and Local Matamodels” Draft unbublished
- Kumar,S., Pippy,R.J., Acar,E., Kim,N.H., Haftka,R.T., “Approximate Probabilistic Optimization Using Exact-capacity-approximate-response-distribution (ECARD)”, Structural and Multidisciplinary Optimization 38 (6): 613-626 JUL 2009
- Laslett, G.M., “Kriging and Splines: An Empirical Comparison of Their Predictive Performance in Some Applications”, Journal of the American Statistical Association, Vol.89, No.426 (Jun.,1994), pp.391-400
- Lemaire, M., “Structural Reliability”, Wiley-ISTE, 2009

- Li,G., Wang,H., Aryasomayajula,S.R., Grandhi,R.V., “Two-level Optimization of Airframe Structures Using Response Surface Approximation”, Structural and Multidisciplinary Optimization 20 (2): 116-124 OCT 2000

- Liu,B., Haftka,R.T., Akgun,M.A., “Two-level Composite Wing Structural Optimization Using Response Surfaces”, Structural and Multidisciplinary Optimization 20 (2): 87-96 OCT 2000

- Liu,J.S., Parks,G.T., Clarkson,P.J., “Optimization of Turbine Disk Profiles by Metamorphic Development”, JOURNAL OF MECHANICAL DESIGN 124 (2): 192-200 JUN 2002

- Matias, J.M., Vaamonde, A., Taboada, J., Gonzalez-Manteiga, W., “Comparison of Kriging and Neural Networks With Application to the Exploitation of a Slate Mine”, Mathematical Geology, Vol.36, No.4, May 2004

- Nelsen,R.B. “An Introduction to copulas”. New York: Springer, 1999

- Oksuz,O., Akmandor,I.S., “Multi-Objective Aerodynamic Optimization of Axial Turbine Blades Using a Novel Multilevel Genetic Algorithm”, JOURNAL OF TURBOMACHINERY-TRANSACTIONS OF THE ASME 132 (4): - OCT 2010

- Panos Y. Papalambros; Douglass J. Wilde, “Principles of Optimal Design”, Cambridge University Press, 2000

- Papaefthymiou, G. 2009 “Using Copulas for Modeling Stochastic Dependence in Power System Uncertainty Analysis” IEEE Transaction on Power Systems, Vol 24-1

- Papaefthymiou,G., Kurowicka,D., “Using Copulas for Modeling Stochastic Dependence in Power System Uncertainty Analysis”, IEEE Transactions on Power Systems Feb2009, Vol. 24 Issue 1, p40-49 10p

- Papoulis, A.; Pillai,S.U., “Probability, Random Variables and Stochastic Processes”, McGraw Hill Higher Education, 4th Ed., 2002

- Paredis,C. 2009 “ME6105 course material”, Georgia Institute of Technology

- Parkinson DB. First-order reliability analysis employing translation systems. Eng Struct 1978;1:31–40.

- Rackwitz, R., (2001), “Reliability Analysis – A Review And Some Perspectives”, Structural Safety 23 (2001) 365-395

- Rao,A.R., Scanlan,J.P., Keane,A.J., “Applying Multiobjective Cost and Weight Optimization to the Initial Design of Turbine Disks”, Journal of Mechanical Design, December 2007, Vol.129 / 1303-1310

- Rao,J.R.J.; Badhrinath,K.; Pakala,R.; Mistree,F., “A Study of Optimal Design Under Conflict Using Models of Multi-Player Games”, Engineering Optimization 28 (1-2): 63-94 1997

- Roy,A., Hinduja,A., Teti,R., “Recent Advances in Engineering Design Optimisation: Challenges and Future Trends”, CIRP ANNALS-MANUFACTURING TECHNOLOGY 57 (2): 697-715 2008

- Saxena,A., Ananthasuresh,G.K., “On an Optimal Property of Compliant Topologies”, Structural and Multidisciplinary Optimization 19 (1): 36-49 MAR 2000

- Sigmund,O., “Topology Optimization: A Tool for the Tailoring of Structures and Materials”, Phil. Trans. R. Soc. Lond. A (2000) 358, 211-227

- Simaan, M. and Cruz, J.B., Jr., “On the Stackelberg Strategy in Nonzero-Sum Games”, Journal of Optimization Theory and Applications, Vol. 11, No. 5, May 1973, pp. 533-555

- Stephan Dempe, “Foundations of Bilevel Programming”, Kluwer Academic Publishers, 2010

- Stephen P. Timoshenko; J.N.Goodier, ”Theory of Elasticity”, McGraw Hill Higher Education; 3rd edition (October 1, 1970)

- Tvedt L. “Distribution of quadratic forms in normal space-application to structural reliability”. J Eng Mech 1990; 116(6):1183–97.

- Tvedt, L., “Two Second-order Approximation to the Failure Probability”, Section on Structural Reliability, A/S Vertas Research, Hovik, Norway, 1984

- Watenberg, D., Uchrin, C., Coogan, P., “Estimating Exposure Using Kriging: A Simulation Study”, Environmental Health Perspectives, Vol.94, pp.75-82, 1991

- Yang,R.J., Chuang,C.H., “Optimal Topology Design Using Linear Programming”, Computers & Structures Vol.52, No.2, pp.265-275, 1994

- Zimmerman, D., Pavlik, C., Ruggles, A., Armstrong, M.P., “An Experimental Comparison of Ordinary and Universal Kriging and Inverse Distance Weighting”, Mathematical Geology, Vol.31, No.4, 1999