Reliability-Based Structural Design Optimization for Nonlinear Structures in OpenSees

by

Hong Liang

B.Eng., Tongji University, China, 1993
M.Eng., Tongji University, China, 1996

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENT FOR THE DEGREE OF
MASTER OF APPLIED SCIENCE
in
THE FACULTY OF GRADUATE STUDIES
Civil Engineering

THE UNIVERSITY OF BRITISH COLUMBIA

May 2005

Abstract

The aspiration of this thesis is to provide a tool for engineers in making rational decisions based on the balance between cost and safety. This objective is accomplished by merging the optimization and reliability analyses with sophisticated finite element models that predict structural response. In particular, two state-of-the-art reliability-based design optimization approaches are implemented in OpenSees, a modern and comprehensive finite element software that has recently been extended with reliability and response sensitivity analysis capabilities. These new implementations enable reliability-based design optimization for comprehensive real-world structures that exhibit nonlinear behaviour.

This thesis considers the problem of minimizing the initial cost plus the expected cost of failure subject to reliability and structural constraints. This involves reliability terms in both objective and constraint functions. In the two implemented approaches, the reliability analysis and the optimization evaluation are decoupled, although they are not bi-level approaches, thus allowing flexibility in the choice of the optimization algorithm and the reliability method. Both solution approaches employ the same reformulation of the optimization problem into a deterministic optimization problem. The decoupled sequential approach using the method of outer approximation (DSA-MOOA) applies a semi-infinite optimization algorithm to solve this deterministic optimization problem. An important feature of the DSA-MOOA approach is that a convergence proof exists in the first-order approximation. The simplified decoupled sequential approach (DSA-S) utilizes an inequality constrained optimization algorithm to solve the deterministic optimization problem. The DSA-S approach is demonstrated to result in a consistent design, which lacks the convergence proof but requires less computational time than the DSA-MOOA approach.

The gradients of the finite element response with respect to model parameters are needed in reliability-based design optimization. These gradients are obtained using the

direct differentiation method, which entails the derivation and implementation of analytical derivatives of the finite element response. The potential negative effect of response gradient discontinuities due to sudden yielding events is stressed in the thesis. The problem is remedied through the use of the smooth material model and a section discretization scheme. Object-oriented programming is utilized when extending optimization and sensitivity capabilities to OpenSees. The superior extensibility and maintainability features of this approach are emphasized.

A numerical example involving a nonlinear finite element analysis of a three-bay, six-storey building is presented in the thesis to demonstrate new implementations in OpenSees. Three cases are studied: a linear pushover analysis using *elasticBeam* elements, a nonlinear pushover analysis using *beamWithHinges* elements, and a nonlinear pushover analysis using *dispBeamColumn* elements with *fibre* sections. This thesis also touches on practical experiences by comparing two implemented approaches, two gradient computation methods, and linear and nonlinear analyses. The experience of speeding up the convergence procedure by removing inactive constraints and scaling the involved functions is also discussed.

Contents

List of Tables vii

List of Figures viii

Conventions and Symbols ix

Acknowledgements x

1 Introduction 1

1.1 Reliability-Based Optimization Problems 2

1.2 Solution Algorithms 6

1.3 Thesis Organization 17

2 Finite Element Reliability Analysis 20

2.1 First-Order and Second-Order Reliability Methods 22

2.2 Monte Carlo and Importance Sampling 25

2.3 Gradient of the Failure Probability 26

3 Optimization Theory and Algorithms 28

3.1 Inequality Constrained Optimization Problem 28

3.1.1 First-Order Optimality Conditions 28

3.1.2 The Polak-He Algorithm 32

3.2 Semi-infinite Optimization Problem 34

3.2.1 First-Order Optimality Conditions 35

3.2.2 Method of Outer Approximation Algorithm 36

4 The OpenSees Software 40

4.1 Nonlinear Finite Element Analysis 41

4.2 Reliability Analysis 44

5 Response Sensitivity for Nonlinear Structures 48

5.1 Finite Difference Method 48

5.2 Direct Differentiation Method 49

5.3 Object-Oriented Implementation in OpenSees 51

5.4 Continuity of Response Gradients 53

5.4.1 Smooth Material Model 54

5.4.2 Section Discretization Scheme 55

6 Implementation of Reliability-Based Design Optimization 57

6.1 Problem Reformulation 57

6.2 DSA-MOOA Approach 61

6.2.1 B1 – Inner Approximation 64

6.2.2 B2 – Constraints Expansion 66

6.2.3 B3 – Outer Approximation 68

6.3 DSA-S Approach 71

6.4 Object-Oriented Implementation in OpenSees 74

7 Numerical Examples and Case Studies 79

7.1 Six-Story Ductile Moment Resisting Frame 79

7.1.1 Case 1: Elastic Analysis using *elasticBeam* Element 81

7.1.2 Case 2: Nonlinear Analysis using *beamWithHinges* Element 88

7.1.3 Case 3: Nonlinear Analysis using *dispBeamColumn* Element and *Fiber* Section 95

7.2 Practical Experience from Case Studies 103

7.2.1 Comparison of Two Optimization Approaches 103

7.2.2 Comparison of Two Gradient Computation Methods 104

7.2.3 Comparison of Linear and Nonlinear Analyses 105

7.2.4 Active and Inactive Constraints 107

7.2.5 Acceleration of Convergence Procedure by Proper Scaling 108

8 Conclusions **110**

8.1 Summary of Major Findings 110

8.2 Further Studies 113

Bibliography **115**

Appendix A: Detailed Software Implementation **120**

A.1 Calling Fortran Routines from C++ 120

A.2 Building LSSOL.LIB 122

A.3 Extending OpenSees with RBDO Capacity 123

Appendix B: User's Guide to Optimization Analysis **129**

B.1 RBDO Modeling 129

B.2 Analysis Tools 131

B.3 Analysis Extension and Results 132

List of Tables

Table 7.1 Vertical loads and lateral loads 81
Table 7.2 Definition and initial values of design variables for Cases 1 and 2 83
Table 7.3 Statistics of random variables in Case 1 83
Table 7.4a Results from RBDO analysis for Case 1 85
Table 7.4b Results from RBDO analysis for Case 1 (continued) 85
Table 7.5 Comparison of computational time for Case 1 87
Table 7.6 Statistics of random variables for Case 2 89
Table 7.7a Results from RBDO analysis for Case 2 92
Table 7.7b Results from RBDO analysis for Case 2 (continued) 92
Table 7.8 Comparison of computational time for Case 2 95
Table 7.9 Definition and initial values of design variables for Case 3 97
Table 7.10 Statistics of random variables for Case 3 98
Table 7.11a Results from RBDO analysis for Case 3 100
Table 7.11b Results from RBDO analysis for Case 3 (continued) 100
Table 7.11c Results from RBDO analysis for Case 3 (continued) 100
Table 7.12 Comparison of computational time for Case 3 102
Table 7.13 Comparison of linear and nonlinear cases 106
Table A.1a New and modified classes for extending RBDO 126
Table A.1b New and modified classes for extending RBDO (continued) 127
Table A.1c New and modified classes for extending RBDO (continued) 128

List of Figures

Figure 2.1 MPP searching algorithm in finite element reliability analysis 23

Figure 3.1 Local and global optimal points 29

Figure 3.2 Constrained optimization problems 30

Figure 4.1 Principal OpenSees objects 41

Figure 4.2 Element, section and material relationship 42

Figure 4.3 Fibre-section examples 43

Figure 4.4 Elastic-perfectly plastic material (F_yP =0) and *Steel01* material 44

Figure 4.5 Software framework for reliability analysis in OpenSees 45

Figure 5.1 The framework of response sensitivity in OpenSees 51

Figure 5.2 Bi-linear steel material model smoothed with circular segment 54

Figure 6.1 Flow chart of DSA-MOOA approach 62

Figure 6.2 μ_l solutions for inner approximation using the Polak-He algorithm 65

Figure 6.3 Reliability constraints set $\psi_l(\bar{\mathbf{x}})$ 67

Figure 6.4 Flow chart of DSA-S approach 72

Figure 6.5 New objects for optimization analysis in *ReliabilityDomain* 75

Figure 6.6 Software framework for optimization analysis in OpenSees 76

Figure 6.7 Interaction between optimization, reliability, and finite element module 77

Figure 7.1 Ductile moment-resisting frame model 80

Figure 7.2 Equal displacement principle 82

Figure 7.3 Structural responses for Case 1 86

Figure 7.4 *beamWithHinges* element 88

Figure 7.5 Evolution of the total expected cost for objective functions for Case 2 93

Figure 7.6 Structural responses for Case 2 94

Figure 7.7 Typical fibre sections for columns and beams 96

Figure 7.8 Structural responses for Case 3 101

Conventions and Symbols

$\varphi(\cdot)$ denotes the standard normal probability density function.

$\Phi(\cdot)$ denotes the standard normal cumulative distribution function.

Superscript ^{T} denotes the transpose of a matrix.

Vectors and matrices denotes by roman letters are in bold.

x_j denotes the j-th component of a vector $\mathbf{x}$.

$\langle \mathbf{x}, \mathbf{y} \rangle$ denotes the inner product of vectors $\mathbf{x}$ and $\mathbf{y}$.

$\|\mathbf{x}\|$ denotes the norm of vector $\mathbf{x}$ and is defined by $\|\mathbf{x}\| = \langle \mathbf{x}, \mathbf{x} \rangle^{1/2}$.

$a_+ = \max\{0, a\}$ for any a.

Acknowledgements

I wish to express my deep appreciation to my supervisor Dr. Terje Haukaas for his willingness to guide me through the challenging path towards a master degree. His approach to research and teaching will always be a source of inspiration. He leads me into the world of reliability and sensitivity analysis. His serious work manner and optimistic life attitude will keep inspiring me on my personal and professional development.

I am grateful to Dr. Johannes Ovrelid Royset for his patient and detailed explanation of the optimization theory. His kind help makes my research work smooth and possible.

I would like to thank Dr. Sigi Stiemer for being willing to serve in my thesis committee. His encouragement, and constructive, criticism are much appreciated. His course gives me a lot of helps understanding the nonlinear and plastic theory of structures.

I am thankful to my parents and brother for their encouragement and support. In particular, I am indebted to Ling Zhu, who takes care of me as a wife, discusses my research as a classmate and continuously gives me strength and confidence as a soul mate.

To Ling and my parents

Chapter 1 Introduction

The primary objective of structural engineering is a structural design that represents an optimal balance between cost and safety. Traditionally, this problem has been addressed through experience, trial and error, and ad-hoc comparisons of different designs. In these approaches, a comprehensive exploration of design alternatives is not performed, and uncertainties are not accounted for in a refined and consistent manner. In recent decades, the optimization theory has been developed to find the optimal design in the mathematical framework of minimizing an objective function subject to constraints. The intention of this thesis is to implement, demonstrate, and improve state-of-the-art algorithms for finding safe and optimal designs, as well as apply these implementations to real-world structures exhibiting nonlinear behaviour.

A number of reliability-based design optimization (RBDO) approaches has been developed, such as response surface methods as well as gradient-free and gradient-based algorithms. These may be utilized to minimize the total volume or the total expected cost of the structure subject to structural reliability constraints, to maximize the structural safety subject to a given structure cost, or simply to achieve a target structural reliability. The approaches adopted in this thesis are to employ sophisticated structural models, as well as advanced reliability methods, to account for uncertainty.

In simulations of structural behaviour, unavoidable uncertainties are present in the material, geometry, and load parameters, as well as in the model itself and the analysis procedures. These uncertainties can be significant in determining the performance of a structure and must be accounted for to ensure safe and reliable structures. Reliability methods have been devised to estimate the probability of response events for random structural properties and loads (Ditlevsen & Madsen, 1996). Reliability methods such as first-order reliability methods and importance sampling are employed in this thesis to evaluate structural reliability in RBDO analyses.

The finite element method is currently the leading-edge approach for numerical simulations of structural behaviour. It is of considerable interest to incorporate sophisticated finite element models into the RBDO analysis. Furthermore, this thesis addresses the need for implementation of state-of-the-art optimization techniques in a finite element code that is in widespread use. Flexible software architecture is required to accommodate the extensive interaction between optimization, reliability, and finite element modules of the software. OpenSees - open system for earthquake engineering simulations - (McKenna et al., 2004), is ideal for this purpose. This is an object-oriented, open-source software that is freely available from *http://opensees.berkeley.edu*. It serves as the computational platform for the prediction of structural and geotechnical responses for the Pacific Earthquake Engineering Research Center (PEER). Recently, OpenSees was extended with reliability and response sensitivity analysis capabilities (Haukaas & Der Kiureghian, 2004). This allows reliability analyses to be conducted in conjunction with static and dynamic inelastic finite element analyses, with random material, geometry, and load parameters.

A novelty of this thesis is the use of the object-oriented programming approach to develop a library of software components (tools) for optimization analysis. This approach provides a software framework that is easily extended and maintained. Indeed, the decoupling optimization approaches considered in this thesis take advantage of the object-oriented approach, in which solution algorithms for reliability and optimization problems are readily substituted by future developed solution algorithms.

1.1 Reliability-Based Optimization Problems

A specific structural design is characterized by particular values of *design variables*. By definition, the values of the design variables are assumed to be at the discretion of the designer. They typically represent geometrical dimensions and material strengths, and are collected in the vector $\mathbf{x}$. The random variables of the structural problem, such as material, geometry, and load parameters, are collected in a separate vector $\mathbf{v}$. It is noted

that design variables may represent distribution parameters of the random variables. For instance, the designer may wish to optimize the dimensions of a girder cross-section. However, the dimensions are uncertain due to imperfect workmanship, etc. Hence, only the mean of the structural dimensions is at the discretion of the designer. Similarly, in some cases the designer may have control over the dispersion of the probability distribution of a random variable through tolerance specifications to the manufacturer of the component.

RBDO can be classified into three broad problem categories: (1) minimization of the structural cost or volume subject to constraints on structural properties and/or reliability; (2) maximization of structural reliability subject to constraints on cost and/or structural properties; and (3) minimization of the discrepancy between structural reliability and specified target reliability subject to structural constraints and possible cost constraints.

In the first category of problems, the cost may include the expected cost of failure in addition to the initial cost. The total cost is written as $c_t = c_0 + c_f p_f$, where c_0 is the initial cost of the design, c_f is the present cost of future failure,[1] and p_f is the probability of failure. The reliability of the structure is defined as $1 - p_f$. The failure probability is obtained from either a component reliability problem or a system reliability problem, as described in Chapter 2. It should be noted that the cost and probability of failure generally depend on design variables: $c_t = c_t(\mathbf{x}) = c_0(\mathbf{x}) + c_f(\mathbf{x}) p_f(\mathbf{x})$. Several problems are identified in this category:

$$\mathbf{x}^* = \arg\min \left\{ \; c_0(\mathbf{x}) + c_f(\mathbf{x}) p_f(\mathbf{x}) \;\middle|\; \mathbf{f}(\mathbf{x}) \le \mathbf{0} \; \right\} \tag{1.1}$$

$$\mathbf{x}^* = \arg\min \left\{ \; c_0(\mathbf{x}) + c_f(\mathbf{x}) p_f(\mathbf{x}) \;\middle|\; \mathbf{f}(\mathbf{x}) \le \mathbf{0}, \;\; p_f(\mathbf{x}) \le \hat{p}_f \; \right\} \tag{1.2}$$

---

[1] c_f is obtained for the case of continuous compounding by Sexsmith (1983): $c_f = c_{f,future} \cdot u/(i+u)$, where $c_{f,future}$ is the future cost, i is the real interest rate (excluding inflation), and u is the rate of occurrence of the Poisson process that describes the hazard.

$$\mathbf{x}^* = \arg\min\left\{ c_0(\mathbf{x}) + \sum_{k=1}^{K} c_k(\mathbf{x}) p_k(\mathbf{x}) \;\middle|\; \mathbf{f}(\mathbf{x}) \le \mathbf{0},\;\; p_k(\mathbf{x}) \le \hat{p}_k \right\} \qquad (1.3)$$

where $\mathbf{x}^*$ is the optimal design, $\mathbf{f}$ is the vector of structural constraints, $\hat{p}_{()}$ denotes the target probability, K denotes the number of failure modes, and c_k and p_k are the cost of failure and the probability of failure of the k^{th} failure mode, respectively. An example of a structural constraint is $f_j = d - d_0$, where d is a structural dimension and d_0 is the prescribed upper bound of d.

The view adopted in this thesis is that the problem in Eq. (1.1) is the fundamental problem in RBDO. Eq. (1.1) seeks to minimize the total expected cost, which implicitly includes structural reliability, in light of various constraints to ensure an esthetical and functional design. This strategy finds its analogy in the field of decision analysis (Benjamin & Cornell, 1970), where rational decisions are made based on the expected utility of the decision alternatives. Consequently, the optimal balance between cost and safety is achieved. Theoretically, no constraint on the reliability is needed, provided that the cost of failure is appropriately defined. However, defining the appropriate cost of failure is a key problem in modern RBDO. The cost of failure potentially includes the value of human life and other intangible costs. For this reason it is useful to introduce Eq. (1.2), which includes a reliability constraint to ensure that the design conforms to minimum safety requirements.

Eq. (1.3) addresses problems where multiple failure modes are possible, each with individual failure costs. This is different from Eqs. (1.1) and (1.2), where multiple failure modes may be present in the system reliability $1 - p_f$, but where the failure cost is associated with the global system failure. A weakness of the formulation in Eq. (1.3) is the implicit assumption of independence between failure modes, which is rarely satisfied in practice.

Simplified versions of the problems in Eqs. (1.1) to (1.3) are obtained by minimizing only the initial design cost:

$$\mathbf{x}^* = \arg\min\{\ c_0(\mathbf{x}) \mid \mathbf{f}(\mathbf{x}) \le \mathbf{0}\ \} \tag{1.4}$$

$$\mathbf{x}^* = \arg\min\{\ c_0(\mathbf{x}) \mid \mathbf{f}(\mathbf{x}) \le \mathbf{0},\ p_f(\mathbf{x}) \le \hat{p}_f\ \} \tag{1.5}$$

These problems are frequently addressed in engineering practice because they avoid the need for assessing the failure cost. In fact, Eq. (1.4) denotes the well-known deterministic (non-RBDO) design optimization problem for which uncertainty is not accounted. Relative to Eq. (1.4), the problem in Eq. (1.5) introduces a safety constraint for which the reliability analysis is required. This is, conceptually, the simplest RBDO problem.

The second category of RBDO problems is identified as

$$\mathbf{x}^* = \arg\min\{\ p_f(\mathbf{x}) \mid \mathbf{f}(\mathbf{x}) \le \mathbf{0}\ \} \tag{1.6}$$

$$\mathbf{x}^* = \arg\min\{\ p_f(\mathbf{x}) \mid \mathbf{f}(\mathbf{x}) \le \mathbf{0},\ c_0(\mathbf{x}) \le \hat{c}\ \} \tag{1.7}$$

$$\mathbf{x}^* = \arg\min\{\ p_f(\mathbf{x}) \mid \mathbf{f}(\mathbf{x}) \le \mathbf{0},\ c_0(\mathbf{x}) + c_f(\mathbf{x}) p_f(\mathbf{x}) \le \hat{c}\ \} \tag{1.8}$$

where $\hat{c}$ is the prescribed upper bound of the cost. An extended set of problems is formulated by replacing $p_f(\mathbf{x})$ with $\max p_k(\mathbf{x})$; namely, the maximum failure probability over all failure modes. The problems in Eqs. (1.6) to (1.8) then turn into *min-max* type problems.

The problem in Eq. (1.6) seeks to maximize the reliability given structural constraints. While this is sometimes referred to as the *inverse reliability problem* in the literature, we reserve this term for a problem introduced below. In Eqs. (1.7) and (1.8) the initial cost and the total expected cost are introduced as constraints. Hence, these two equations are counterparts to Eqs. (1.5) and (1.2), respectively. However, although Eqs. (1.7) and (1.8) represent the "flipped" version of Eqs. (1.5) and (1.2), they are not equivalent problems. That is, the optimal design achieved by addressing Eqs. (1.8) and (1.2) is generally different.

The third category of RBDO problems contains what is referred to as inverse reliability problems (Der Kiureghian et al., 1994; Li & Foschi, 1998). Here, the discrepancy

between structural reliability and prescribed target reliability is minimized, subject to constraints:

$$\mathbf{x}^* = \arg\min\left\{ \left| p_f(\mathbf{x}) - \hat{p}_f \right| \;\middle|\; \mathbf{f}(\mathbf{x}) \le \mathbf{0} \right\} \tag{1.9}$$

$$\mathbf{x}^* = \arg\min\left\{ \left| p_f(\mathbf{x}) - \hat{p}_f \right| \;\middle|\; \mathbf{f}(\mathbf{x}) \le \mathbf{0},\ c_0(\mathbf{x}) \le \hat{c} \right\} \tag{1.10}$$

$$\mathbf{x}^* = \arg\min\left\{ \left| p_f(\mathbf{x}) - \hat{p}_f \right| \;\middle|\; \mathbf{f}(\mathbf{x}) \le \mathbf{0},\ c_0(\mathbf{x}) + c_f(\mathbf{x}) p_f(\mathbf{x}) \le \hat{c} \right\} \tag{1.11}$$

In addition, an extended set of problems is formulated by replacing $\left| p_f(\mathbf{x}) - \hat{p}_f \right|$ with $\max\left| p_k(\mathbf{x}) - \hat{p}_k \right|$, where $p_k(\mathbf{x})$ is the failure probability of failure mode *k*. Ideally, the value of the objective function of Eqs. (1.9) to (1.11) at the design point is zero. This would imply that target reliability is achieved and that constraints are satisfied. However, it may not be possible to achieve the reliability $1 - \hat{p}_f$ for given structural/cost constraints. In fact, Eqs. (1.9) to (1.11) are related to the problems in Eqs. (1.6) to (1.8), which seek maximization of reliability rather than convergence to a target reliability. For instance, Eq. (1.6) results in design variable values that minimize failure probability, while Eq. (1.9) potentially results in design variable values that provide a less safe design, but that complies with the prescribed reliability $1 - \hat{p}_f$.

In this thesis, the first category of RBDO problems is considered. This choice is founded on our belief that the principles of rational decision-making should form the basis for RBDO. Moreover, we address the problem for a single failure event in Eq. (1.2). Ideally, the problem in Eq. (1.1) should be addressed. However, the difficulties in obtaining the "true" cost of failure make this problem less practical. The problem in Eq. (1.3) is not considered in this thesis, because only one failure cost is assumed.

1.2 Solution Algorithms

In this thesis it is assumed that the problem in Eq. (1.2) is defined in terms of a finite element model. Specifically, the design variables **x** and the random variables **v** are

specified in terms of the input parameters of the finite element model. In this situation, a number of challenges are present when attempting to solve Eq. (1.2):

Challenge 1. The failure probability $p_f(\mathbf{x})$ must be computed using reliability methods that are coupled with the finite element analysis. This type of analysis is termed the finite element reliability analysis and may be challenging in itself. A number of reliability methods exists, all approximate. The choice of method influences the choice of optimization algorithm and its behaviour. The failure probability is a nonlinear function of $\mathbf{x}$, regardless of whether the limit-state function is linear. Moreover, the failure probability may not be continuously differentiable.

Challenge 2. The structural response may be nonlinear, which is the case under consideration in this thesis. The nonlinearities of the structural response and the failure probability cause the objective function and constraint functions to be nonlinear as well.

Challenge 3. The objective function, constraint functions, and the limit-state function are implicit functions expressed by structural responses from the finite element analysis.

Challenge 4. The most effective algorithms to solve Eq. (1.2) are gradient-based. That is, they require the gradient of the objective function and constraint functions with respect to $\mathbf{x}$ to be computed accurately and efficiently. Unless a reformulation technique is employed, the gradient of the failure probability and possibly the finite element response must be computed. The gradient computation may be both analytically problematic and computationally costly. Additionally, inaccuracies in the gradients lead to convergence problems in the optimization analysis.

Challenge 5. In this thesis, we typically consider problems including 10–100 design variables and 10–500 random variables. It is imperative that the solution

algorithms provide feasible computation times as the number of variables grows. This is known as the high-dimensional problem.

We conclude that candidate algorithms to address Eq. (1.2) must be assessed according to their efficiency, accuracy, and convergence robustness. In recent years, significant research efforts have been assigned to solve this RBDO problem. Consequently, a number of algorithms is available. In the following we briefly review the main approaches and comment on their performance relative to the challenges listed above, before arriving at the approaches adopted in this thesis.

A broad class of algorithms used to address Eq. (1.2) is termed gradient-free. Their key characteristic is that derivatives (gradients) of the objective function and constraint functions are not required in the analysis. Instead, the algorithms evaluate the objective and constraint functions at a number of trial points in the space of design variables. Several classes of gradient-free algorithms are available to solve RBDO problems. One example is genetic algorithms (Itoh & Liu, 1999; Thampan & Krishnamoorthy, 2001). Analogous to principles of natural genetics, these algorithms are evolutionary computation techniques. The key to genetic algorithms is the representation of a design as a set of binary numbers. A set of designs is termed a *population*. The first population is established randomly, through a user-defined number of candidate solutions. The objective of the procedure is to improve the population (the set of designs) in a manner similar to genetic operations in real nature. Three genetic operators (reproduction, crossover, and mutation) are used to create a new population (generation). During this process, the candidate solutions with the better designs are selected as parents to produce the next generation with improved design. The more genetic operations are present, the better solutions are achieved. Usually, the user stops the optimization at a predefined maximum number of operations.

In the following we address the advantages and disadvantages of generic algorithms based on their performance relative to the aforementioned challenges:

1. The failure probability is computed using any available finite element reliability methods.

2. Genetic algorithms are applicable to complex and nonlinear RBDO problems.
3. Genetic algorithms have the ability to couple with the finite element analysis.
4. Genetic algorithms avoid the convergence problems caused by inaccurate or discontinuous response gradients because of their gradient-free properties.
5. Genetic algorithms are computationally expensive and converge slowly since they require many more objective function evaluations than some alternative techniques. Their computational effort increases rapidly as the number of design variables increases. Therefore, genetic algorithms are not suitable for high-dimensional problems.

- The optimal solutions are not highly accurate because the representation of design variables as binary numbers does not allow high accuracy. Genetic algorithms have been suggested to narrow down the optimal solution region, followed by a traditional optimization method to identify a more precise optimum.
- Genetic algorithms are able to address problems with continuous and discrete design variables. An example of a discrete design variable is the cross-section type within a class of steel cross-sections. The problem with discrete design variables is not considered in this thesis.

The response surface method is an alternative to the gradient-free algorithms. The use of response surface methods is generally contemplated in problems where the objective function, and possibly the constraints, is not explicitly defined. This is indeed the case in Eq. (1.2). Even when cost functions are explicitly defined, a reliability analysis is required to obtain the probability of failure. In the response surface method, the value of the function in question is evaluated using a number of realizations of the design variable vector $\mathbf{x}$. An explicit algebraic continuously differentiable expression is then used to fit to these points in the space of design variables. The reformulated optimization problem is then solved using standard nonlinear gradient-based optimization algorithms. The gradients are readily found because response surfaces are simple algebraic expressions. An important advantage of the response surface method is that sensitivities of the

functions themselves are unnecessary. Hence, the response surface method may be termed *quasi gradient-free.*

As an example of a response surface method, Gasser and Schueller (1998) approximate the failure probability using an exponential function dependent on the design variables. This approximation makes the failure probability numerically continuous. Torczon and Trosset (1998) use an algebraic function to approximate the objective function in the optimization problem. The approximate functions are constructed from numerous evaluations of the objective function. Eldred et al. (2002) employ approximate functions for both the objective function and the limit-state function.

Response surface methods have the following advantages and disadvantages relative to the challenges listed above:

1. The failure probability is computed using any available finite element reliability method.
2. Response surface methods are numerically robust since the reformulated failure probabilities, objective functions, or limit-state functions are explicit and continuous and can be solved using standard nonlinear optimization algorithms. However, more sampling points are required for approximating the highly nonlinear functions in RBDO problems.
3. Response surface methods have the ability to couple with the finite element analysis.
4. Response surface methods can solve complex problems involving discontinuous functions in optimization since continuous functions are used to replace the original discontinuous functions.
5. The accuracy of the optimal solution depends on the accuracy of the approximation to the original problem. The efficiency of the optimization process relies on the computational time of constructing response surface, which is dependent on design variables. The numerical effort increases tremendously with

the increase in the number of design variables. Therefore, response surface methods are not suitable for high-dimensional problems.

- Response surface methods are likely to find a global minimizer rather than simply identifying a local minimizer.

Although gradient-free algorithms and response surface methods are viable alternatives, gradient-based algorithms are the most efficient in solving Eq. (1.2). In fact, traditional nonlinear optimization techniques are gradient-based. A variety of gradient-based algorithms is proposed in the literature. These algorithms are categorized into nested bi-level approaches, mono-level approaches, and decoupled sequential approaches. Enevoldsen and Sorensen (1994) solve Eq. (1.2) using the nested bi-level approach. This is the "brute force" approach, in which the standard nonlinear optimization algorithm is employed and the failure probability is evaluated each time the values of the objective function and of the reliability constraint are needed. (Hence the name *nested bi-level*.) A noteworthy feature of the work by Enevoldsen and Sorensen (1994) is that the first-order reliability method (FORM) is utilized to evaluate the failure probability. This is a reasonable approximation unless the limit-state function is strongly nonlinear. However, in nonlinear finite element reliability analyses, which are the case under consideration in this thesis, strong nonlinearities may be present. The nested bi-level approach requires the objective function and constraints to be continuous and to have first-order derivatives with respect to optimization variables. An advantage of using FORM analysis is that the gradient of the failure probability with respect to design variables is available, as shown in Chapter 2.

The nested bi-level approach has the following advantages and disadvantages relative to the challenges listed above:

1. Failure probability is practically limited to the FORM analysis.
2. Additional computational costs are required to determine the optimal design for highly nonlinear functions in RBDO problems.
3. The nested bi-level approach has the ability to couple with finite element analysis.

4. The gradient discontinuity problem may cause non-convergence or slow convergence.
5. The computation time is large since reliability analyses must be performed in each step of the optimization analysis. Therefore, the nested bi-level approach is not suitable for high-dimensional problems.

Madsen and Friis Hansen (1992) and Kuschel and Rackwitz (2000) develop the mono-level approach to improve the nested bi-level approach. Since the FORM analysis on the inner reliability level is actually an optimization problem itself (see Chapter 2), it is substituted by its first-order optimality conditions. In this manner, the reliability problem becomes an additional constraint in the RBDO problem. The reformulated optimization problem is then solved using standard nonlinear optimization algorithms in the augmented space of design variables. The mono-level approach has the following features:

1. The failure probability is computed using the FORM analysis. Extensions to second-order reliability methods are possible but are rather expensive.
2. Additional computational costs are required to determine the optimal design for the highly nonlinear functions in RBDO problems.
3. The second order derivative of the limit-state function is necessary even when using first-order optimization algorithms. This makes the RBDO for nonlinear structures impossible since the second order derivative of structural response with respect to model parameter is rarely available in the finite element analysis.
4. The gradient discontinuity problem may cause non-convergence or slow convergence.
5. The reformulated optimization has a greater number of design variables than the original one because of additional constraints from the reliability analysis. The reformulated problem may be too big to be solved for high-dimensional problems.

- Standard optimization algorithms without any links to an external code are used since the inner reliability level is eliminated.
- The explicit transformation between the original space and the standard normal space of random variables is required in the mono-level approach.
- The mono-level approach is only applicable for the component reliability problem and separable series systems, which do not have correlation between different failure modes.

A "direct" mono-level approach was developed by Chen et al. (1997) and generalized by Wang and Kodiyalam (2002) and Agarwal et al. (2003). In this approach design variables are defined as the mean values of some random variables. Furthermore, FORM reliability analysis is employed. Under the assumption of uncorrelated random variables, a direct relationship is established between the design variables and the approximation point in FORM analysis (this will later be termed *the most probable failure point*) for fixed target reliability. The reformulated optimization problem is a deterministic optimization problem not requiring a reliability analysis. It is solved using a standard nonlinear optimization solver. This direct mono-level approach is appealing for several reasons:

1. The failure probability evaluation is not required in the reformulated deterministic optimization analysis.
2. Additional computational costs are required to determine the optimal design for the highly nonlinear functions in RBDO problems.
3. Only first-order derivatives of the structural response with respect to design variables are required. They are available from the finite element analysis.
4. The gradient discontinuity problem may cause non-convergence or slow convergence.
5. The direct mono-level approach is efficient in the absence of a reliability analysis in the optimization analysis. The computational cost is approximately the same as

in the deterministic optimization analysis (without adding additional constraints). Therefore, this approach is applicable to high-dimensional problems.

- The transformation between the original space and the standard normal space of random variables does not have to be explicit.
- The direct mono-level approach can only deal with mutually independent random variables. A further study of correlated random variables is required.

Finally, we arrive at decoupled sequential approaches, some of which are considered to be state-of-the-art solution algorithms. One decoupling approach is developed by Du and Chen (2002) and Agarwal and Renaud (2004) to improve the nested bi-level approach. Reliability and optimization analyses are performed separately and sequentially. That is, a deterministic optimization is executed to obtain a new design without re-computing the reliability. Then, the failure probability is updated by performing a reliability analysis for the new design. The process is repeated until a consistent design is obtained. The term *decoupled* is used because the analyst may freely select different methods for the reliability and optimization problems. This is different from the mono-level approach where FORM reliability analysis is implicit. The decoupled bi-level approach has the following advantages and disadvantages:

1. The failure probability is computed using any available reliability methods since the reliability analysis is decoupled from the optimization analysis.
2. Additional computational effort is required to determine the optimal design for highly nonlinear functions in RBDO problems.
3. The decoupled bi-level approach has the ability to couple with finite element analysis.
4. The gradient discontinuity problem may cause non-convergence or slow convergence.

5. The decoupled approach is more efficient than the nested bi-level approach because the number of reliability evaluations is significantly reduced. Therefore, this approach is applicable to high-dimensional problems.

- It is easy to code and to combine the reliability analysis with any optimization software without having to reformulate the problem.
- The true local optimal solution cannot be guaranteed because the failure probability is always computed for a previous design.

Further improvements of the decoupled approach are developed by Kirjner-Neto et al. (1998), Der Kiureghian and Polak (1998), and Royset et al. (2001a). Most notable of these improvements is a decoupled sequential approach utilizing *the method of outer approximations* (Polak, 1997) to solve several RBDO problems. Royset et al. (2001b, 2002, & 2004a) further develop the methodology to solve Eq. (1.2). In this approach, Eq. (1.2) is reformulated as a semi-infinite optimization problem (Polak, 1997). This reformulated problem has proven to be identical to the original problem, when FORM analysis is used to compute the failure probability. Moreover, a heuristic scheme is implemented to improve the reliability estimate. The term *semi-infinite* comes from the fixed number of design variables and the infinite number of reliability constraints in the reformulated problem. This approach has the following advantages and disadvantages:

1. The failure probability is computed using any available reliability methods since the reliability analysis is completely decoupled from the optimization analysis.
2. Additional computational costs are required to determine the optimal design for the highly nonlinear functions in RBDO problems.
3. The decoupled sequential approach has the ability to couple with the finite element analysis.
4. The gradient discontinuity problem may cause non-convergence or slow convergence.

5. This approach is more efficient than the nested bi-level approach. However, this approach requires an infinite number of reliability constraints to achieve the "true" optimal solution. This is not attainable in practice. Usually, the user stops the optimization at a predefined accuracy and obtains an approximate solution. Therefore, this approach is applicable to high-dimensional problems.

- The method of outer approximations has proofs of convergence (Kirjner-Neto et al., 1998). This implies that there is a convergence proof for the decoupled sequential approach when the limit-sate function is linear in the space of random variables.

In this thesis we implement the decoupled sequential approach in OpenSees and apply it to structures that exhibit nonlinear behaviour. This approach is termed *the decoupled sequential approach by the method of outer approximations* (DSA-MOOA). We also implement a *simplified decoupled sequential approach* (DSA-S) by combining the problem reformulation of DSA-MOOA with the findings of Du and Chen (2002) and Agarwal and Renaud (2004). In the DSA-S approach, the reformulated optimization problem is an inequality constrained optimization problem (Polak, 1997) with a single reliability constraint, as opposed to the infinitely many constraints of the DSA-MOOA approach. The DSA-S approach has the same properties as the DSA-MOOA approach when facing the aforementioned challenges 1–5. However, while the DSA-S approach is more efficient than the DSA-MOOA approach, it lacks the convergence proof of the DSA-MOOA approach.

In additional to the above methods, Royset and Polak (2004b) develop *the decoupled sequential approach by sample average approximations* to solve the RBDO problem in Eq. (1.2). Failure probabilities are here computed using Monte Carlo or importance sampling. The first-order derivative of the failure probability with respect to design variables is obtained analytically and computed using Monte Carlo or importance sampling. The original RBDO problem is reformulated as an inequality constraint optimization problem and solved using standard optimization algorithms. The number of samples increases as the design approaches the optimal design point. Royset and Polak

(2004b) prove that the optimization algorithm converges with the optimal design when the number of samples approaches infinity. This approach has the following advantages and disadvantages:

1. Failure probabilities are computed using Monte Carlo sampling and importance sampling.
2. Additional computational costs are required to determine the optimal design for the highly nonlinear functions in RBDO problems.
3. This approach has the ability to couple with the finite element analysis.
4. The gradient discontinuity problem may cause non-convergence or slow convergence.
5. The computational cost is higher than in the DSA-MOOA approach since the sample average approximations use Monte Carlo sampling or importance sampling to compute the values and gradients of failure probabilities. The number of sampling points increases as the design nears the optimal stage. Therefore, this approach is still applicable to high-dimensional problems.

- The sample average approximations have proofs of convergence even if the limit-sate function is nonlinear in the space of random variables.

1.3 Thesis Organization

Following the introduction, the fundamentals of finite element reliability analysis and optimization theory are introduced in Chapters 2 and 3, respectively. Chapter 2 reviews the concept of finite element reliability and describes FORM, the second-order reliability method, Monte Carlo sampling, and importance sampling. Chapter 3 presents the inequality constraint optimization problem and the semi-infinite optimization problem, as well as their corresponding first-order necessary optimality conditions. The Polak-He algorithm and the method of outer approximation algorithm are also described in this chapter.

Chapter 4 describes the finite element software, OpenSees, which is extended with reliability analysis capability. The element, section, and material objects used in this thesis are emphasized in this chapter. The reliability analysis module adds the *ReliabilityDomain* and *ReliabilityAnalysis* objects to the original OpenSees. The former defines the random variables and their correlation. It also maps the values of the random variables into the finite element model. The latter includes eight analysis types and several analysis tools.

Chapter 5 introduces the sensitivity analysis capability in OpenSees. Two main response sensitivity methods, the finite difference method and the direct differentiation method, are two analysis tools in OpenSees. The direct differentiation method is stressed in the chapter by briefly describing its equation derivation and several implementation issues. Chapter 5 ends with a section on the continuity of response gradients. The potential negative effects of discontinuous structural responses are observed and remedied using two methods: the smooth material model and the section discrimination scheme.

Chapter 6 presents the problem reformulation and algorithms of the DSA-MOOA and DSA-S approaches. The reformulated optimization problem is identical to the original RBDO problem when the limit-state function is linear in the space of random variables. The applications of the method of outer approximation and the Polak-He algorithms are described in detail in this chapter. The optimization capability of OpenSees is extended through the addition of several objects defining all functions involved in the optimization problem. Moreover, two analysis types (DSA-MOOA analysis and DSA-S analysis) and several analysis tools are added.

Chapter 7 presents a numerical example involving a nonlinear finite element analysis of a three-bay, six-storey building to demonstrate the new implementations in OpenSees. Three cases are studied: a linear pushover analysis using *elasticBeam* elements, a nonlinear pushover analysis using *beamWithHinges* elements, and a nonlinear pushover analysis using *dispBeamColumn* elements with *fibre* sections. The case studies focus on convergence performance and computational time. This chapter also presents practical

experience by comparing two implemented approaches, two gradient computation methods, and linear and nonlinear analyses. Speeding up the convergence procedure by removing inactive constraints and scaling the functions involved are also discussed.

Chapter 8 summarizes the major findings of this thesis and points to areas of future study. Appendix A offers the detailed software implementation used in the thesis, while Appendix B contains a user guide to the new implementations.

Chapter 2 Finite Element Reliability Analysis

Since the introduction of limit-state design in the early 1980s, reliability methods have been masked by partial safety coefficients in prescriptive design code requirements. Direct usage of reliability methods has only been observed in special projects, such as offshore structures and nuclear power plants. This paradigm is changing with the introduction of performance-based engineering. Over the past several decades, analytical and numerical models have drastically improved the engineers' ability to predict structural performance. However, such predictions can only be made in a probabilistic sense. Unavoidable uncertainties are present in model parameters and in the numerical model itself. Reliability analysis and probabilistic methods are therefore rapidly becoming required tools in engineering practice. This chapter presents reliability analysis in conjunction with nonlinear finite element models to make probabilistic predictions of structural response. Such analysis is required to solve the reliability-based design optimization (RBDO) problem in Eq. (1.2).

The name *finite element reliability* stems from the combination of advanced reliability methods with finite element analysis to estimate the probability of exceeding prescribed structural response criteria. As mentioned in Chapter 1, the structural model is defined in terms of vector **v** of random variables and vector **x** of deterministic design variables. The structural response of interest is collected in a vector **d**, which depends upon random and design variables: **d**=**d**(**x**,**v**). A failure criterion is prescribed in terms of the response quantities **d** by means of a limit-state function $g(\mathbf{d})$. Conventionally, a negative outcome of the limit-state function, $g \le 0$, is defined as a failure, while a positive outcome, $g > 0$, is defined as safe. A typical limit-state function employed in this thesis is expressed by a threshold $\hat{d}$ and a response quantity d in the form

$$g(\mathbf{d}(\mathbf{x},\mathbf{v})) = \hat{d} - d(\mathbf{x},\mathbf{v}) \tag{2.1}$$

We observe that when the response d exceeds the threshold $\hat{d}$, the limit-state function becomes negative, as required by the syntax rules. It is emphasized that response quantities d may represent displacements, stresses, strains, etc.

The reliability problem described by a single limit-state function is termed the *component reliability problem.* If failure is prescribed by the joint state of several limit-state functions, the reliability problem is referred to as a *system reliability problem.*

In the component reliability problem, the probability of failure $p_f(\mathbf{x})$ is defined by the integration of the joint probability density function (PDF) $f(\mathbf{v})$ of the random vector $\mathbf{v}$ over the failure domain in the space of random variables:

$$p_f(\mathbf{x}) = \int\int_{g(\mathbf{d}(\mathbf{x},\mathbf{v}))\leq 0} f(\mathbf{v})d\mathbf{v} \tag{2.2}$$

The failure probability depends on the design variables $\mathbf{x}$. In addition, we note that the failure probability does not change if the limit-state function is arbitrarily scaled using a finite positive number. This is important for later developments.

Analytical solutions to Eq. (2.2) are generally not available, and approximate methods are employed to evaluate the failure probability. In these reliability methods, it is common to transform the problem into the standard normal space. That is, the original random variables $\mathbf{v}$ are transformed into a vector $\mathbf{u}$ of uncorrelated standard normal random variables. The Nataf model (Liu & Der Kiureghian, 1986) is an example of such a transformation. $T_{\mathbf{x}}$ is denoted as the transformation for a given design vector $\mathbf{x}$ and replace the random vector $\mathbf{v}$ by $T_{\mathbf{x}}^{-1}(\mathbf{u})$. We then obtain the equivalent limit-state function $g(\mathbf{d}(\mathbf{x},\mathbf{v})) = g(\mathbf{d}(\mathbf{x},T_{\mathbf{x}}^{-1}(\mathbf{u})))$. The joint PDF of $\mathbf{u}$ is the joint standard normal PDF $\varphi(\mathbf{u})$. Hence, the reliability problem is re-defined in the standard normal space as:

$$p_f(\mathbf{x}) = \int\int_{g(\mathbf{d}(\mathbf{x},\mathbf{u}))\leq 0} \varphi(\mathbf{u})d\mathbf{u} \tag{2.3}$$

For system reliability problems with limit-state functions g_k, $k \in \{1,2,...,K\}$, the failure domain is specified as a series system, a parallel system, or a general system. In a series

system, failure occurs when any components (limit-state functions) fail. That is, the failure domain is defined by the union of failure domains of components:

$$\left\{ \mathbf{u} \;\middle|\; \bigcup_k g_k(\mathbf{d}(\mathbf{x},\mathbf{u})) \le 0 \right\} \tag{2.4}$$

Conversely, the failure domain of a parallel structural system is defined by the intersection of failure domains of all components:

$$\left\{ \mathbf{u} \;\middle|\; \bigcap_k g_k(\mathbf{d}(\mathbf{x},\mathbf{u})) \le 0 \right\} \tag{2.5}$$

For a general system, the definition of the failure domain involves both union and interaction operations.

2.1 First-Order and Second-Order Reliability Methods

The first-order reliability method (FORM) approximates the integration boundary $g(\mathbf{d}(\mathbf{x},\mathbf{u})) = 0$ using a hyperplane in the standard normal space. The ideal point to linearize the limit-state function is denoted as $\mathbf{u}^*$ and is the point on the hyperplane closest to the origin:

$$\mathbf{u}^*(\mathbf{x}) = \arg\min\{ \|\mathbf{u}\| \mid g(\mathbf{d}(\mathbf{x},\mathbf{u})) = 0 \} \tag{2.6}$$

$\mathbf{u}^*$ is the point in the failure domain with the highest probability density and is therefore termed the *most probable point* (MPP). Often, Eq. (2.6) is defined with the inequality constraint $g(\mathbf{d}(\mathbf{x},\mathbf{u})) \le 0$ in place of the equality constraint. This is acceptable, as long as the origin is in the safe domain. This is the case for most practical problems, where the failure probability is much less than 0.5.

Searching for the MPP in Eq. (2.6) is an optimization problem in itself. This optimization process requires the first-order derivative of the limit-state function to be continuous in the standard normal space $\mathbf{u}$. One effective algorithm for searching for the MPP is the iHLRF-algorithm (Hasofer & Lind, 1974) (Rackwitz & Fiessler, 1978)

(Zhang & Der Kiureghian, 1997), which is gradient-based and employs a line search scheme. In addition, Eq. (2.6) with an inequality constraint formulation can be solved using the Polak-He algorithm (Polak, 1997) or using standard nonlinear optimization algorithms such as NLPQL (Schittkowski, 1985) or NPSOL (Gill et al., 1998). The latter algorithms, however, are not specialized for the present case of one constraint.

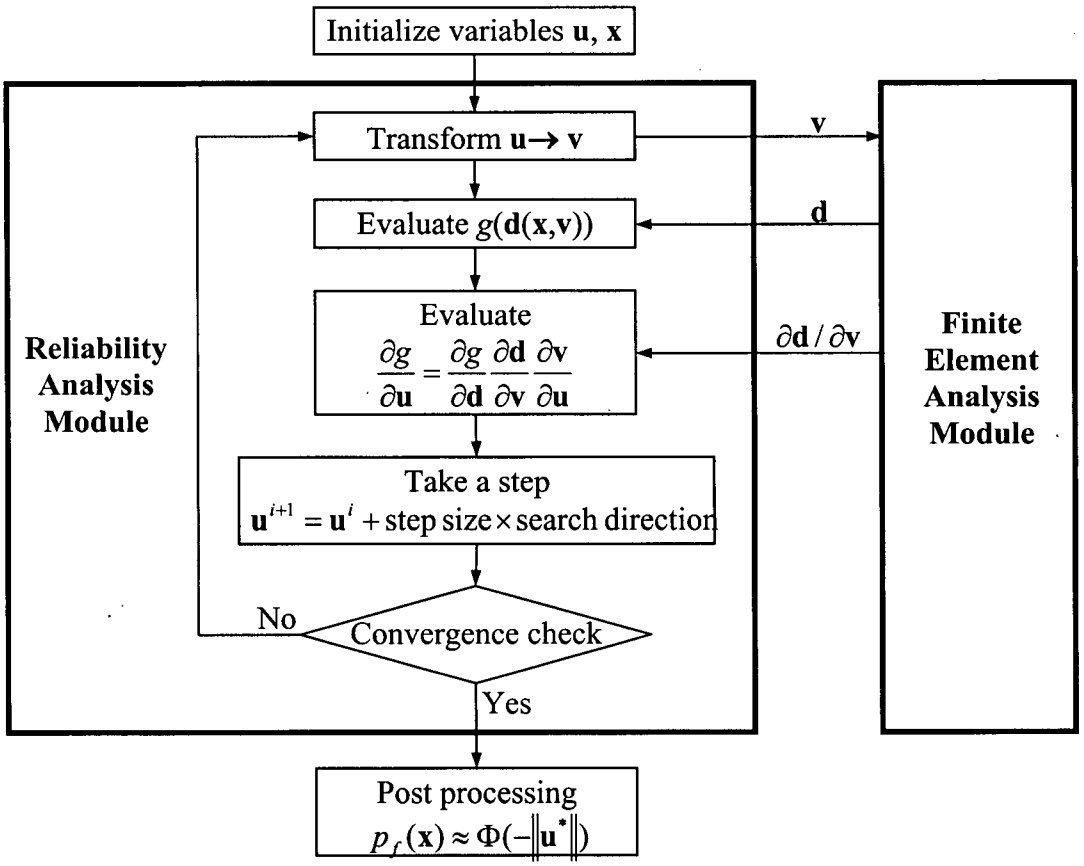

Figure 2.1 MPP searching algorithm in finite element reliability analysis

Haukaas and Der Kiureghian (2004) employ the iHLRF algorithm and the Polak-He algorithm to find the MPP for finite element reliability problems. The outline of the search algorithm is shown in Figure 2.1. The search algorithm requires a transformation between the original **v**-space and the standard normal **u**-space. The value of limit-state function g and the gradient $\partial g / \partial \mathbf{u}$ are evaluated in this algorithm. They are used in finding a search direction and a step size. The figure also shows the interaction between the search algorithm and the finite element code. The finite element analysis module is

repeatedly updated with new realizations of the random variables **v**. In return, the finite element analysis module produces the response **d** and the response gradients $\partial \mathbf{d} / \partial \mathbf{v}$. The need for response derivatives is due to the need for the gradient of the limit-state function in the standard normal space. The chain rule of differentiation yields:

$$\frac{\partial g}{\partial \mathbf{u}} = \frac{\partial g}{\partial \mathbf{d}} \frac{\partial \mathbf{d}}{\partial \mathbf{v}} \frac{\partial \mathbf{v}}{\partial \mathbf{u}} \tag{2.7}$$

where $\partial g / \partial \mathbf{d}$ is easily found because g is a simple algebraic expression in terms of **d**, $\partial \mathbf{d} / \partial \mathbf{v}$ are the response gradients, and $\partial \mathbf{v} / \partial \mathbf{u}$ is the Jacobian matrix from the probability transformation.

The distance from the origin to the MPP in the standard normal space is called the reliability index β:

$$\beta(\mathbf{x}) = \left\| \mathbf{u}^*(\mathbf{x}) \right\| \tag{2.8}$$

The FORM approximation of the failure probability reads:

$$p_f(\mathbf{x}) \approx \Phi(-\beta(\mathbf{x})) \tag{2.9}$$

where $\Phi(\cdot)$ is the standard normal cumulative distribution function.

The second-order reliability method approximates the limit-state function using a quadratic surface that passes through the MPP. Breitung (1984) offers a simple formula to compute the component failure probability based on such a quadratic surface approximation:

$$p_f(\mathbf{x}) \approx \Phi(\beta(\mathbf{x})) \prod_{j=1}^{m-1} (1 + \beta(\mathbf{x}) \kappa_j(\mathbf{x}))^{-1/2} \tag{2.10}$$

where $\kappa_j(\mathbf{x}), j = 1,\ldots,m-1$ are the principal curvatures of the limit-state surface $g(\mathbf{d}(\mathbf{x},\mathbf{u})) = 0$ at the MPP.

2.2 Monte Carlo and Importance Sampling

Monte Carlo sampling is an alternative method used to approximate the failure probability $p_f(\mathbf{x})$. Monte Carlo sampling generates a family of simulated points $\mathbf{u}_1$, ..., $\mathbf{u}_N$, which are statistically independent standard normal random variables. The indicator function $I(\mathbf{u}_i)$ corresponding to each simulated point is established using the following rule: $I(\mathbf{u}_i) = 1$ whenever $g(\mathbf{d}(\mathbf{x},\mathbf{u})) \le 0$, and $I(\mathbf{u}_i) = 0$ otherwise. Monte Carlo sampling gives an approximation of the component failure probability $p_f(\mathbf{x})$ by:

$$p_f(\mathbf{x}) \approx \frac{1}{N}\sum_{i=1}^{N} I(\mathbf{u}_i) \tag{2.11}$$

The quality of the solution of Monte Carlo sampling is measured using the coefficient of variation (c.o.v.) of the probability estimate: $\text{c.o.v.} = \sqrt{(1 - p_f(\mathbf{x}))/(N \cdot p_f(x))}$. Monte Carlo sampling is stopped at the user-defined number of samplings, or when the coefficient of variation achieves a specified target, for example 2%.

In order to estimate small failure probabilities accurately, crude Monte Carlo sampling requires a large number of simulations. This is because the sampling distribution of crude Monte Carlo sampling is centered at the mean point, while failure events are inclined to occur in the tail region of the probability distributions.

Importance sampling improves the efficiency of Monte Carlo sampling by centering the sampling distribution near the failure domain. Usually, this centre is selected as the MPP from FORM analysis. Importance sampling generates a family of simulated points $\mathbf{u}_1$, ..., $\mathbf{u}_N$ with the probability density $h(\mathbf{u})$. All simulated points are statistically independent standard normal random variables. $h(\mathbf{u})$ is a joint PDF and nonzero in the failure domain. Importance sampling gives an approximation of component failure probability $p_f(\mathbf{x})$ using:

$$p_f(\mathbf{x}) \approx \frac{1}{N}\sum_{i=1}^{N} I(\mathbf{u}_i)\frac{\varphi(\mathbf{u}_i)}{h(\mathbf{u}_i)} \tag{2.12}$$

where $\varphi(\cdot)$ is the standard normal PDF (Ditlevsen & Madsen, 1996). The coefficient of variation of failure probability in importance sampling is defined by:

$$\text{c.o.v.} = \frac{\sqrt{\frac{1}{N(N-1)}\left[\sum_{i=1}^{N}\left(I(\mathbf{u}_i)\frac{\varphi(\mathbf{u}_i)}{h(\mathbf{u}_i)}\right)^2 - \frac{1}{N}\left(\sum_{i=1}^{N} I(\mathbf{u}_i)\frac{\varphi(\mathbf{u}_i)}{h(\mathbf{u}_i)}\right)^2\right]}}{p_f(\mathbf{x})} \tag{2.13}$$

Importance sampling is efficient and requires fewer simulations than Monte Carlo sampling since the sampling distribution is centered on the MPP, where failure realizations are frequently encountered.

2.3 Gradient of the Failure Probability

When gradient-based algorithms are employed in RBDO, it is important to note whether the derivative of the failure probability can be computed and whether it is continuous. The answers to these questions are not straightforward. In fact, the "brute force" application of gradient-based algorithms, such as the mono-level approach explored by Enevoldsen and Sorensen (1994), requires the gradient of failure probability with respect to design variables to be available. In FORM analysis, an analytical formulation of $\partial p_f / \partial x$ is possible. Eq. (2.9) is differentiated to obtain:

$$\frac{\partial p_f}{\partial x} = \frac{\partial}{\partial x}\Phi(-\beta) = \frac{\partial}{\partial x}\left[1-\Phi(\beta)\right] = -\frac{\partial \beta}{\partial x}\frac{\partial}{\partial \beta}\Phi(\beta) = -\frac{\partial \beta}{\partial x}\varphi(\beta) \tag{2.14}$$

The derivative of the reliability index is:

$$\frac{\partial \beta}{\partial x} = \frac{1}{\|\nabla_{\mathbf{u}} g\|}\frac{\partial g}{\partial x} \tag{2.15}$$

where $\nabla_{\mathbf{u}} g$ is the by-product of FORM analysis and $\partial g / \partial x = (\partial g / \partial \mathbf{d})(\partial \mathbf{d} / \partial x)$ is readily obtained by utilizing response sensitivities from the finite element code (Hohenbichler & Rackwitz, 1986) (Bjerager & Krenk, 1989). However, the derivative of the failure probability in Eq. (2.14) cannot be proven to be continuous. This is because in a

reliability problem, the MPP may jump to a different location on the limit-state surface g=0 due to an infinitesimal perturbation of the design variables. This jump leads to a kink on the function $p_f(\mathbf{x})$ in the **x**-space. In effect, the gradient of the failure probability in the **x**-space is discontinuous.

Having demonstrated that the derivative of the failure probability in FORM is possible to obtain but is not continuously differentiable, we move to the case of sampling analysis. Formulae for the sensitivity of failure probability from sampling became available only recently. In Royset and Polak (2004b), indicator functions $I(\mathbf{u}_i)$ in Eqs. (2.11) and (2.12) are replaced by the joint standard normal cumulative distribution function. Then, the reformulated failure probability is differentiated. The sensitivity of the failure probability is expressed through the joint standard normal PDF and evaluated using Monte Carlo or importance sampling.

It is important to note that not all gradient-based algorithms that address the problem in Eq. (1.2) require the gradient $\partial p_f / \partial x$. In fact, as will be shown in subsequent chapters, the algorithms implemented in this thesis circumvent the problem of actually computing this gradient by using an augmented design variable to take place of the failure probability.

Chapter 3 Optimization Theory and Algorithms

This thesis implements two decoupled sequential approaches to solve the reliability-based design optimization problem expressed in Eq. (1.2). These approaches profit from the advantages of both mono-level and bi-level approaches that were discussed in Chapter 1. The optimization problem in Eq. (1.2) is reformulated to enable the use of the method of outer approximation (MOOA) for semi-infinite inequality optimization problems, or the use of the Polak-He algorithm for ordinary inequality constraint optimization problems (Polak, 1997). This chapter defines the fundamental concepts of the optimization theory, which forms the basis for the subsequent problem reformulation and the corresponding solution algorithms.

3.1 Inequality Constrained Optimization Problem

This section introduces the first-order optimality conditions and the Polak-He algorithm that builds upon them for solving the deterministic inequality constrained optimization problem of the form

$$\mathbf{x}^* = \arg\min\{\ F(\mathbf{x}) \mid f(\mathbf{x}) \le 0\ \} \tag{3.1}$$

where $\mathbf{x}^*$ is the design point, $F(\mathbf{x})$ is the objective function, and $f(\mathbf{x})$ is the maximum value of the n-dimensional vector of constraints $\mathbf{f}(\mathbf{x})$ (Polak, 1997):

$$f(\mathbf{x}) = \max\ \mathbf{f}(\mathbf{x}) \tag{3.2}$$

3.1.1 First-Order Optimality Conditions

A candidate solution to any optimization problem must satisfy the optimality conditions of the problem. These are generally *necessary* conditions, but they are not *sufficient* to guarantee that the optimal point has been found. Moreover, optimality conditions can

only indicate whether a local minimum has been found. This is a fundamental problem with any optimization algorithm; only through engineering insight and repeated searches from different starting points, among other strategies, can we confidently state that the global optimal point has been found. The problem is schematically illustrated in Figure 3.1, where it is shown that an objective function may have a local as well as a global minimum. It is easy to imagine that the search algorithm may "get stuck" at the local minimum, without realizing that another point is the actual solution to the optimization problem. A repeated search with a new start point may reveal the global solution.

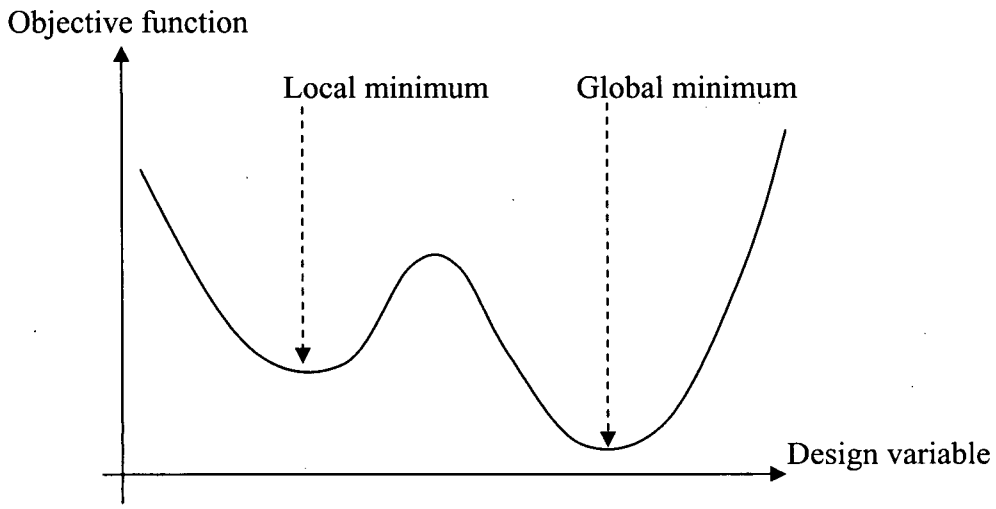

Figure 3.1 Local and global optimal points

Optimality conditions are not only convergence criteria for the optimization algorithm. They are often used to construct search algorithms to solve the optimization problem. This further motivates the following exposure of optimality conditions for different optimization problems.

For pedagogical purposes, first consider a deterministic optimization problem without constraints:

$$\mathbf{x}^* = \arg\min\{ F(\mathbf{x}) \} \tag{3.3}$$

An optimality condition for this problem is clearly $\nabla F = 0$, where $\nabla F \equiv \partial F / \partial \mathbf{x}$ is the gradient of the objective function. This is equivalent to the requirement that a function with one variable have a zero derivative at extremum points.

Next, consider a deterministic inequality constrained optimization problem with one constraint:

$$\mathbf{x}^* = \arg\min\{ F(\mathbf{x}) \mid f(\mathbf{x}) \le 0 \} \tag{3.4}$$

where $f(\mathbf{x})$ is the constraint. Note that a problem with the equality constraint $f(\mathbf{x})=0$ may be reformulated into an inequality constrained problem with two constraints, $f(\mathbf{x}) \le 0$ and $-f(\mathbf{x}) \le 0$. In other words, we introduce two inequality constraints for every equality constraint.

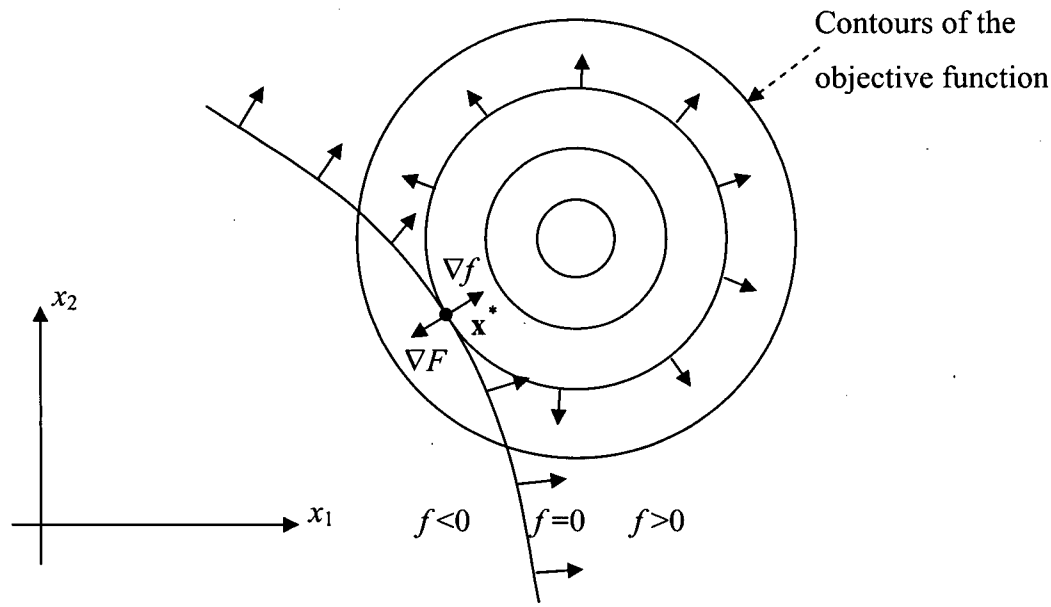

Figure 3.2 Constrained optimization problems

For the constrained optimization problem in Eq. (3.4), the gradient of $F(\mathbf{x})$ need not vanish at the solution point (refered to as the design point). Instead, two cases are possible: (1) the constraint is not active at the design point, in which case

$$\nabla F = 0 \tag{3.5}$$

at the design point; (2) the constraint is active at the design point, in which case the gradient of the objective function is proportional to the gradient of the constraint at the design point. Figure 3.2, in which the minimization of an objective function with two design variables is considered, clarifies this concept. The contours of the objective

function are shown as circles, while the constraint limit $f = 0$ is shown as a line. The solid arrows in the figure depict the gradients of objective functions and constraint functions at certain points. At the design point $\mathbf{x}^*$ gradient vectors ∇F and ∇f are clearly parallel, although they point in different directions. This orthogonality at the design point between the gradient of the objective function and the gradient of the constraint is written as

$$\mu_0 \nabla F = -\mu_1 \nabla f \tag{3.6}$$

where μ_0 and μ_1 are positive constants.

The two optimality conditions in Eqs. (3.5) and (3.6) are combined into one equation by first defining an auxiliary function termed the "Lagrange function:"

$$L(\mathbf{x}) = \mu_0 F(\mathbf{x}) + \mu_1 f(\mathbf{x}) \tag{3.7}$$

where μ_0 and μ_1 are denoted Lagrange multipliers. The method of Lagrange multipliers is a traditional method of enforcing constraints in an optimization problem. This method requires the derivatives of the Lagrange function with respect to the design variables to be zero:

$$\nabla L = \mu_0 \nabla F + \mu_1 \nabla f = 0 \tag{3.8}$$

With the additional requirement that either $\mu_1 = 0$ or $f = 0$, both of the above cases (active and inactive constraint) are included. First, considering the case where the constraint is active at the design point, Eq. (3.8) can be turned into Eq. (3.6) by letting $f = 0$ and $\mu_1 \geq 0$. Second, considering the case where the constraint is inactive, Eq. (3.8) can be turned into Eq. (3.5) by setting $\mu_1 = 0$. In conclusion, the optimality conditions for the problem in Eq. (3.4) read

$$\nabla L = 0 \quad \text{and} \quad \mu_1 f = 0 \tag{3.9}$$

where $\mu_1 f = 0$ implies that either μ_1 or f must be zero. Additionally, we must have $f \leq 0$, $\mu_0 > 0$, and $\mu_1 \geq 0$.

Turning to the case of multiple inequality constraints, the optimization problem reads

$$\mathbf{x}^* = \arg\min\{\ F(\mathbf{x}) \mid \mathbf{f}(\mathbf{x}) \le \mathbf{0}\ \} \tag{3.10}$$

where $\mathbf{f}(\mathbf{x})$ is the vector of constraints. In this case, one positive Lagrange multiplier is introduced for each constraint. Consequently, the Lagrange function reads

$$L(\mathbf{x}) = \mu_0 F(\mathbf{x}) + \mu_1 f_1(\mathbf{x}) + \mu_2 f_2(\mathbf{x}) + \cdots + \mu_n f_n(\mathbf{x}) = \mu_0 F(\mathbf{x}) + \boldsymbol{\mu}\mathbf{f}(\mathbf{x}) \tag{3.11}$$

where $\boldsymbol{\mu}$ is the n-dimensional vector of Lagrange multipliers. The optimality conditions for this case are referred to as the Karush-Kuhn-Tucker conditions, which take the form

$$\nabla L = 0 \quad \text{and} \quad \boldsymbol{\mu}\mathbf{f} = 0 \tag{3.12}$$

again with $\mathbf{f} \le \mathbf{0}$, $\mu_0 > 0$, and $\boldsymbol{\mu} \ge \mathbf{0}$. These are the first-order optimality conditions for the inequality constrained optimization problem in Eq. (3.1). We note that

$$\nabla L = \mu_0 \nabla F(\mathbf{x}^*) + \boldsymbol{\mu} \nabla \mathbf{f}(\mathbf{x}^*) \tag{3.13}$$

Additionally, we require that $\mu_0 + \mu_1 + \mu_2 + \cdots + \mu_n = 1$ for the reason of normalization, and that all involved functions F and $\mathbf{f}$ are continuously differentiable (Polak, 1997).

3.1.2 The Polak-He Algorithm

The use of optimality conditions to obtain corresponding solution algorithms is addressed by Polak (1997). The Polak-He algorithm is presented for the case of multiple objective functions, where the objective is to minimize the maximum of the objective functions, subject to multiple constraints. In this thesis we apply the algorithm to two cases: (1) one objective function with one constraint; and (2) one objective function with multiple constraints. We start with the more general latter case. The Polak-He algorithm based on the first-order optimality conditions is introduced below.

Polak-He Algorithm

Parameters. Select parameters $0 < \alpha \le 1$, $0 < \beta \le 1$, $0 < \delta$, $0 < \gamma$.

Input Data. Input the initial design $\mathbf{x}_0$.

Step 0. Set $i = 0$.

Step 1. Compute the search direction vector $\mathbf{h}_i$ (see below).

Step 2. Calculate the step size λ_i along the search direction $\mathbf{h}_i$ using the Armijo rule (see below).

Step 3. Update the design $\mathbf{x}_{i+1} = \mathbf{x}_i + \lambda_i \mathbf{h}_i$, replace i by $i + 1$, and go to **Step 1**.

Step 1 computes the search direction vector $\mathbf{h}_i$ by solving a quadratic sub-optimization problem, which is the combination of the first-order optimality conditions in Eq. (3.12).

$$\theta_i(\mathbf{x}) = -\min\left\{\mu_0 \gamma\, f(\mathbf{x})_+ + \boldsymbol{\mu}\left[f(\mathbf{x})_+ - \mathbf{f}(\mathbf{x})\right] + \frac{1}{2\delta}\left\|\mu_0 \nabla F(\mathbf{x}) + \boldsymbol{\mu}\nabla \mathbf{f}(\mathbf{x})\right\|^2\right\} \quad (3.14)$$

where $f(\mathbf{x})_+ = \max\{0, \mathbf{f}(\mathbf{x})\}$. Eq. (3.14) minimizes a quadratic function dependent on Lagrange multipliers μ_0 and $\boldsymbol{\mu}$, subject to the linear constraint $\mu_0 + \mu_1 + \mu_2 + \cdots + \mu_n = 1$. This problem is solved in a finite number of iterations, and its solutions are μ_0^* and $\boldsymbol{\mu}^*$. Then, the search direction vector $\mathbf{h}_i$ is computed using

$$\mathbf{h}_i = -\frac{1}{\delta}\left\{\mu_0^* \nabla F(\mathbf{x}_i) + \boldsymbol{\mu}^* \nabla \mathbf{f}(\mathbf{x})\right\} \quad (3.15)$$

The Armijo rule used in Step 2 is a line search scheme, which applies a merit function in the step-size selection. Ideally, the step size is determined at the minimum merit function since this leads to an optimal rate of convergence. The merit function in this case is

$$M(\mathbf{x}_i, \mathbf{x}_i + \beta^s \mathbf{h}_i) = \max\left\{F(\mathbf{x}_i + \beta^s \mathbf{h}_i) - F(\mathbf{x}_i) - \gamma\, f(\mathbf{x}_i)_+, \mathbf{f}(\mathbf{x}_i + \beta^s \mathbf{h}_i) - f(\mathbf{x}_i)_+\right\} \quad (3.16)$$

where s is an integer to be introduced shortly. The step-size selection must meet the requirement of $M(\mathbf{x}_i, \mathbf{x}_i + \beta^s \mathbf{h}_i) \le \alpha \beta^s \theta_i$. In Eq. (3.16), the parameter s is an integer starting from the initial value 0. The parameter s increases or decreases by unit steps until an *acceptable* and *maximum* step size is found. Finally, the appropriate step size λ_i is set equal to the maximum value of β^s and corresponding to the minimum merit function:

$$\lambda_i = \max\left\{ \beta^s \;\middle|\; M(\mathbf{x}_i, \mathbf{x}_i + \beta^s \mathbf{h}_i) \le \alpha \beta^s \theta_i \right\} \tag{3.17}$$

The inequality constrained optimization problem in Eq. (3.1) is a nonlinear optimization problem. Theoretically, it requires an infinite number of iterations (step 1 to 3) to converge to an optimal solution. This is not applicable to practical problem solving. Usually, the Polak-He algorithm is terminated when a user-defined precision ε (*e.g.*, 10^{-6}) is reached. When two adjacent solutions are very close ($\|\mathbf{x}_{i+1} - \mathbf{x}_i\| \le \varepsilon$), the merit function $M(\mathbf{x}_i, \mathbf{x}_{i+1})$ and/or the value of sub-optimization $\theta(\mathbf{x}_i)$ in Eq. (3.14) is sufficiently close to zero, the Polak-He algorithm is stopped, resulting in an approximate optimal solution.

It can be shown that solutions generated by the Polak-He algorithm are optimal for the inequality constrained optimization problem. Solution $\mathbf{x}_i$ from the Polak-He algorithm approaches the optimal solution $\mathbf{x}^*$ as the number of iterations approaches infinity. Since the Polak-He algorithm is developed from the first-order optimality conditions, the optimality conditions in Eq. (3.12) are satisfied automatically at the point of optimal solution. In the meantime, the sub-optimization problem in Eq. (3.14) also reaches the minimum value, *i.e.*, $\theta(\mathbf{x}^*) = 0$.

3.2 Semi-Infinite Optimization Problem

This section describes the first-order optimality conditions and the MOOA algorithm for solving the semi-infinite optimization problem of the form

$$\mathbf{x}^* = \arg\min\left\{ \psi_0(\mathbf{x}) \mid \psi(\mathbf{x}) \le 0 \right\} \tag{3.18}$$

where $\psi_0(\mathbf{x})$ is the objective function, and $\psi(\mathbf{x})$ is the maximum value of the n-dimensional constraints (Polak, 1997). The objective and constraint functions ψ are defined by

$$\psi(\mathbf{x}) = \max\ \psi_j(\mathbf{x}) \tag{3.19}$$

and ψ_j, $j = \{0, 1, ..., n\}$ is given by

$$\psi_j(\mathbf{x}) = \max_{\mathbf{y} \in \mathbf{Y}_j} \phi_j(\mathbf{x}, \mathbf{y}) \tag{3.20}$$

where the function $\phi_j(\mathbf{x}, \mathbf{y})$ is determined by the design vector $\mathbf{x}$ and the extra argument $\mathbf{y}$, which are all points in the domain $\mathbf{Y}_j$, *i.e.*, $\mathbf{y} \in \mathbf{Y}_j$. The design vector $\mathbf{x}$ is finite-dimensional, but the number of functions $\phi_j(\cdot, \mathbf{y})$ is infinity because of the infinite number of points $\mathbf{y}$ in the domain $\mathbf{Y}_j$. That is the reason for the term "semi-infinite." An example of function $\phi_j(\mathbf{x}, \mathbf{y})$ is the negative value of limit-state function $-g(\mathbf{x}, \mathbf{u})$, in which $\mathbf{x}$ is the design vector, $\mathbf{u}$ is the random vector (namely point $\mathbf{y}$), and domain $\mathbf{Y}_j$ is the standard normal space. As mentioned in Chapter 2, a positive outcome of the limit-state function ($g \geq 0$) is defined as safe; hence the constraints $-g(\mathbf{x}, \mathbf{u}) \leq 0$ ensure a safe structure.

3.2.1 First-Order Optimality Conditions

First-order optimality conditions for semi-infinite optimization problems require the involved functions ϕ_j to be continuously differentiable. Additionally, the domain $\mathbf{Y}_j$ must be bounded and closed. Similar to Eq. (3.12), the necessary optimality conditions for semi-infinite optimization problems are

$$0 \in \sum_{j=0}^{n} \mu_j \nabla \psi_j(\mathbf{x}^*) \quad \text{and} \quad \sum_{j=0}^{n} \mu_j \psi_j(\mathbf{x}^*) = 0 \tag{3.21}$$

where $\mathbf{x}^*$ denotes the optimal design, and Lagrange multipliers $\mu_0, \cdots, \mu_n$ are positive-defined. Additionally it is required that $\mu_0 + \mu_1 + \mu_2 + \cdots + \mu_n = 1$ for the reason of normalization (Polak, 1997).

The optimality conditions in Eq. (3.21) require that at least one constraint $\psi_j(\mathbf{x}^*)$ be active at the optimal design point. Therefore, the value of this constraint must be zero

($\psi_j(\mathbf{x}^*) = 0$), while the corresponding Lagrange multiplier must be larger than zero ($\mu_j > 0$). This situation is illustrated in Figure 3.2, where the constraint $f = 0$ is active at the design point $\mathbf{x}^*$, and the objective function reaches minimum in the mean time.

3.2.2 Method of Outer Approximation Algorithm

The MOOA algorithm solves the semi-infinite optimization problem by repeatedly solving a series of inequality-constrained problems. In these inequality-constrained problems, progressively more constraints are used to get a gradually more accurate solution (Polak, 1997). A discretization method is used to discretize the domain $\mathbf{Y}_j$ and produce more constraints, in which case, the domain $\mathbf{Y}_j$ is discretized into N number of sub-domains $\mathbf{Y}_{j,N} \subset \mathbf{Y}_j$. For example, a domain [0,1] is descretized as the sub-domain {0, 1/4, 2/4, 3/4, 1} when $N = 4$. These approximate sub-domains $\mathbf{Y}_{j,N}$ are sequentially constructed as the algorithm progresses. Then, the semi-infinite problem is approximated as an inequality constrained problem with N constraints. When N equals infinity, the solution of the inequality-constrained problem is the optimal solution of the semi-infinite problem.

Using the above discretization method, we approximate Eqs. (3.19) and (3. 20) for a natural number $N = 1, 2, 3, \ldots$ as

$$\psi_N(\mathbf{x}) = \max\ \psi_{j,N}(\mathbf{x}) \tag{3.22}$$

$$\psi_{j,N}(\mathbf{x}) = \max_{\mathbf{y} \in \mathbf{Y}_{j,N}} \phi_j(\mathbf{x}, \mathbf{y}) \tag{3.23}$$

Similar to the Polak-He algorithm, the approximation to the merit function in the Armijo rule is denoted in the following:

$$M_N(\mathbf{x}', \mathbf{x}'') = \max\left\{\psi_{0,N}(\mathbf{x}'') - \psi_{0,N}(\mathbf{x}') - \gamma\psi_N(\mathbf{x}')_+,\ \psi_N(\mathbf{x}'') - \psi_N(\mathbf{x}')_+\right\} \tag{3.24}$$

where $\gamma > 0$ is a parameter, and $\psi_N(\mathbf{x})_+ = \max\{0, \psi_N(\mathbf{x})\}$.

In addition, a quadratic sub-optimization problem $\theta_N(\mathbf{x})$ is constructed to obtain a search direction **h**

$$\theta_N(\mathbf{x}) = \min_{\mathbf{h}} \tilde{M}_N(\mathbf{x}, \mathbf{x}+\mathbf{h}) \tag{3.25}$$

where $\tilde{M}_N(\mathbf{x}, \mathbf{x}+\mathbf{h})$ comes from the following definitions:

$$\tilde{M}_N(\mathbf{x}, \mathbf{x}+\mathbf{h}) = \max\left\{\tilde{\psi}_{0,N}(\mathbf{x}, \mathbf{x}+\mathbf{h}) - \psi_{0,N}(\mathbf{x}) - \gamma\psi_N(\mathbf{x})_+,\ \tilde{\psi}_N(\mathbf{x}, \mathbf{x}+\mathbf{h}) - \psi_N(\mathbf{x})_+\right\} \tag{3.26}$$

$$\tilde{\psi}_N(\mathbf{x}, \mathbf{x}+\mathbf{h}) = \max_{j\in\mathbf{q}} \tilde{\psi}_{j,N}(\mathbf{x}, \mathbf{x}+\mathbf{h}) \tag{3.27}$$

$$\tilde{\psi}_{j,N}(\mathbf{x}, \mathbf{x}+\mathbf{h}) = \max_{\mathbf{y}\in\mathbf{Y}_{j,N}}\left\{\phi_j(\mathbf{x}, \mathbf{y}) + \left\langle \nabla_{\mathbf{x}}\phi_j(\mathbf{x}, \mathbf{y}), \mathbf{h}\right\rangle + \frac{1}{2}\|\mathbf{h}\|^2\right\}, j = \{0,1,...,n\} \tag{3.28}$$

The search direction **h** is defined as

$$\mathbf{h} = -\sum_{j=0}^{n} \mu_j \nabla\psi_j(\mathbf{x}^*) \tag{3.29}$$

The solutions to the sub-optimization problem in Eq. (3.25) are Lagrange multipliers $\mu_0, \cdots, \mu_n$, which satisfy the requirement $\mu_0 + \mu_1 + \mu_2 + \cdots + \mu_n = 1$. Then, the search direction **h** is computed using Eq. (3.29).

Method of Outer Approximations Algorithm

Input Data. Input the initial discretization number N_0, the initial design $\mathbf{x}_{N_0}$, the tolerances $\sigma_N = \tau_N = 0.1/N^2$ and $\varepsilon_{N,k} = \rho^k - \rho^N$, where $0 < \rho < 1$.

Step 0. Set $N = N_0$.

Step 1. Inner approximation: Find a point $\mathbf{y}_{j,N}$ in the domain $\mathbf{Y}_j$.

Step 2. Constraints expansion: Collect all appropriate points $\mathbf{y}_{j,N}$ and update the constraint set using the corresponding constraint $\psi_j(\mathbf{x}_N, \mathbf{y}_{j,N})$.

Step 3. Outer approximation: Find an approximate new design $\mathbf{x}_{N+1}$.

Step 4. Replace **N** with **N** + 1 and go to **Step 1**.

Step 1 employs the Polak-He algorithm to solve the following inequality constrained optimization problem:

$$\mathbf{y}_{j,N} = \arg\min \left\{ \psi_j(\mathbf{x}_N, \mathbf{y}) \right\} \tag{3.30}$$

The solution to Eq. (3.30) is a point $\mathbf{y}_{j,N}$ in the domain $\mathbf{Y}_j$. This step is an inner optimization problem and is terminated when a user-defined tolerance σ_N is satisfied.

In Step 2, all points $\mathbf{y}_{j,k}$, $(j = 1,\cdots,n,\ k = 1,\cdots,N)$ from Step 1 are collected. If the requirement $\max\{0,\ \phi_j(\mathbf{x}_k, \mathbf{y}_{j,k})\} > \varepsilon_{N,k}$ is satisfied, the corresponding constraint $\phi_j(\mathbf{x}_k, \mathbf{y}_{j,k})$ is added to the constraint set of the semi-infinite optimization problem.

Step 3 computes an approximate new solution x_{N+1} of the following inequality constrained optimization problem using the Polak-He algorithm:

$$\mathbf{x}_{N+1} = \arg\min\{\ \psi_{0,N+1}(\mathbf{x}) \mid \psi_{N+1}(\mathbf{x}) \le 0\ \} \tag{3.31}$$

which satisfies

$$\theta_{N+1}(\mathbf{x}_{N+1}) \ge -\tau_N \tag{3.32}$$

$$\psi_{N+1}(\mathbf{x}_{N+1}) \le \tau_N \tag{3.33}$$

with $\theta_{N+1}(\cdot)$ and $\psi_{N+1}(\cdot)$ defined in Eqs. (3.25) and (3.23), respectively.

As a nonlinear optimization problem, the semi-infinite optimization problem requires an infinite number of iterations (steps 1 to 4) to converge to an optimal solution. In general, the MOOA algorithm is terminated when a user-defined precision ε (*e.g.*, 10^{-6}) is reached. In this thesis, when two adjacent solutions are very close ($\|\mathbf{x}_{i+1} - \mathbf{x}_i\| \le \varepsilon$), the MOOA algorithm is stopped, resulting in an approximate optimal solution.

After an iteration of MOOA, Step 3 finds a solution $\mathbf{x}_{N+1}$ of the approximate inequality constrained problem. As the discretization number N increases, the MOOA algorithm

results in a gradually more accurate solution. The optimal solution $\mathbf{x}^*$ is reached when N equals infinity. At the design point $\mathbf{x}^*$, the first-order optimality conditions in Eq. (3.21) are satisfied; namely, at least one constraint is active at that point, and the value of the objective function reaches its minimum.

Chapter 4 The OpenSees Software

The OpenSees software framework (McKenna et al., 2004) serves as the computational platform for research within the Pacific Earthquake Engineering Research (PEER) Center and is rapidly gaining users internationally. Its source code, documentation, and executable files are freely available on the web site *http://opensees.berkeley.edu*.

OpenSees was originally designed to compute the response of nonlinear structural and geotechnical systems using finite element techniques. Haukaas and Der Kiureghian (2004) extended OpenSees with reliability and response sensitivity capabilities for nonlinear finite element analysis. This chapter introduces nonlinear finite element analyses and reliability analyses in OpenSees. The response sensitivity analysis is discussed separately in Chapter 5.

The object-oriented programming approach was employed in the development of OpenSees. The introduction of object-oriented programming has brought with it a revolution in software development (Deitel & Deitel, 1998). This revolution is based on the notion of standardized, interchangeable software components. These components are called objects or, abstractly, classes. Objects are instantiated at run-time based on specifications made by the developer in the corresponding classes. Each class, and hence object, may contain member functions and member data. Detailed specification of the member functions and data members is found in class interfaces. Class interfaces contain key information necessary to understand an object-oriented software framework. Class interfaces also facilitate the transparent nature of object-oriented programming. Their structure is common to all object-oriented software. Armed with the knowledge of universal syntax rules of the programming language such as C++, a user is able to understand the software architecture of a specific object-oriented framework. Such software design has extensibility and maintainability as its integral feature. The programming language C++ is employed in this thesis for the purpose of object-oriented programming.

4.1 Nonlinear Finite Element Analysis

This section introduces the OpenSees software framework for the nonlinear finite element analysis as detailed in the *OpenSees Command Language Manual* (Mazzoni et al., 2005). Element, section, and material objects used in this thesis are emphasized, as is the fundamental knowledge needed for the case studies in Chapter 7.

OpenSees consists of a set of modules, which create a finite element model, specify an analysis procedure, analyze the model, and output the results. A complete finite element analysis involves four main types of objects, as shown in Figure 4.1. The *ModelBuilder* object establishes the finite element model by defining the nodes, elements, loads, and constraints. The *Analysis* object performs simple static linear analyses or transient nonlinear analyses. The structural response, such as the displacement history at a node or the entire state of the model at each load step, is recorded and output by the *Recorder* object. The *Domain* object stores the model created by the *ModelBuilder* object and provides model information for the *Analysis* and *Recorder* objects.

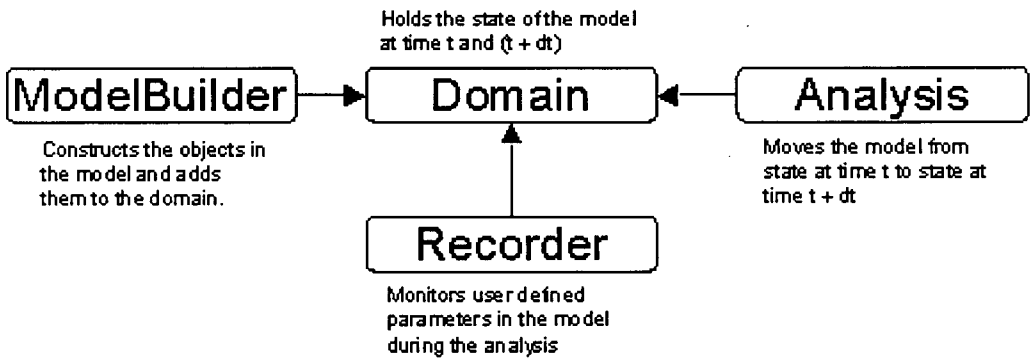

Figure 4.1 Principal OpenSees objects (Mazzoni et al., 2005)

Figure 4.2 shows the relationship among elements, sections, and materials. In general, a structure is comprised of many elements. Each element is divided into several sections, while each section consists of several materials, such as steel and concrete materials. Elements, sections, and materials have their own properties, which are described in the following paragraphs. Hence, the structures created by them can exhibit different types of behaviour, such as linear elastic and nonlinear behaviour.

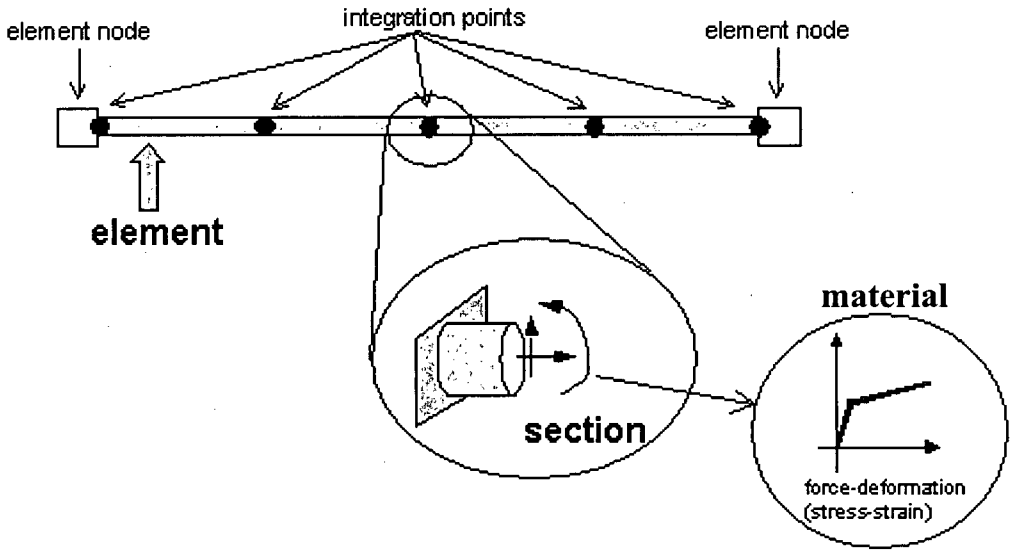

Figure 4.2 Element, section and material relationship (Mazzoni et al. 2005)

One type of element objects is the elastic beam-column element, *elasticBeamColumn*, which is described by the following main parameters: cross-section area A, Young's Modulus E, and the second moment of area I. Moreover, there are basically two types of nonlinear beam-column elements: the displacement-based element, *dispBeamColumn*, and force-based elements, including *beamWithHinges* and *nonlinearBeamColumn*. The *beamWithHinges* element follows the flexibility formulation and contains an elastic interior and two plastic hinges at the each ends. The parameters to construct this element are pre-defined sections at two ends, ratios of the hinge length to the total element length, cross-sectional area A, Young's Modulus E, and the second moment of area I. The parameters A, E, and I are used for the elastic interior, which has linear-elastic properties. Two plastic hinges represent the inelastic regions, in which forces and deformations are sampled at the hinge midpoints using mid-point integration. A *nonlinearBeamColumn* element spreads the distributed plasticity along the element and follows the force formulation. A *dispBeamColumn* element is a displacement-based beam-column element, which has distributed plasticity with linear curvature distribution. To describe these two elements, pre-defined sections and a number of integration points along the element are

required. The integration along the element is based on the Gauss-Lobatto quadrature rule, with two integration points at the element's ends.

A section object defines the stress resultant force-deformation response at a cross section of a beam-column or of a plate element. There are three types of sections: elastic section, defined by material and geometric constants; resultant section, which is a general nonlinear description of force-deformation response (*e.g.* moment-curvature); and fibre section, which is discretized into smaller regions for which the material stress-strain response is integrated to produce resultant behaviour (*e.g.* reinforced concrete) (Mazzoni et al., 2005). A fibre section has a general geometric configuration formed by sub-regions of simpler, regular shapes (*e.g.* quadrilateral, circular, and triangular regions) called patches. In addition, layers of reinforcement bars can be specified. The subcommands patch and layer are used to define the discretization of the section into fibres. Individual fibres, however, can also be defined using the *fibre* object. During generation, the *fibre* objects are associated with material objects, which enforce Bernoulli beam assumptions (Mazzoni et al., 2005). Two examples of fibre sections are shown in Figure 4.3 to describe a circular section and a rectangular section.

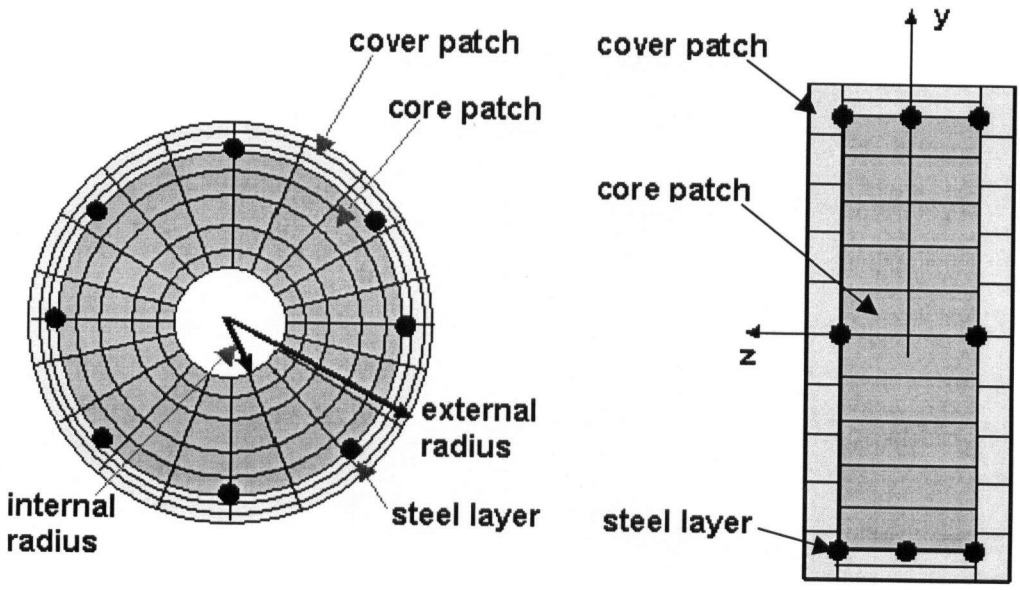

Figure 4.3 Fibre-section examples (Mazzoni et al., 2005)

There are several material objects in OpenSees. This thesis uses the *uniaxialMaterial* object, which represents uniaxial stress-strain or force-deformation relationships (Mazzoni et al., 2005). As a *uniaxialMaterial* object, the *Elastic-Perfectly Plastic* (EPP) material constructs an elastic-perfectly plastic uniaxial material object. Several parameters used in this object are tangent E, yield stress or force in tension FyP, and yield stress or force in compression FyN. By setting $FyP = 0$, the *Elastic-Perfectly Plastic* material can be used to simulate concrete material, as shown in Figure 4.4. The steel material, *Steel01*, is used to construct a uniaxial bilinear steel material object with kinematic hardening and optional isotropic hardening (Mazzoni et al., 2005). This material needs the following parameters: yield strength F_y, initial elastic tangent E, and hardening ratio α. This material object is also illustrated in Figure 4.4.

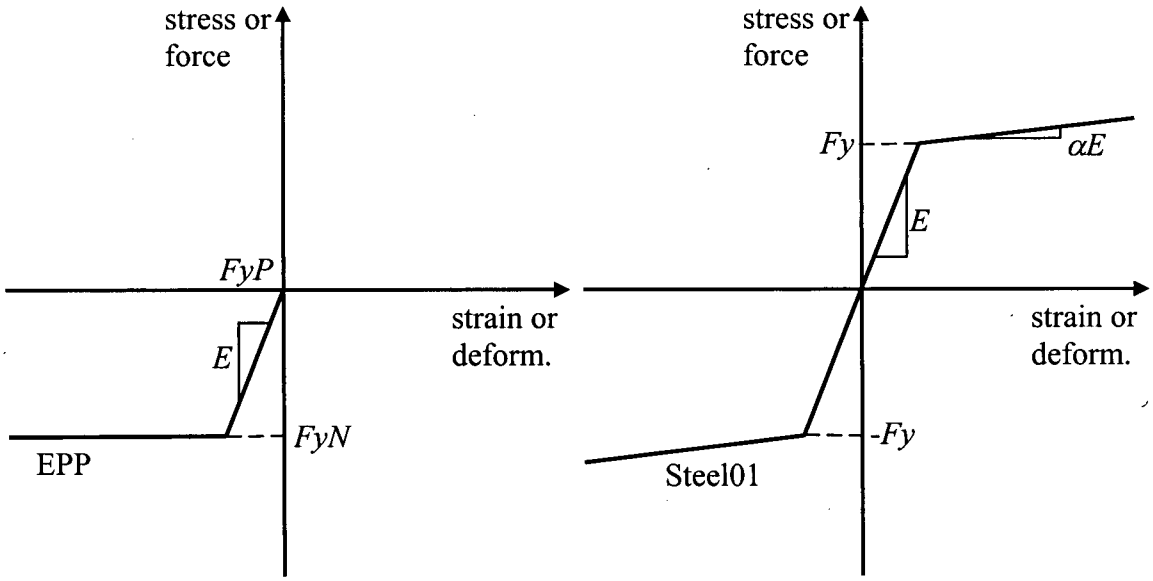

Figure 4.4 Elastic-perfectly plastic material (F_yP =0) and *Steel01* material

4.2 Reliability Analysis

This section focuses on the implementation of the reliability analysis based on the work of Haukaas and Der Kiureghian (2004). A *ReliabilityDomain* object was added to the *Domain* object, while a *ReliabilityAnalysis* object was included in the *Analysis* object of

OpenSees. An overview of some of the objects/classes in the reliability analysis module is shown in Figure 4.5.

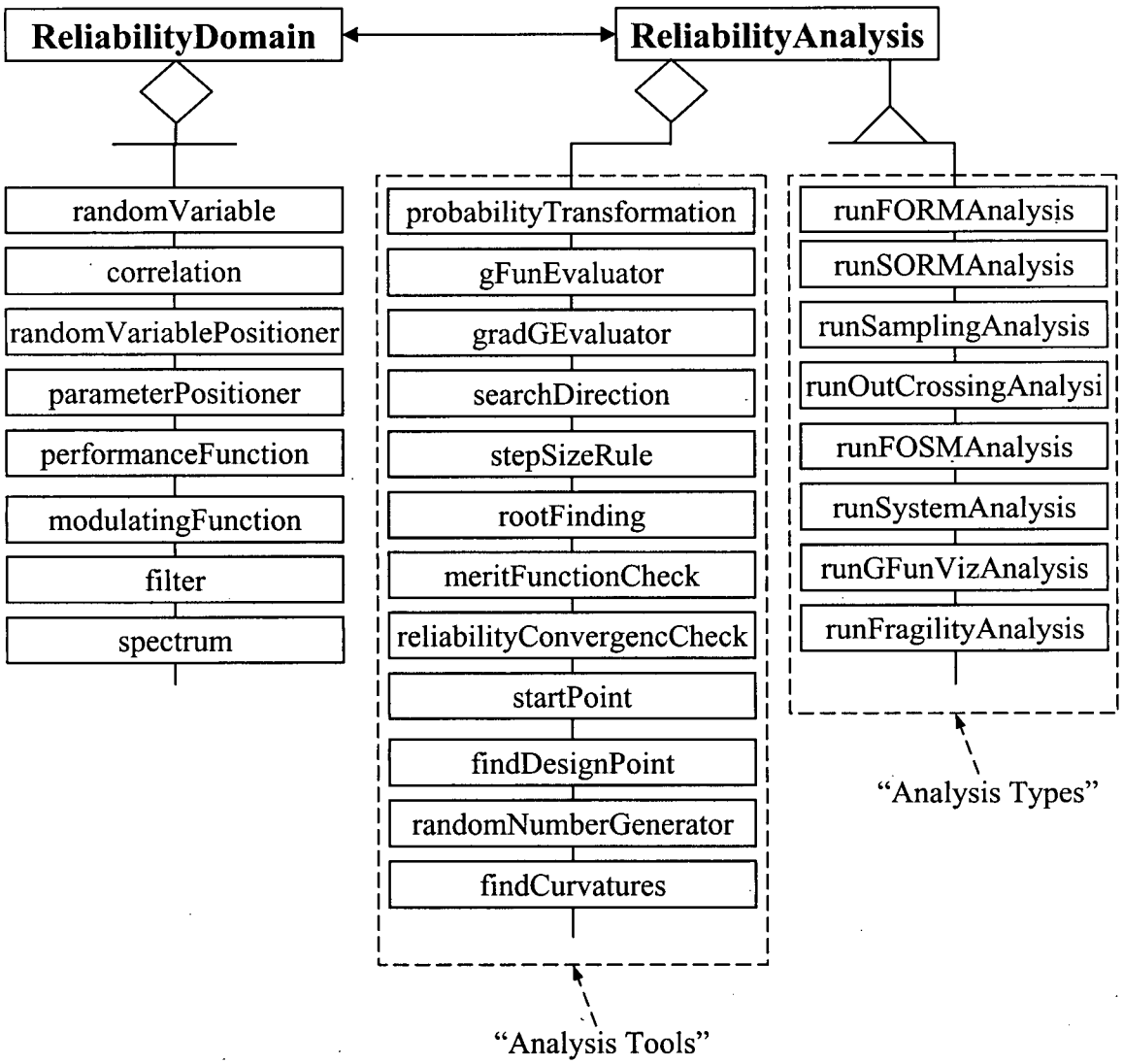

Figure 4.5 Software framework for reliability analysis in OpenSees (Triangle symbol denotes the relationship between base class and subclasses, while the diamond symbol denotes analysis tools)

Three categories of classes are present in Figure 4.5. First, the *ReliabilityDomain* contains model data. A *randomVariable* object creates random variables in several ways, by given the random variable type, the mean, the standard deviation, etc. A *correlation* object specifies the correlation coefficient between random variables. A

randomVariablePositioner object and a *parameterPositioner* object map random variables and parameters into a finite element model. A *performanceFunction* object defines the limit-state function, also called the performance function, using an expression. This expression may include random variables, design variables, and the structural response from the finite element analysis. To create stochastic ground motion input in OpenSees, the discretized random processes is used as a time series object. A *modelatingFunction* object, a *filter* object, and a *spectrum* object are used to simulate this random process.

The next category in Figure 4.5 is the "analysis tools," used for reliability analysis in OpenSees. This framework of analysis tools makes use of the concept of object-oriented programming, which allows the "base classes" to promise feature that is implemented by the "sub-classes." In this manner, several implementations are made available for each of the analysis tools. For instance, a number of sub-classes are available to evaluate the limit-state function, and OpenSees only executes the sub-class that is specified by the user. This illustrates the extensibility feature of OpenSees: new algorithms to perform various analysis tasks are implemented without modifying the software framework.

A *probabilityTransformation* object transfers random variables between the original space and the standard normal space. The Nataf model is applied in the current implementation. A *gFunEvaluator* object computes the value of limit-state functions for a given realization of the random variables. A *gradGEvaluator* object evaluates the gradients of limit-state functions with respect to the random variables. Currently, two alternatives are available: the finite difference method and the direct differentiation method. Both are described in Chapter 5. A *serachDirection* object computes the search direction vector when finding the most probable point (MPP) in the algorithm. Current implementations include the iHLRF algorithm, the Polak-He algorithm, the sequential quadratic programming algorithm, and the gradient projection algorithm. A *stepSizeRule* object obtains an appropriate step size along a search direction using the line search scheme. A simple algorithm uses the fixed step size throughout the search, and the alternative algorithm employs the Armijo line search algorithm. A *rootFinding* object is

used by the gradient projection search algorithm to visualize the limit-state surface. A *meritFunctionCheck* object checks the value of merit function and determines the suitability of a step size. A *reliabilityConvergenceCheck* object checks the convergence when searching for the MPP. One criterion determines the closeness of the MPP to the limit-state surface; another criterion determines how closely the gradient vector points towards the origin in the standard normal space. A *startPoint* object provides the starting point when searching for the MPP; it can also serve as the centre of the sampling density in an importance sampling analysis. Usually, the analysis starts from the mean of the random variables, the origin of the standard normal space, or user-defined values. A *findDesignPoint* object searches for the MPP using a step-by-step search scheme. The search direction is determined by the serachDirection object, and the trial point is determined by computing the step size along this search direction using a line search scheme. A *randomNumberGenerator* object is used in the sampling analysis. The standard library function in the programming language C++ is used in this object. A *findCurvature* object is required in the second-order reliability analysis. It finds the curvatures of the limit-state surface at the MPP.

The third category in Figure 4.5 shows eight analysis types in the reliability module of OpenSees. The users are required to specify some necessary analysis tools before executing these analysis commands. OpenSees prints corresponding information to a file or a computer monitor, thereby allowing the users to monitor the reliability analysis process. Following a successful analysis, OpenSees outputs the results into a user-specified file. In this thesis, the first-order reliability analysis (*runFORMAnalysis*) and the importance sampling analysis (*runSamplingAnalysis*) are employed in case studies in Chapter 7.

Chapter 5 Response Sensitivity for Nonlinear Structures

Response sensitivity is essential in reliability-based design optimization (RBDO) when a gradient-based approach is employed. Reliability analysis requires the gradient of the limit-state function with respect to the random variables when searching for the most probable point. The optimization analysis requires the gradient of the objective function with respect to the design variables to satisfy the first-order optimality conditions. When it comes to the optimization of real-world structures, where the finite element method is employed, the RBDO algorithms require finite element response sensitivities to be available.

Two response sensitivity methods are described in this chapter: the finite difference method (FDM) and the direct differentiation method (DDM). The derivations below follow Haukaas and Der Kiureghian (2004, 2005). The negative effects of the response gradient discontinuity are remedied using two methods, smooth material models and section discretization, which make possible the RBDO for nonlinear structures.

5.1 Finite Difference Method

The FDM consists of perturbing the values of model parameters, re-evaluating structural responses, and finally obtaining a finite difference estimate of the gradient vector. A typical equation of the FDM is

$$\frac{\partial F(\theta)}{\partial \theta} \approx \frac{F(\theta + \Delta\theta) - F(\theta - \Delta\theta)}{2\Delta\theta} \tag{5.1}$$

where θ is the model parameter representing a material, geometry, or load parameter, $F(\theta)$ is the response function evaluated by the finite element analysis, and $\Delta\theta$ is the perturbation. This form of the FDM requires two additional executions of the response function for each parameter.

A simplified FDM is the forward finite difference method, which is similar to Eq. (5.1), but requires only one additional execution of the response function for each parameter. It has the form:

$$\frac{\partial F(\theta)}{\partial \theta} = \frac{F(\theta + \Delta\theta) - F(\theta)}{\Delta\theta} \tag{5.2}$$

Eqs. (5.1) and (5.2) indicate that the FDM is not efficient because of additional executions of the response function for each derivative. In addition, the accuracy of the gradient depends on the selection of the perturbation factor of each parameter, which is a challenging task.

5.2 Direct Differentiation Method

The DDM consists of analytically differentiating the response equations and implementing them inside the finite element code. Therefore, the DDM computes response sensitivities alongside the response estimate without additional response computations. The DDM differentiates the equilibrium with respect to a genetic model parameter θ. For inelastic static analyses, the equilibrium equation requires internal forces $\mathbf{P}_n^{int}$ to equal external loads $\mathbf{P}_n^{ext}$ at a pseudo-time t_n.

$$\mathbf{P}_n^{int}(\mathbf{u}_n) = \mathbf{P}_n^{ext} \tag{5.3}$$

where $\mathbf{P}_n^{int}$ is a function of the displacement vector $\mathbf{u}_n$, and the subscript $_n$ denotes a quantity at time t_n. By differentiating Eq. (5.3) with respect to θ and rearranging, we obtain:

$$\mathbf{K}_n \frac{\partial \mathbf{u}_n}{\partial \theta} = \frac{\partial \mathbf{P}_n^{ext}}{\partial \theta} - \frac{\partial \mathbf{P}_n^{int}}{\partial \theta}\Big|_{\mathbf{u}_n \text{ fixed}} \tag{5.4}$$

where $\mathbf{K}_n = \partial \mathbf{P}_n^{int} / \partial \mathbf{u}_n$ is the algorithmically consistent stiffness matrix, and $\partial \mathbf{u}_n / \partial \theta$ is the desired displacement sensitivity for reliability and optimization analyses. $\partial \mathbf{u}_n / \partial \theta$ is

solved after every convergence of the equilibrium Eq. (5.3). The computation of $\partial \mathbf{u}_n / \partial \theta$ is efficient since Eq. (5.4) is linear with respect to $\partial \mathbf{u}_n / \partial \theta$.

The derivative of internal force for the fixed current displacement appears in Eq. (5.4) and is assembled over all elements based on the strain-displacement matrix **B**, the stress vector $\boldsymbol{\sigma}$, and the element stiffness matrix **k**.

$$\frac{\partial \mathbf{P}_n^{int}}{\partial \theta}\Big|_{\mathbf{u}_n \text{ fixed}} = \bigcup \int_{\Omega_{el}} \left(\frac{\partial \mathbf{B}_n^T}{\partial \theta}\Big|_{\mathbf{u}_n \text{ fixed}} \sigma_n + \mathbf{B}_n^T \mathbf{k}_n \frac{\partial \boldsymbol{\varepsilon}_n}{\partial \theta}\Big|_{\mathbf{u}_n \text{ fixed}} + \mathbf{B}_n^T \mathbf{k}_n \frac{\partial \boldsymbol{\sigma}_n}{\partial \theta}\Big|_{\varepsilon_n \text{ fixed}} \right) \mathbf{dx} \quad (5.5)$$

where $\bigcup$ denotes assembly over all elements, and Ω_{el} denotes the domain of each element. The differentiation of the element integration is also required in the assembly. When parameter θ represents material properties, the derivative of the internal force is assembled from derivatives of stress at each material point for the fixed current strain. Then, Eq. (5.5) can be simplified as follows:

$$\frac{\partial \mathbf{P}_n^{int}}{\partial \theta}\Big|_{\mathbf{u}_n \text{ fixed}} = \bigcup \int_{\Omega_{el}} \mathbf{B}_n^T \frac{\partial \boldsymbol{\sigma}_n}{\partial \theta}\Big|_{\varepsilon_n \text{ fixed}} \mathbf{dx} \quad (5.6)$$

Two important issues are considered in the implementation of the DDM. First, the response sensitivity $\partial \mathbf{u}_n / \partial \theta$ must be solved at each load increment for inelastic materials since sensitivity computations require history variables and their derivatives to be computed and stored at each increment. In each sensitivity computation, the material routine is called twice because the derivative of the stress $\partial \boldsymbol{\sigma} / \partial \theta |_{\varepsilon \text{ fixed}}$ in Eq. (5.6) is conditioned upon fixed strain at the *current* increment only. The first call obtains the derivative of the stress conditioned upon fixed stain, and the second call computes and stores unconditional derivative of the history variables when displacement and strain sensitivities are obtained.

Second, $\partial \boldsymbol{\sigma} / \partial \theta |_{\varepsilon \text{ fixed}}$ must be assembled over all inelastic material points to compute $\partial \mathbf{P}_n^{int} / \partial \theta |_{\mathbf{u}_n \text{ fixed}}$ since all components of the displacement sensitivity $\partial \mathbf{u} / \partial \theta$ are generally not zero. The strain sensitivity $\partial \varepsilon / \partial \theta$ in the finite element domain is not zero, and thus $\partial \boldsymbol{\sigma} / \partial \theta |_{\varepsilon \text{ fixed}}$ is also not zero at all material points after the first load step

because $\partial\varepsilon / \partial\theta$ enters the derivatives of the history variables. For further details, see Haukaas and Der Kiureghian (2004).

5.3 Object-Oriented Implementation in OpenSees

The FDM and the DDM are implemented in OpenSees based on the object-oriented software architecture. Figure 5.1 shows the framework of response sensitivity in OpenSees. The *finiteDifferenceGradGEvaluator* utilizes the *GfunEvaluator* object to compute the value of limit-state functions for perturbed parameters and then to calculate the response gradient using the FDM. The *OpenSeesGradGEvaluator* makes use of the DDM implementation in OpenSees. Two new classes are applied to sensitivity computations: the *sensitivity algorithm* and the *sensitivity integrator*. The *sensitivity algorithm* computes response sensitivities with respect to all desired parameters. Currently, two options are available for the *sensitivity algorithm*: the *computeAtEachStep*

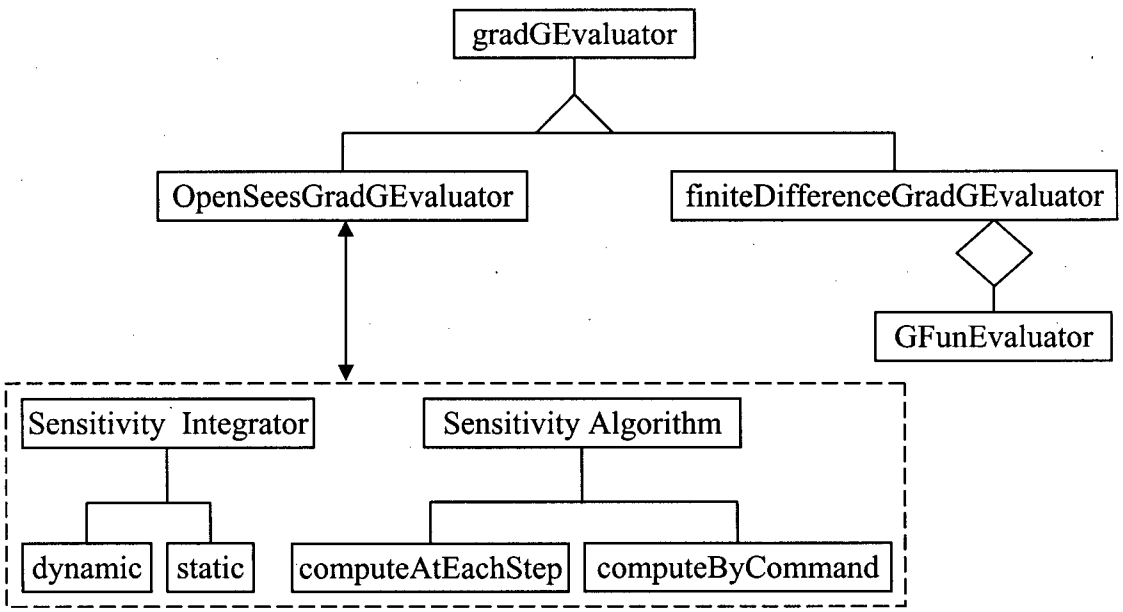

Figure 5.1 The framework of response sensitivity in OpenSees
(Triangle symbol denotes the relationship between base class and subclasses, while the diamond symbol denotes analysis tools)

option is used in all dynamic analyses and all inelastic analyses, while the *computeByCommand* option is used in elastic static analyses. The *sensitivity integrator* assembles the right-hand side of Eq. (5.4) for each parameter for either *static* analysis or *dynamic* analysis.

The DDM computes response sensitivities at each equilibrium state of an inelastic static analysis. After the finite element analysis converges at a pseudo-time t_n, the *sensitivity algorithm* is used to compute response sensitivities and update the tangent matrix. The *sensitivity algorithm* iterates over all parameters for which response sensitivities are desired, and performs the following operations: First, parameter θ in the finite element domain is "activated" to obtain correct contributions from element, section, material, node, and load objects. Second, the *sensitivity integrator* assembles the right-hand side of Eq. (5.4) and collects contributions from the objects of the finite element domain. Next, Eq. (5.4) is solved to obtain the displacement sensitivity $\partial \mathbf{u} / \partial \theta$, and the results are stored in the node objects. Finally, all material objects are called by the *sensitivity integrator* to store unconditional derivatives of history variables through element and section objects. Stain sensitivities are computed based on the displacement sensitivity by using ordinary kinematic equations.

As part of the finite element reliability analysis, the reliability module maps random variables and design variables to the finite element module and receives structural responses and response sensitivities from the finite element module. The mapping procedure updates the value of model parameters in the finite element model each time new random variables and design variables are available. It also identifies parameters to ensure that correct contributions to response sensitivity computations are assembled. Three member functions are used in the classes containing desired model parameters: a member function *identifying* the parameters, a member function *updating* the value of the parameters, and a member function *activating* the parameters for response sensitivity computations.

Two objects are involved in the mapping procedure: the *randomVariablePositioner* object (as part of the reliability analysis) and an object (*e.g.*, a material object) in the

finite element domain. The *randomVariablePositioner* object makes use of the object from the finite element domain as its data member. The detail mapping procedure is described as follows. First, the *setParameter* method in the finite element object creates a link between relevant random variables and the parameter in the finite element object. Then, when the reliability analysis updates the random variables in the finite element model, the update method of the *randomVariablePositioner* object and the *updateParameter* method of the finite element object are called upon update the values of the model parameters using new random variables from the previously created link.

5.4 Continuity of Response Gradients

In the nonlinear finite element analysis it is common to employ material models with sudden transitions from elastic to plastic response. As discussed in Haukaas and Der Kiureghian (2004), this may lead to discontinuities in response sensitivities. This also has an adverse effect on the convergence to the most probable point in reliability analysis. In this thesis we emphasize the potential negative effect of the optimization analysis. In fact, the effect of gradient discontinuities due to sudden transitions from elastic to plastic response is dramatically detrimental to the performance of the optimization algorithm. This is because (1) the proof of convergence of the optimization algorithms requires continuously differentiable limit-state functions; (2) the discontinuities in the gradient may cause the algorithm to stall; and (3) the abrupt changes in the gradient may cause ill-conditioning (even though it is theoretically acceptable) and hence slow convergence to a solution. This leads to the conclusion that the issue of gradient discontinuities is even more important in the RBDO analysis than in the search for the most probable point in the stand-alone reliability analysis. It is stressed that the nonconvergence or slow convergence problems are expected since the assumption of continuous differentiability is violated. In fact, all standard nonlinear programming algorithms will experience difficulties when applied to such inappropriate problems.

In the following subsections, two remedies are introduced to solve discontinuity problems in response sensitivities.

5.4.1 Smooth Material Model

To avoid gradient discontinuities at yielding, the original bi-linear material model is replaced with a smoothed version developed by Haukaas and Der Kiureghian (2004). The bi-linear material model contains an elastic range and a plastic range. Unloading is assumed to be elastic. The tangent stiffness is E in the elastic range and αE in the plastic range, where $0 < \alpha < 1$. The yield strength Fy identifies at the transition between elastic and plastic states. In the smooth material model, a circular segment is used to smooth the transition between the elastic and plastic response state. The tangent stiffness of the

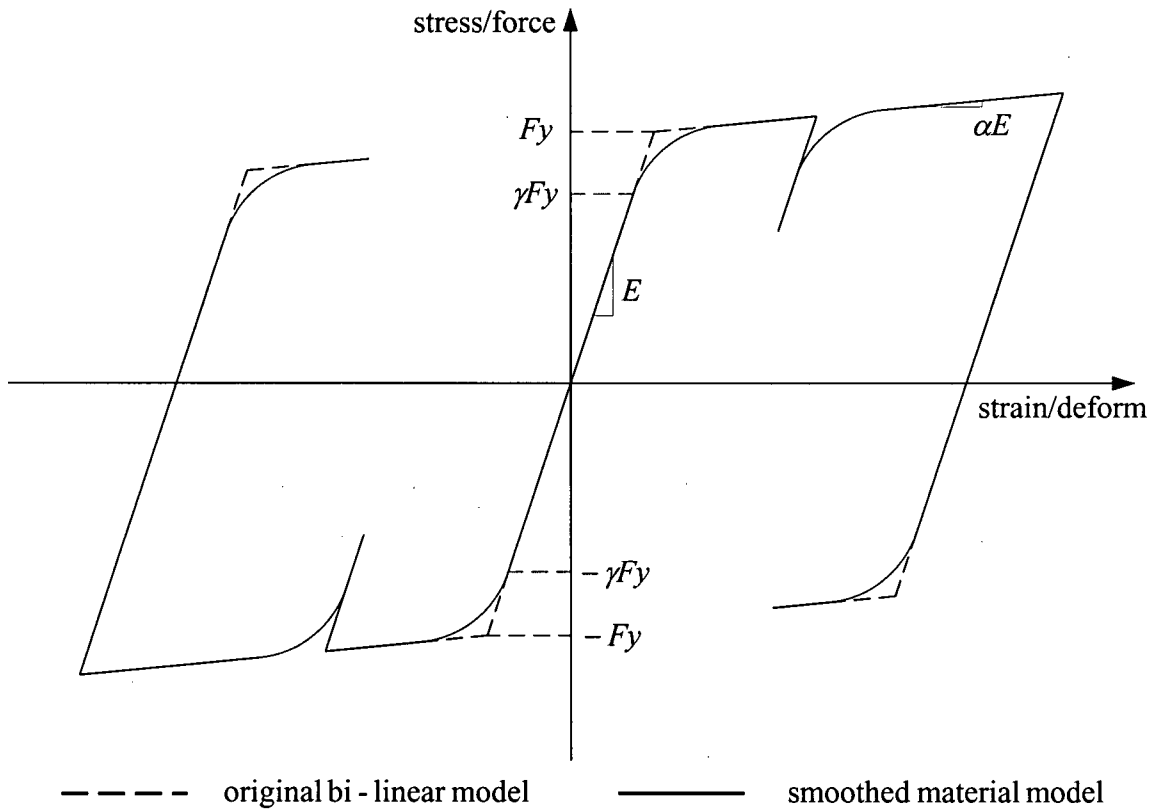

Figure 5.2 Bi-linear steel material model smoothed with circular segment

circular segment coincides with those of the elastic and plastic ranges at intersection points. The smoothed stress-stain curve and its tangent stiffness are thus continuous. The circular segment starts from $\gamma \cdot Fy$ in the stress-strain curve, where the parameter is $0 < \gamma < 1$. Figure 5.2 illustrates how the bi-linear steel material model is smoothed with a circular segment. Haukaas and Der Kiureghian (2004) also show that the overall response does not change significantly as a result of smoothing. They recommend the selection of parameter $\gamma \geq 0.8$ to avoid results that differ significantly from those obtained with the bi-linear model.

It is demonstrated in Haukaas and Der Kiureghian (2004) that the smooth material model leads to continuity in response sensitivities. To compute response sensitivities using the DDM approach, Haukaas and Der Kiureghian (2004) differentiate the DDM equations for the smooth material model and implement them in OpenSees. They also present several examples to show that the smooth material model successfully avoids the response sensitivity discontinuity problem in the reliability analysis.

5.4.2 Section Discretization Scheme

The section discretization scheme discretizes the element cross-section into smaller regions or fibres. The uniaxial material response for each fibre is integrated to produce approximately smooth behaviour. The use of this section discretization scheme makes the nonlinear structural response "approximately continuously differentiable" to meet the requirement of the RBDO algorithms.

The section discretization scheme takes advantage of the *fibre* section object in OpenSees. The fibre section is ideal for defining a reinforced concrete section: 1-2 top and bottom fibres of the concrete cover using normal strength concrete, 10-20 side fibres of the concrete cover using normal strength concrete, 10-20 fibres of the concrete core using higher strength confined concrete, and several layers of reinforced bars. Examples of such fibre section are shown in Figure 4.3.

The section discretization scheme makes the structural response approximately continuously differentiable. During the finite element analysis, individual fibres pass the yield point and enter the plastic range one by one. Each fibre is only a small part of the whole cross section. The discontinuity of one fibre affects the property of the whole section, but the effect becomes progressively smaller as the number of fibres in the whole section increases. Thus, theoretically, the discretized cross section possesses continuous properties when the section is discretized by an infinite number of fibres. Practical studies in this work suggest that approximately 20 fibres are sufficient to avoid convergence problems. It is emphasized, however, that a smooth material model must be employed for the reinforcing steel, since this is represented by a single fibre layer that is not affected by an overall increase in the number of fibres in the cross-section.

Chapter 6 Implementation of Reliability-Based Design Optimization

Two decoupling approaches for reliability-based design optimization (RBDO) problems are implemented in this chapter: a decoupled sequential approach using the method of outer approximations (DSA-MOOA) and a simplified decoupled sequential approach (DSA-S). The two approaches employ the same problem reformulation, which is presented below. The algorithms that are used in the DSA-MOOA approach and the DSA-S approach are also detailed. Finally, OpenSees is extended in light of these optimization capabilities through object-oriented programming.

6.1 Problem Reformulation

The RBDO problem in Eq. (1.2) minimizes the initial design cost plus the expected cost of failure subject to reliability and structural constraints. Let $\mathbf{x}$ be the vector of design variables. Then, this problem takes the form (see Chapter 1)

$$\mathbf{x}^* = \arg\min\left\{ c_0(\mathbf{x}) + c_f(\mathbf{x}) p_f(\mathbf{x}) \;\middle|\; \mathbf{f}(\mathbf{x}) \le \mathbf{0},\ p_f(\mathbf{x}) \le \hat{p}_f \right\} \tag{6.1}$$

where $c_0(\mathbf{x})$ is the initial cost of the design, $c_f(\mathbf{x})$ is the present cost of future failure, $p_f(\mathbf{x})$ denotes the probability of failure for one failure event, $\mathbf{f}(\mathbf{x})$ are deterministic constraints, and $\hat{p}_f$ is the upper bound on the probability of failure.

The solution algorithm for Eq. (6.1) requires the functions involved, $c_0(\mathbf{x})$, $c_f(\mathbf{x})$, and $\mathbf{f}(\mathbf{x})$, to be continuous, and the constraint set $\mathbf{f}(\mathbf{x})$ to be closed and bounded. Since the failure probability $p_f(\mathbf{x})$ is involved in both the objective and constraint functions, the failure probability is also required to be continuous. Royset et al. (2002) have proven that this is the case in realistic design problems.

The problem in Eq. (6.1) is computationally difficult since the failure probability, which depends on the design variables, is defined in terms of a high-dimensional integral over the domain of random variables. Royset et al. (2002) replace the failure probability $p_f(\mathbf{x})$ with parameter a, which is updated during the optimization analysis to develop a computationally feasible problem. This parameter is included in an augmented design vector $\bar{\mathbf{x}} = (\mathbf{x}, a)$ and appears as an added constraint of a reformulated optimization problem, which reads:

$$\bar{\mathbf{x}} = \arg\min\left\{ c_0(\mathbf{x}) + c_f(\mathbf{x})a \;\middle|\; \mathbf{f}(\mathbf{x}) \le 0,\, p_f(\mathbf{x}) = a,\, 0 \le a \le \hat{p}_f \right\} \tag{6.2}$$

Royset et al. (2002) have proven that Eqs. (6.1) and (6.2) have identical global optimal solutions when some assumptions are satisfied.

Because the gradient of the true failure probability is unavailable, it is problematic that the failure probability still appears among the constraints. This is addressed by making use of concepts from the first-order reliability method (FORM). As outlined in Chapter 2, the FORM estimate of the failure probability is $p_f = \Phi(-\beta)$, where the reliability index β is the minimum distance from the origin to the limit-state surface g=0. To this end, the equality constraint $p_f(\mathbf{x}) = a$ in Eq. (6.2) is replaced by constraint $\psi = 0$, where ψ is the negative value of the minimum of limit-state function g within a hyper-sphere of radius $-\Phi^{-1}(a)$. Hence, the problem is now reformulated into the form

$$\bar{\mathbf{x}} = \arg\min\left\{ c_0(\mathbf{x}) + c_f(\mathbf{x})a \;\middle|\; f_j(\mathbf{x}) \le 0, \psi(\bar{\mathbf{x}}) = 0, 0 \le a \le \hat{p} \right\} \tag{6.3}$$

where

$$\psi(\bar{\mathbf{x}}) = -\min_{\mathbf{u} \in \mathbf{B}(\mathbf{0}, r)} \{ g(\mathbf{x}, \mathbf{u}) \} \tag{6.4}$$

with $B(\mathbf{0},r)=\{\mathbf{u} \mid \|\mathbf{u}\| \le r\}$, where $r=-\Phi^{-1}(a)$.[1] It should be stressed that this reformulation is motivated by the desire to cast the optimization problem in a semi-infinite form, thereby allowing it to be solved using the method of outer approximations (DSA-MOOA). This is because the constraint $\psi = 0$ in fact represents an infinite number of constraints, one for each point within the hyper-sphere. In this thesis we also demonstrate that a simplified approach (DSA-S) can be used to solve the problem by only including one constraint to enforce $\psi = 0$.

If a FORM approximation of the failure probability is acceptable, then the reformulation of the constraint $p_f(\mathbf{x}) = a$ in terms of the function ψ is correct should the parameter r be equal to $-\Phi^{-1}(a)$. However, when the limit-state function $g(\mathbf{x},\mathbf{u})$ is prescribed in terms of response quantities from a nonlinear finite element analysis, then the limit-state function is nonlinear. To account for such nonlinearity, a correction factor t is introduced: $r=-\Phi^{-1}(a)t$. The start value of the auxiliary variable t is unity, and it is updated during the optimization analysis.

Eq. (6.3) is a semi-infinite optimization problem with equality constraints. As described in Chapter 3, the available algorithms address problems with inequality constraints. Hence, a more suitable problem formulation is obtained by converting the equality constraint into an inequality constraint:

$$\bar{\mathbf{x}} = \arg\min\left\{c_0(\mathbf{x}) + c_f(\mathbf{x})a \,\middle|\, \mathbf{f}(\mathbf{x}) \le 0, \psi(\bar{\mathbf{x}}) \le 0, 0 \le a \le \hat{p}_f\right\} \tag{6.5}$$

Royset et al. (2002) have proven that replacing the equality constraint in Eq. (6.3) by an inequality constraint does not alter the solution. This proof assumes that the failure cost is positive, and that the origin in the standard normal space is in the safe domain. The

---

[1] The solution algorithm requires that the solution domain of Eq. (6.3) remain fixed. This is obviously not the case because r varies in the optimization process. In the computer implementation, this problem is solved by applying the $\mathbf{u} = r\hat{\mathbf{u}}$ transformation, where $\hat{\mathbf{u}}$ is the solution domain that remains a ball of unit radius, namely $\hat{\mathbf{u}} \in B(\mathbf{0},1)$.

former assumption is trivially satisfied, and the latter one is generally satisfied due to the high reliability of structures.

Eq. (6.5) is the final reformulation of the original problem. If the limit-state function is linear or if the FORM approximation of the failure probability is acceptable, then t is set at 1 and the reformulated problem in Eq. (6.5) has the same solution as the original problem in Eq. (6.1). This is proven by Royset et al. (2002). Moreover, the method of outer approximation (MOOA) algorithm to solve the semi-infinite problem in Eq. (6.5) has convergence proofs (Polak, 1997). Hence, we are guaranteed to find a converged solution for our final approximate problem in Eq. (6.5).

However, the first-order approximation of the reliability problem may be a poor assumption when the limit-state function is highly nonlinear. That is, in nonlinear finite element reliability problems, the parameter a from the first-order approximation does not equal the probability of failure $\tilde{p}_f(\mathbf{x})$ from a more precise reliability analysis (for example, importance sampling). To be able to deal with these cases, we solve the final approximate problem through updating the value of the parameter t. The parameter t starts from unity and is updated during the optimization analysis to account for the nonlinearity in the limit-state function. In this manner, approximate solutions are obtained with increasing accuracy as the algorithm proceeds. Specifically, the parameter t is updated by multiplying with the correction factor $\Phi^{-1}(a)/\Phi^{-1}(\tilde{p}_f(\mathbf{x}))$. The philosophy behind this update is that if $\tilde{p}_f(\mathbf{x}) > a$, then the constraint $\psi \le 0$ in the final approximate problem allows the limit-state surface $\{\mathbf{u} \mid g(\mathbf{x},\mathbf{u}) = 0\}$ to come too close to the origin in the **u**-space, thus requiring the radius of the ball associated with ψ to be increased. The increase of the ball radius is obtained by increasing t. If $\tilde{p}_f(\mathbf{x}) < a$, then the limit-state surface is required to be too far away from the origin in the **u**-space by the constraint $\psi \le 0$, and the size of the ball must be reduced (*i.e.*, t is reduced).

6.2 DSA-MOOA Approach

In this thesis we employ solution algorithms that are termed "decoupled" and "sequential." The justification for the decoupled characterization is that any reliability method can be used to obtain a more precise estimate of the failure probability than the FORM analysis. The justification for the sequential characterization is that the optimization analysis in Eq. (6.5) and the reliability analysis in Eq. (6.4) are solved repeatedly and in sequence to address the bi-level problem in Eq. (6.1). The reliability constraint is updated for each optimization analysis in Eq. (6.5). It is stressed that the decoupled approach allows flexibility in the choice of optimization algorithm and reliability computation method.

In this section we present the implementation of the DSA-MOOA approach developed by Kirjner-Neto et al. (1998) and Royset et al. (2002). It makes use of the problem formulation in Eq. (6.5). Figure 6.1 presents a flow chart of the DSA-MOOA algorithm, which consists of iterations at several levels. Upon the initialization of design $\mathbf{x}$ and parameters a and t, the top level iteration includes three tasks:

A1. *Update design vector* $\mathbf{x}$ *and auxiliary variable* a.

A2. *Compute failure probability* $\tilde{p}_f(\mathbf{x})$ *using a method that is more accurate than FORM.*

A3. *Update parameter* t.

This procedure is repeated until the design vector $\mathbf{x}$, the auxiliary design variable a, and the parameter t stabilize. Usually, 5 to 15 iterations (A1 to A3) are needed to reach a satisfactory solution.

Task A1 consists of obtaining updated values of $\mathbf{x}$ and a by solving the semi-infinite optimization problem in Eq. (6.5). The user can choose any suitable algorithm for this purpose. The MOOA algorithm is employed in the current OpenSees implementation. In this algorithm, the ball $B(\mathbf{0},1)$ is discretized into a finite number of points, $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_N$, in the following manner: approximate the constraint $\psi \leq 0$ by N constraints

$g(\mathbf{x},-\Phi^{-1}(a)\cdot t\cdot \mathbf{u}_j)\geq 0,\ j=1,2,...,N$ and solve the resulting standard nonlinear inequality constrained problem. In task A1, the MOOA algorithm consists of three iteratively performed tasks (B1-B3):

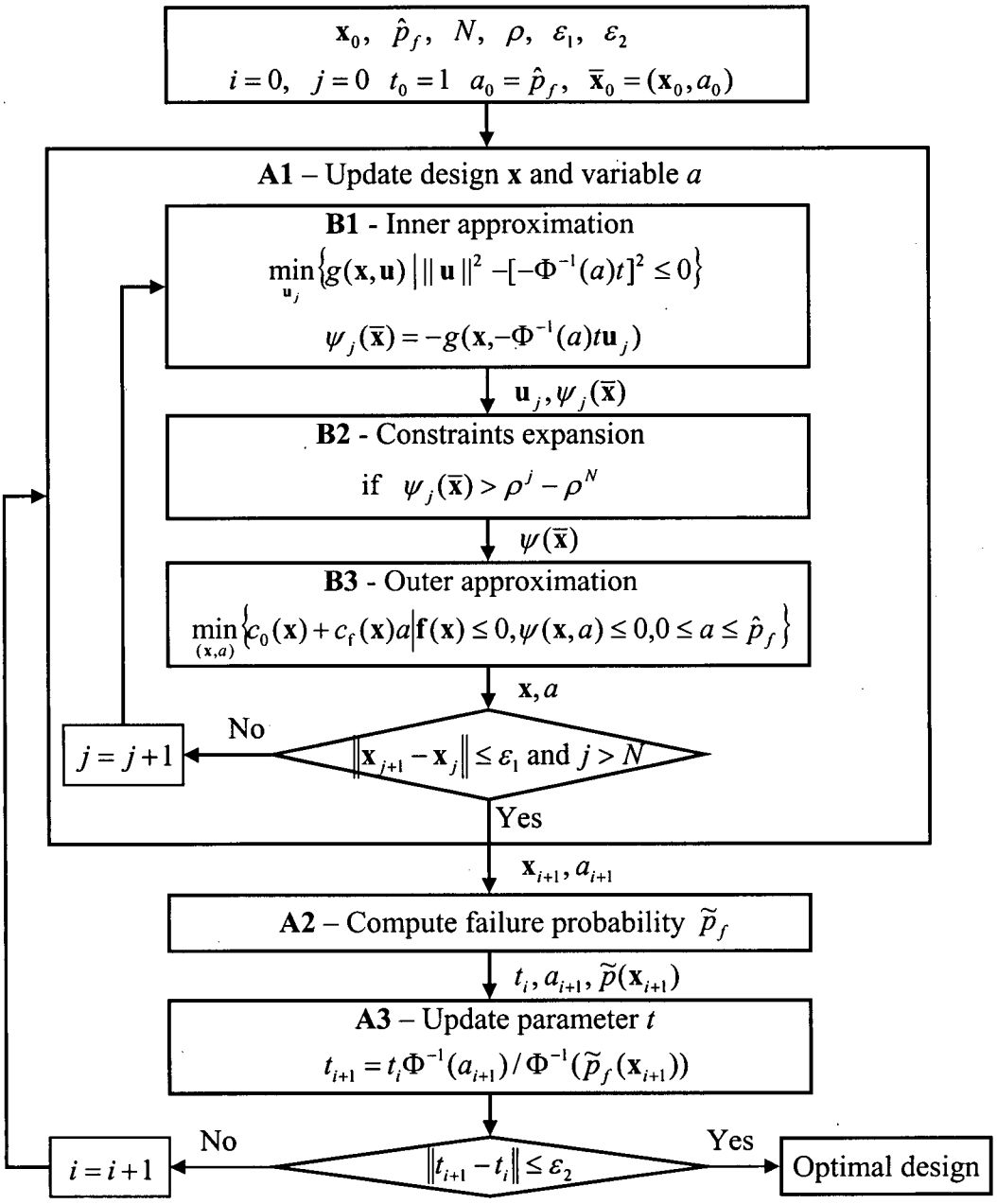

Figure 6.1 Flow chart of DSA-MOOA approach

B1. Inner approximation: *Obtain the minimum value of the limit-state function within the ball* $\mathrm{B}(\mathbf{0}, -\Phi^{-1}(a)\,t)$ *and the corresponding random vector* $\mathbf{u}_j$ *using the Polak-He algorithm. In this thesis, the negative value of the limit-state function at the vector* $\mathbf{u}_j$ *is denoted as* ψ. *Terminate the Polak-He algorithm when tolerance* $-\sigma_N$ *is satisfied. Here,* $\sigma_N = 0.1/N^2$ *is a user-defined series,* $\sigma_N \to 0$ *as* $N \to \infty$.

B2. Constraints expansion: *Update the constraints by accumulatively storing solutions* $\boldsymbol{u}_j$ *from task B1 for which solution* ψ_j *exceeds* $\rho^j - \rho^N$, *where* ρ *is a user-defined parameter usually set at 0.5. In this manner, the number of constraints represented by* $\psi \le 0$ *evolves during these iterations. We have observed that these constraints are simply a collection of points* $\boldsymbol{u}_j$ *for which the limit-state function is required to stay positive in task B3.*

B3. Outer approximation: *Solve the constrained optimization problem in Eq. (6.5) using the Polak-He algorithm. The number of constraints in this problem is equal to the number of structural constraints, plus the N constraints added by the previous item and the single constraint* $a \le \hat{p}_f$. *The result is a new augmented design vector* $\bar{\mathbf{x}} = (\mathbf{x}, a)$. *According to proofs presented by Polak (1997) for the MOOA algorithm, an "exact" solution is found if the discretization number N approaches infinity.*

Tasks B1, B2, and B3 are repeated until the optimality conditions are satisfied according to a user-defined precision tolerance. Typically, 75 to 150 iterations are required to find the optimal solution. These tasks are described in further detail in subsections 6.2.1-6.2.3, in which we focus on the connections between the particular problems discussed in this chapter and the general algorithms discussed in Chapter 3.

One important advantage of DSA-MOOA approach is reiterated, namely that the reliability and optimization calculations are decoupled, thus allowing flexibility in the choice of optimization algorithm in task A1 and reliability computation method in task A2. In addition to the MOOA algorithm, Polak (1997) provides a *pre-defined discretization scheme* to solve the semi-infinite problem in task A1. Similarly, the user is

free to estimate the failure probability in task A2 by selecting a suitable computational reliability method such as the second-order reliability method, Monte Carlo sampling, or importance sampling. This selection depends on the desired accuracy in satisfying the probability constraint $\tilde{p}_f(\mathbf{x}) < \hat{p}_f$. For example, an importance-sampling scheme with a required 1-2% coefficient of variation of the sampling result is an appropriate choice if the user wants a high degree of confidence that the reliability constraint is satisfied.

Typically, users require the structural reliability to be very high. In effect, the failure probability $p_f(\mathbf{x})$ is very small. The DSA-MOOA approach may experience numerical difficulties caused by the potential difference in orders of magnitude between a and the other design variables $\mathbf{x}$. For this reason, in our implementation the parameter $b = -\Phi^{-1}(a)$ is used in place of a. With reference to Eq. (2.9), we note that b is a substitute for the reliability index β, in the same way that a is a substitute for the failure probability $p_f(\mathbf{x})$. In conclusion, the optimization in DSA-MOOA approach is over the design vector ($\mathbf{x}$, b).

6.2.1 B1 - Inner Approximation

Given design $\mathbf{x}$ and parameters a and t, task B1 solves the following reliability problem:

$$\mathbf{u}_j = \mathbf{u}^* = \arg\min\left\{ g(\mathbf{x},\mathbf{u}) \,\middle|\, \|\mathbf{u}\|^2 - r^2 \le 0 \right\} \tag{6.6}$$

where $r = -\Phi^{-1}(a) \cdot t$. This problem is equivalent to Eq. (6.4). The result is a random vector $\mathbf{u}^*$, or $\mathbf{u}_j$ (for j^{th} discretization point). The corresponding constraint $\psi_j = -g(\mathbf{x}, r\mathbf{u}_j)$ is the minimum of the limit-state function within a ball of radius r.

The Polak-He algorithm described in Chapter 3 is used to solve Eq. (6.6), an inequality constraint optimization problem with a single constraint. The fact that there is only a single constraint simplifies the optimization process. The values and gradients of functions F and f in Eq. (3.1) are

$$F(\mathbf{x},\mathbf{u}) = g(\mathbf{x},\mathbf{u}), \qquad \nabla F(\mathbf{x},\mathbf{u}) = \nabla g(\mathbf{x},\mathbf{u}) \tag{6.7}$$

$$f(\mathbf{x},\mathbf{u}) = f_1(\mathbf{x},\mathbf{u}) = \|\mathbf{u}\|^2 - r^2, \qquad \nabla f_1(\mathbf{x},\mathbf{u}) = 2\mathbf{u} \tag{6.8}$$

The first step searches for the direction vector $\mathbf{h}_i$ by solving a quadratic optimization problem $\theta_i(\mathbf{x})$ in Eq. (3.14), subject to linear constraint $\mu_0 + \mu_1 = 1$. By setting $\mu_0 = 1 - \mu_1$ and substituting Eqs. (6.7) and (6.8) into Eq. (3.14), this sub-optimization problem can be simplified as

$$\begin{aligned}\theta_i(\mathbf{x}) = -\min\{&\gamma f(\mathbf{x},\mathbf{u}_i) - \mu_1[\gamma f(\mathbf{x},\mathbf{u}_i) - f(\mathbf{x},\mathbf{u}_i)_+ + f_1(\mathbf{x},\mathbf{u}_i)] \\ &+ \frac{1}{2\delta}\|\nabla g(\mathbf{x},\mathbf{u}_i) + \mu_1[2\mathbf{u}_i - \nabla g(\mathbf{x},\mathbf{u}_i)]\|^2\}\end{aligned} \tag{6.9}$$

Eq. (6.9) can be solved by $\nabla\theta_i / \nabla\mu_1 = 0$, and has the following solution:

$$\mu_1^* = \frac{[\gamma f(\mathbf{x},\mathbf{u}_i) - f(\mathbf{x},\mathbf{u}_i)_+ + f_1(\mathbf{x},\mathbf{u}_i)] - \frac{1}{\delta}\nabla g(\mathbf{x},\mathbf{u}_i)[2\mathbf{u}_i - \nabla g(\mathbf{x},\mathbf{u}_i)]}{\frac{1}{\delta}[2\mathbf{u}_i - \nabla g(\mathbf{x},\mathbf{u}_i)]^2} \tag{6.10}$$

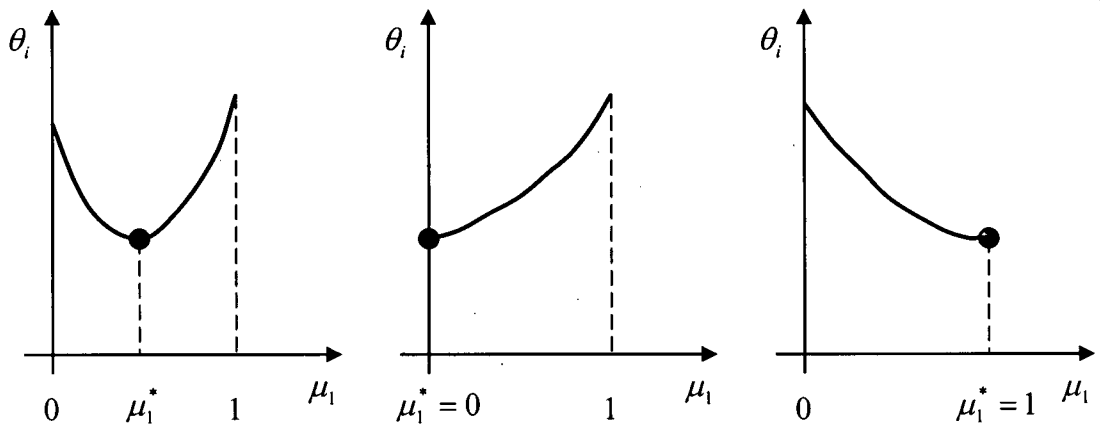

Figure 6.2 μ_1 solutions for inner approximation using the Polak-He algorithm

whenever the right-hand side of Eq. (6.10) has a value in [0,1]. Otherwise, the solution of Eq. (6.9) is either $\mu_1^* = 0$ or $\mu_1^* = 1$, whichever yields the lowest value for the objective

function in Eq. (6.9). Three types of solutions are illustrated in Figure 6.2. Finally, the search direction is

$$\mathbf{h}_i = -\frac{1}{\delta}\left[(1-\mu_1^*)\cdot\nabla g(\mathbf{x},\mathbf{u}_i)+2\mu_1^*\mathbf{u}_i\right] \tag{6.11}$$

The second step finds an appropriate step size λ_i along the search direction $\mathbf{h}_i$ using the Armijo rule. Then, the random vector $\mathbf{u}_{i+1}$ is replaced by $\mathbf{u}_{i+1} = \mathbf{u}_i + \lambda_i \mathbf{h}_i$ and is used as the input data for the next iteration.

For task B1, the Polak-He algorithm requires an infinite number of iterations before it converges to the optimal solution $\mathbf{u}^*$. However, we terminate the Polak-He algorithm when user-defined tolerance $-\sigma_N = -0.1/N^2$ is satisfied. N starts from 1 and goes to infinity, so σ_N starts from the larger tolerance 0.1 and goes to 0 as N increases. Since high accuracy is only needed when approaching the design point, a high tolerance in the beginning of task B1 is acceptable and can save computational time.

6.2.2 B2 - Constraints Expansion

Task B2 first collects the $\mathbf{u}_j$ and $\psi_j(\bar{\mathbf{x}})$ from the inner approximation (task B1) into a matrix. For N discretization points, we have an N-column matrix, in which each column contains random variables $\mathbf{u}_j$ and $\psi_j(\bar{\mathbf{x}})$ of the form

$$\begin{bmatrix} \mathbf{u}_1 & \cdots & \mathbf{u}_j & \cdots & \mathbf{u}_N \\ \psi_1(\bar{\mathbf{x}}) & \cdots & \psi_j(\bar{\mathbf{x}}) & \cdots & \psi_N(\bar{\mathbf{x}}) \end{bmatrix} \tag{6.12}$$

Second, task B2 assembles the reliability constraints set $\psi(\bar{\mathbf{x}})$, which includes the constraint $\psi_0(\bar{\mathbf{x}})$ at the origin ($\mathbf{u}_0 = 0$) in the whole assembling procedure. The reason to include the constraint at the origin is to make sure $\psi_0(\bar{\mathbf{x}}) = -g(\mathbf{x}, \mathbf{0}) < 0$, *i.e.*, $g(\mathbf{x}, \mathbf{0}) > 0$, which is the requirement of problem reformulation in subsection 6.1. Task B2 updates reliability constraints by accumulatively storing solutions $\mathbf{u}_j$ when $\psi_j(\bar{\mathbf{x}})$

exceeds $\rho^j - \rho^N$. $\rho \in (0, 1)$ is a user-defined parameter set at 0.5 in the current implementation. $\rho^j - \rho^N$ is a positive series and approaches zero when j approaches N. Finally, the reliability constraint set $\psi(\bar{\mathbf{x}})$ has the following form:

$$\psi(\bar{\mathbf{x}}) = \begin{bmatrix} \mathbf{u}_0 = 0 & \mathbf{u}_1 & \cdots & \mathbf{u}_j & \cdots & \mathbf{u}_N \\ \psi_0(\bar{\mathbf{x}}) < 0 & \psi_1(\bar{\mathbf{x}}) > \rho^1 - \rho^N & \cdots & \psi_j(\bar{\mathbf{x}}) > \rho^j - \rho^N & \cdots & \psi_N(\bar{\mathbf{x}}) > 0 \end{bmatrix} \quad (6.13)$$

In this manner, the number of constraints represented by $\psi(\bar{\mathbf{x}})$ evolves with the increase in the number of discretization points. These constraints are simply a collection

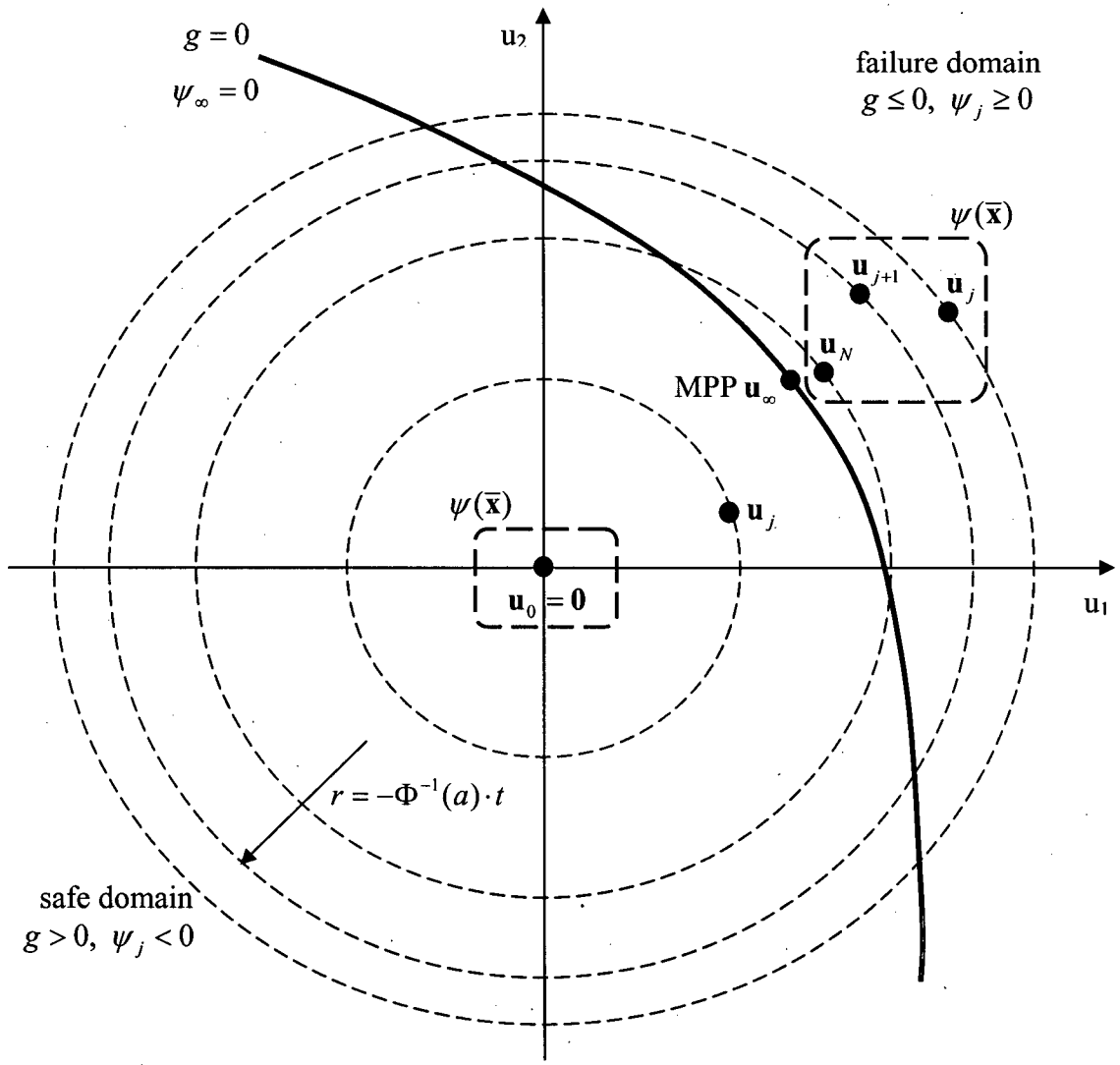

Figure 6.3 Reliability constraints set $\psi(\bar{\mathbf{x}})$

of points $\mathbf{u}_j$ for which the limit-state function is negative in task B2 but is required to stay positive in task B3. Hence, the result of task B2 is an expanded constraints set $\psi(\bar{\mathbf{x}})$, which enters task B3 for outer approximation. From the above description, we know that the number of columns of matrix $\psi(\bar{\mathbf{x}})$ is less than or equal to N+1. Figure 6.3 illustrates the procedure used to assemble reliability constraints set $\psi(\bar{\mathbf{x}})$ by collecting all of the points in the failure domain in the standard normal space.

6.2.3 B3 - Outer Approximation

Given the objective function $c_0(\mathbf{x}) + c_f(\mathbf{x})a$, m deterministic constraints $\mathbf{f}(\mathbf{x}) \le 0$, n reliability constraints $\psi(\bar{\mathbf{x}}) \le 0$, and an extra constraint $a \le \hat{p}_f$, task B3 solves Eq. (6.5) and updates design $\mathbf{x}$ and parameter a. This task is also solved using the Polak-He algorithm. As opposed to task B1, the number of constraints in task B3 is greater than one. In fact, the number of constraints increases during the analysis. According to Polak (1997), in the Polak-He algorithm a quadratic sub-optimization problem with linear constraints must be solved to obtain the search direction $\mathbf{h}_i$. This is currently addressed by linking the quadratic programming software LSSOL (Gill et al., 1986) to OpenSees.

For a fixed number (q=m+n+1) of constraints, Eq. (6.5) is an inequality-constrained problem. If the Polak-He algorithm is applied to task B3, functions F and f in Eq. (3.1) have the following form:

$$\begin{array}{ll} \text{Objetive function} & F(\bar{\mathbf{x}}) = c_0(\mathbf{x}) + c_f(\mathbf{x})a \\ \text{Deterministic constraints} & f_1(\bar{\mathbf{x}}) = f_1(\mathbf{x}), \cdots, f_m(\bar{\mathbf{x}}) = f_m(\mathbf{x}) \\ \text{Reliability constraints} & f_{m+1}(\bar{\mathbf{x}}) = \psi_0(\bar{\mathbf{x}}), f_{m+2}(\bar{\mathbf{x}}) = \psi_1(\bar{\mathbf{x}}), \cdots, f_{m+n}(\bar{\mathbf{x}}) = \psi_{n-1}(\bar{\mathbf{x}}) \\ \text{Extra constraint} & f_{m+n+1}(\bar{\mathbf{x}}) = a - \hat{p}_f \end{array} \tag{6.14}$$

The first step searches for the direction vector $\mathbf{h}_i$ by solving a quadratic optimization problem

$$\theta_i = -\min\left\{\mu_0 \gamma\, f(\bar{\mathbf{x}}_i)_+ + \mu_j\left[f(\bar{\mathbf{x}}_i)_+ - \mathbf{f}(\bar{\mathbf{x}}_i)\right] + \frac{1}{2\delta}\left\|\mu_0 \nabla F(\bar{\mathbf{x}}_i) + \mu_j \nabla \mathbf{f}(\bar{\mathbf{x}}_i)\right\|^2\right\} \quad (6.15)$$

where $f(\bar{\mathbf{x}}) = \max_{j\in\{1,2,\ldots,q\}} f_j(\bar{\mathbf{x}})$, $f(\bar{\mathbf{x}})_+ = \max\{0, f(\bar{\mathbf{x}})\}$. Note that θ_i is a quadratic problem dependent on Lagrange multipliers $(\mu_0, \mu_1, \cdots, \mu_q)$ subject to linear constraints $\mu_0 + \mu_1 + \cdots + \mu_q = 1$. Eq. (6.15) cannot be solved in the same way as Eq. (6.9) in the inner approximation, so it is instead solved using the quadratic programming software LSSOL (Gill et al., 1986) or the Matlab OPTM toolbox (Matlab, 1999). LSSOL is used in the current implementation of OpenSees since it is a collection of FORTRN 77 subroutines and is faster than the equivalent "Quadprog.m" in the Matlab code. In this thesis, the constrained least-squares problem "LS2 Type Objective Function" is used and stated in the following form:

$$\begin{aligned} &\underset{\boldsymbol{\mu}}{\text{minimize}} \quad F(\boldsymbol{\mu}) = \mathbf{g}^T\boldsymbol{\mu} + \frac{1}{2}\left\|\mathbf{b} - \mathbf{G}\boldsymbol{\mu}\right\|^2 \\ &\text{subject to} \quad \mathbf{L} \le \begin{Bmatrix} \boldsymbol{\mu} \\ \mathbf{C}\cdot\boldsymbol{\mu} \end{Bmatrix} \le \mathbf{U} \end{aligned} \quad (6.16)$$

where $\boldsymbol{\mu} = [\mu_0,\ \mu_1,\ \cdots, \mu_q]^T$, general constraints $\mathbf{C} = [1,\ 1,\ \cdots, 1]_{q+1}$, vectors $\mathbf{L}$ and $\mathbf{U}$ are the lower and upper bounds for all the variables and general constraints, respectively: $\mathbf{L} = [0,\ 0,\ \cdots, 1]^T_{q+1}$ and $\mathbf{U} = [1,\ 1,\ \cdots, 1]^T_{q+1}$. The constraints in Eq. (6.16) are equivalent to the constraints in Eq. (6.15):

$$\begin{aligned} &0 \le \mu_j \le 1, \quad j = 0, 1, \cdots, q \\ &\mu_0 + \mu_1 + \cdots + \mu_q = 1 \end{aligned} \quad (6.17)$$

We refer to $\mathbf{G}$ as the least-square matrix and to vector $\mathbf{b}$ as the vector of observations, here $\mathbf{b} = 0$. The objective function in Eq. (6.16) is equivalent to the objective function in Eq. (6.15), given the following vector $\mathbf{g}^T$ and matrix $\mathbf{G}$:

$$\mathbf{g}^T = \left[\gamma f(\bar{\mathbf{x}}_i)_+ \quad \cdots \quad f(\bar{\mathbf{x}}_i)_+ - f_j(\bar{\mathbf{x}}_i) \quad \cdots \quad f(\bar{\mathbf{x}}_i)_+ - f_q(\bar{\mathbf{x}}_i) \right]$$

$$\mathbf{G} = \begin{bmatrix} \dfrac{\partial F(\bar{\mathbf{x}}_i)}{\partial \mathbf{x}_i} & \cdots & \dfrac{\partial f_j(\bar{\mathbf{x}}_i)}{\partial \mathbf{x}_i} & \cdots & \dfrac{\partial f_q(\bar{\mathbf{x}}_i)}{\partial \mathbf{x}_i} \\ \dfrac{\partial F(\bar{\mathbf{x}}_i)}{\partial a_i} & \cdots & \dfrac{\partial f_j(\bar{\mathbf{x}}_i)}{\partial a_i} & \cdots & \dfrac{\partial f_q(\bar{\mathbf{x}}_i)}{\partial a_i} \end{bmatrix} \tag{6.18}$$

LSSOL can solve Eq. (6.15) in a finite number of iterations and find the solution to $\boldsymbol{\mu}^* = [\mu_0^*, \ \mu_1^*, \ \cdots, \mu_q^*]^T$. Finally, the search direction is

$$\mathbf{h}_i = -\frac{1}{\delta}\left\{ \mu_0^* \nabla F(\bar{\mathbf{x}}_i) + \sum_{j=1}^{q} \mu_j^* \nabla f_j(\bar{\mathbf{x}}_i) \right\} \tag{6.19}$$

The second step finds an appropriate step size λ_i along the search direction $\mathbf{h}_i$ using the Armijo rule. Then, a new design is found by $\mathbf{x}_{i+1} = \mathbf{x}_i + \lambda_i \mathbf{h}_i$ and used as input data for the next iteration.

Like the inner approximation, the Polak-He algorithm may require an infinite number of iterations before it converges to the optimal solution $\mathbf{x}^*$ for Eq. (6.5). In our analysis, we terminate the Polak-He algorithm when

$$-\sigma_N \le \theta_{N+1}(\mathbf{x}_{N+1}) \le 0 \tag{6.20}$$

$$0 \le \psi_{N+1}(\mathbf{x}_{N+1}) \le \sigma_N \tag{6.21}$$

with $\theta_{N+1}(\cdot)$ and $\psi_{N+1}(\cdot)$ defined in Eqs. (3.25) and (3.23), respectively. The definition of $\sigma_N = 0.1/N^2$ is the same as for the inner approximation, which starts from larger tolerance and goes to 0 as N increases. This is reasonable here since the accuracy of the semi-infinite optimization algorithm gradually increases with the increase in the discretization number and since high accuracy is only required when approaching the design point.

In task B3 we need to evaluate three functions and their gradients with respect to the augmented design variable vector $\bar{\mathbf{x}}$: the objective function, structural constraint functions, and reliability constraint functions.

$$\frac{\partial[c_0(\mathbf{x})+c_f(\mathbf{x})a]}{\partial \mathbf{x}}, \frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}}, \frac{\partial \psi(\bar{\mathbf{x}})}{\partial \mathbf{x}} = \frac{\partial \psi(\bar{\mathbf{x}})}{\partial \mathbf{d}} \frac{\partial \mathbf{d}}{\partial \mathbf{x}}$$
$$\frac{\partial[c_0(\mathbf{x})+c_f(\mathbf{x})a]}{\partial a}, \frac{\partial \mathbf{f}(\mathbf{x})}{\partial a}, \frac{\partial \psi(\bar{\mathbf{x}})}{\partial a} = \frac{\partial \psi(\bar{\mathbf{x}})}{\partial \mathbf{d}} \frac{\partial \mathbf{d}}{\partial a} \tag{6.22}$$

where ψ is the negative value of the limit-state function g, $\partial\psi/\partial\mathbf{d}$ is easily found because g is a simple algebraic expression in terms of $\mathbf{d}$, and $\partial\mathbf{d}/\partial\mathbf{x}$ are response gradients. Again, the existing FDM or DDM implementations in OpenSees are used to obtain the required response gradients.

In conclusion, the MOOA algorithm solves a series of inequality-constrained problems in tasks B1 and B3. As the discretization number N increases, the MOOA algorithm results in a gradually more accurate solution $\mathbf{x}$. The optimal solution $\mathbf{x}^*$ is reached when N equals infinity. At the optimal point, the value of the objective function reaches its minimum, and the first-order optimality conditions in Eq. (3.21) are satisfied. Therefore, one of reliability constraints is active and equal to zero at the optimal point. This means that the reliability analysis finds the most probable failure point (MPP) $\mathbf{u}_\infty^*$ at the optimal design point, shown in Figure 6.3.

6.3 DSA-S Approach

This section develops a DSA-S approach by combining the problem reformulation in Section 6.1 with the findings of Du and Chen (2002) and Agarwal and Renaud (2004). However, the reformulated optimization problem is defined as an inequality constrained optimization problem with a single reliability constraint as opposed to infinite constraints in the DSA-MOOA approach. In this study, a deterministic optimization problem in Eq. (6.5) is first solved to find a new design $\mathbf{x}$. Second, a reliability analysis is performed to update the reliability constraint based on the new design $\mathbf{x}$. This sequential iteration is repeated until a consistent design is obtained.

Convergence cannot be proven mathematically for the DSA-S approach, since the failure probability used for the deterministic optimization analysis is obtained from the

last iteration. However, the DSA-S approach is still attractive because a consistent design is obtained at a considerably lower computational cost. The computational time of discretization in the DAS-MOOA approach is saved since there is only one reliability constraint. Another advantage of the DSA-S approach is the decoupling of the reliability and optimization calculations. It is flexible in selecting optimization algorithms and reliability computation methods.

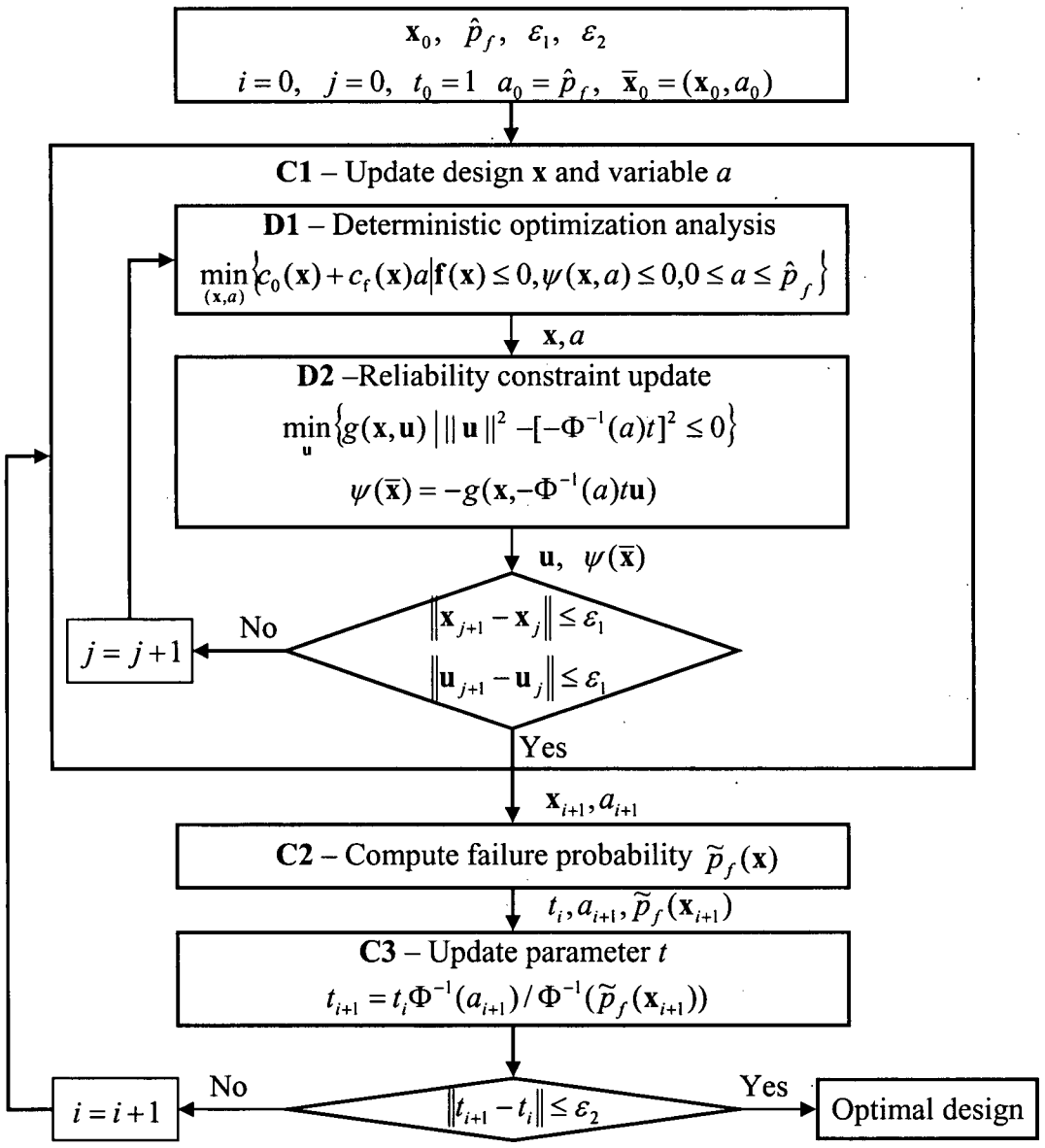

Figure 6.4 Flow chart of DSA-S approach

Figure 6.4 presents a flow chart of the DSA-S approach, which consists of iterations at several levels. Upon the initialization of design **x** and parameters a and t, the top level iteration includes three tasks:

C1. *Update design vector* **x** *and auxiliary variable a.*

C2. *Compute failure probability* $\tilde{p}_f(\mathbf{x})$ *using a method that is more accurate than FORM.*

C3. *Update parameter t.*

Tasks C2 and C3 are same as tasks A2 and A3 in the DSA-MOOA approach. The top level iteration is repeated until design vector **x**, auxiliary design variable a, and parameter t stabilize. Usually, 5 to 15 iterations (C1 to C3) are needed to reach a consistent reliability based design.

Task C1 updates values of **x** and a in two steps: deterministic optimization analysis and reliability constraint update. These are described in tasks D1 and D2, respectively. In the current implementation, the Polak-He algorithm is employed to solve these two tasks.

D1. Deterministic optimization analysis: *The inequality constrained optimization problem in Eq. (6.5) is solved using the Polak-He algorithm. The constraints in this problem include several structural constraints, one reliability constraint* $\psi(\bar{\mathbf{x}}) \le 0$, and an extra *constraint* $a \le \hat{p}_f$. *The result is a new augmented design vector* $\bar{\mathbf{x}} = (\mathbf{x}, a)$. *During the first iteration, the random variables are set equal to their mean values, and parameter a is set as* $\hat{p}_f$. *The Polak-He algorithm is terminated when user-defined tolerance* ε *is satisfied. Task D1 is similar to task B3 in the DSA-MOOA approach with only one reliability constraint and fixed high tolerance. Task D1 also needs to solve a quadratic sub-optimization problem with linear constraints using the quadratic programming software LSSOL (Gill* et al., *1986) or Matlab OPTM toolbox "Quadprog.m" (Matlab, 1999).*

D2. Reliability constraint update: *Obtain the minimum value of the limit-state function within the ball* $\mathrm{B}(\mathbf{0}, -\Phi^{-1}(a)\,t)$ *and the corresponding random vector* **u** *using*

the Polak-He algorithm. Then, the reliability constraint is updated with the new random vector **u.** *Terminate the Polak-He algorithm when the user-defined tolerance* ε *is satisfied. Task D2 is similar to task B1 in the DSA-MOOA approach with fixed high tolerance.*

Tasks D1 and D2 are repeated until both design variables and random variables are consistent. In other words, the calculations are terminated when $\|\mathbf{x}_{i+1} - \mathbf{x}_i\| \le \varepsilon$, $\|\mathbf{u}_{i+1} - \mathbf{u}_i\| \le \varepsilon$ and/or $i \ge i_\varepsilon$, where ε is a predefined positive parameter and i_ε is the maximum number of iterations. Typically, 3 to 10 iterations (D1 to D2) are required to find an optimal solution.

6.4 Object-Oriented Implementation in OpenSees

Implementation of RBDO procedures for finite element applications poses a number of challenges to the software developer. First, the software must be able to incorporate new developments and features as research progresses. Second, the interaction between the finite element, reliability, and optimization procedures must be robust and efficient. For instance, it is required that realizations of random variables and design variables be repeatedly mapped into the finite element model. Additionally, the response and response gradients must be accurately and efficiently computed in the finite element model and returned to the reliability and optimization algorithms. OpenSees has turned out to be an ideal software platform for this purpose. This is mainly due to the object-oriented software architecture that throughout the evolution of OpenSees has kept focus on maintainability and extensibility.

To implement the RBDO, five new objects were added into the *ReliabiltyDomain*, while two new analysis types (*runDSA-MOOAAnalysis* and *runDSA-SAnalysis*) were added to the original *ReliabilityAnalysis* object. As shown in Figure 6.5, the present optimization work adds *designVariable*, *designVariablePositioner*, *objectiveFunction*, *costFunction*, and *constraintFunction* objects to the *ReliabilityDomain*. A *designVariable*

object defines design variables by given their start points. A *designVariablePositioner* object is used to map the design variables into structural properties in the finite element model. The *objectiveFunction*, *costFunction*, and *constraintFunction* objects are user-defined expressions. A *costFunciton* object is used to create functions that are subsequently combined into an expression for an *objectiveFunction* object. A *constraintFunction* object may be expressed using various quantities including design variables and structural response quantities from an OpenSees finite element analysis.

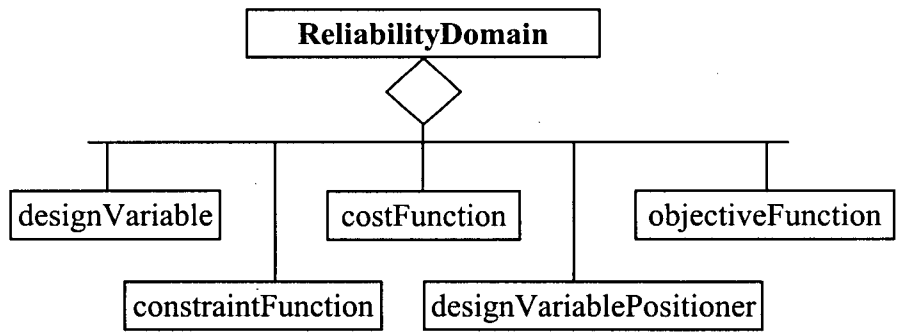

Figure 6.5 New objects for optimization analysis in *ReliabilityDomain*
(The diamond symbol denotes analysis tools)

In Figure 6.6, two new "analysis types" added to OpenSees are named *runDSA-MOOAAnalysis* and *runDSA-SAnalysis*. They are the orchestrating algorithms introduced in this thesis. The *runDSA-MOOAAnalysis* is the top level of the DSA-MOOA approach and is responsible for obtaining the optimal design by orchestrating tasks A1 to A3. The *runDSA-SAnalysis* is the top level of the DSA-S approach and is responsible for obtaining the optimal design by orchestrating tasks C1 to C3.

Three new "analysis tools" are also implemented: *NonlinSingleIneqOpt*, *NonlinMultiIneqOpt*, and *LinMultiIneqOpt*. These are so-called base classes that promise features but do not contain actual implementations. Any number of sub-classes may be implemented to perform the promised features. This illustrates the extensibility feature of OpenSees: new algorithms to perform various analysis tasks can be implemented without having to modify the software framework. The base class *NonlinSingleIneqOpt* promises to solve tasks B1 and D2. The sub-class implemented for this base class is named *PolakHeNonlinSingleIneqOpt*. Similarly, the base class *NonlinMultiIneqOpt* promises to

solve tasks B3 and D1. The current subclass implementation is *PolakHeNonlinMultiIneqOpt*. As mentioned above, in tasks B3 and D1 a quadratic sub-optimization problem with linear constraints is solved to find the search direction. This is fulfilled by the base class *LinMultiIneqOpt*. Currently, subclass *LSSOLLinMultiIneqOpt* is available in the implementation of OpenSees.

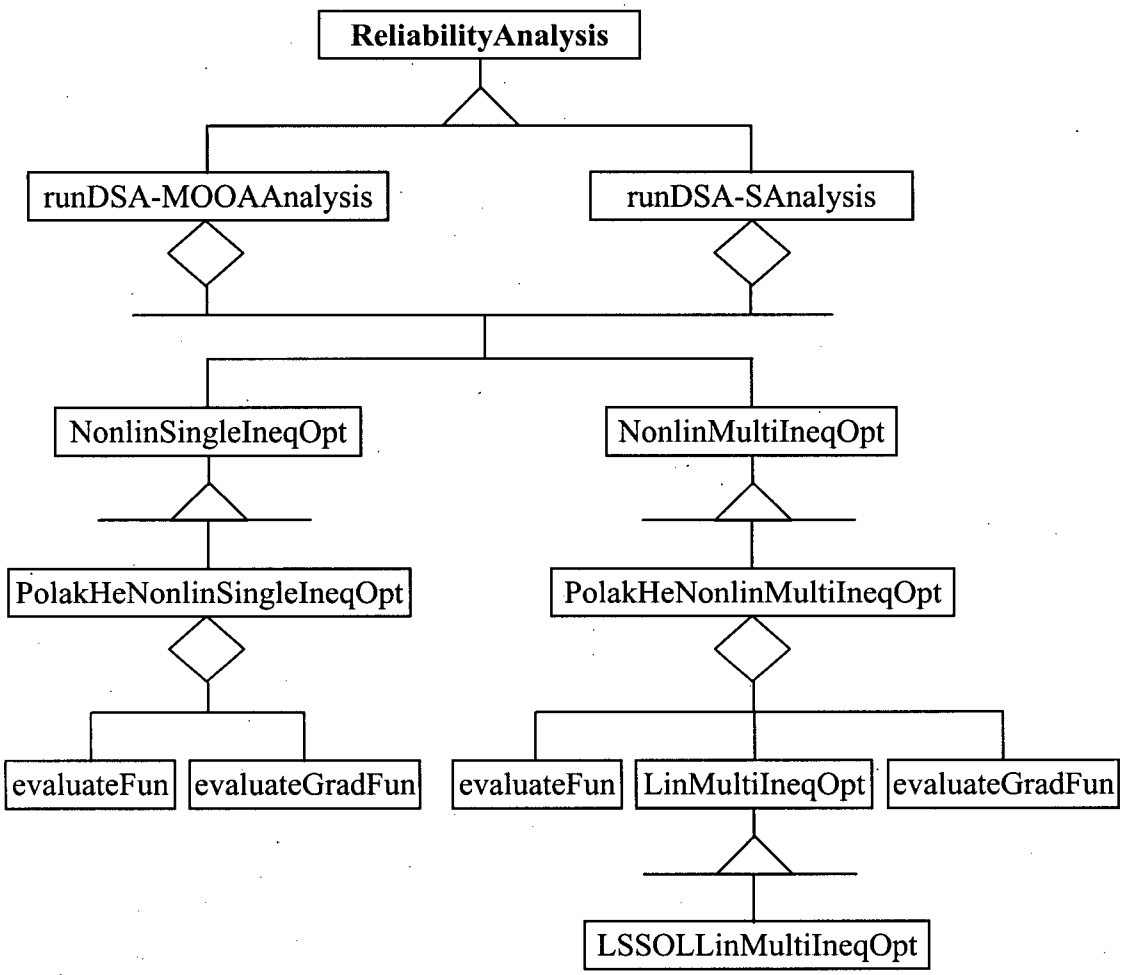

Figure 6.6 Software framework for optimization analysis in OpenSees
(Triangle symbol denotes the relationship between base class and subclasses, while the diamond symbol denotes analysis tools)

The category of "analysis tools" also contains classes such as *evaluateFun* and *evaluateGradFun*. The *evaluateFun* object evaluates the values of objective functions,

cost functions, and constraint functions. The *evaluateGradFun* object evaluates the gradients of objective functions, cost functions, and constraint functions.

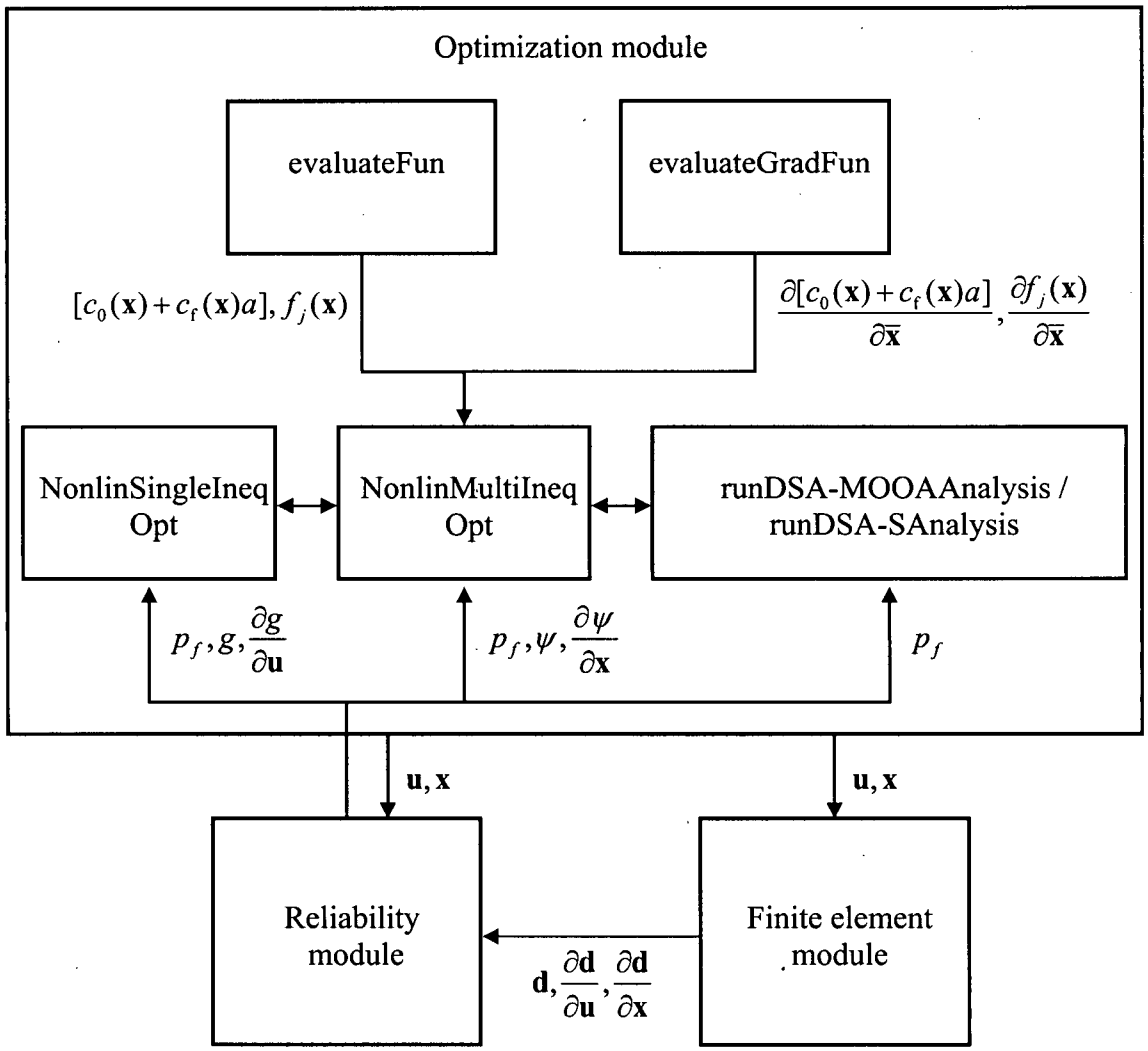

Figure 6.7 Interaction between optimization, reliability, and finite element module

Of particular interest in this implementation is the interaction between the above algorithms and the existing analysis tools in OpenSees. Implementations were already available to map random variables **u** and design variables **x** into the finite element model and the reliability module, as well as to obtain response and response sensitivities. Figure 6.7 schematically shows the interfaces among the optimization algorithm, the finite

element module, and the reliability module in OpenSees. Given design vector **x** and random vector **u**, the finite element module can compute response **d** and response sensitivities $\partial \mathbf{d} / \partial \mathbf{u}$ and $\partial \mathbf{d} / \partial \mathbf{x}$ with respect to random variables and design variables, respectively. The reliability module can return failure probability p_f, limit-state function value g, and gradients $\partial g / \partial \mathbf{u}$ with respect to random variables and $\partial g / \partial \mathbf{x}$ with respect to design variables. Function ψ in Figure 6.7 is the negative value of limit-state function g. Inside of the optimization module, *evaluateFun* and *evaluateGradFun* objects can provide the values and gradients of the objective functions, cost functions, and constraint functions. Response sensitivity methods FDM and DDM in Chapter 4 are used to compute such gradients.

Chapter 7 Numerical Examples and Case Studies

In this chapter, the new implementations in OpenSees are demonstrated by performing reliability-based design optimization (RBDO) of a real-world structure. The structure under consideration is a six-storey reinforced concrete building that serves as a design example in the Canadian Concrete Design Handbook (Cement Association of Canada, 1995). Three structural models are considered: (1) a linear elastic model, where the elements are modelled using the *elasticBeam* element of OpenSees; (2) a nonlinear model, where the elements are modelled using the *beamWithHinges* element of OpenSees; and (3) a nonlinear model, where the elements are modelled using the *nonlinearBeamColumn* element of OpenSees with fibre-discretized cross-sections. The response of the building is assessed using the static "pushover" analysis. The limit-state function is specified in terms of the total drift of the structure. The objective is to minimize the total expected cost of the design, given structural constraints and reliability constraints. Moreover, this study compares the convergence performance and the computation time for the algorithms and structural models presented herein. In particular, convergence problems may occur in the algorithms that address the inner and outer approximation problems described in the previous chapter. For example, a scaling of the involved functions is required when the Polak-He algorithm is used to address these problems. We also observe that the computational time is increased when redundant (inactive) constraints are added. Such experience from the hands-on optimization analysis is valuable for users of the developed software and is reported below.

7.1 Six-Storey Ductile Moment Resisting Frame

Consider a six-storey reinforced concrete frame building located in Vancouver, Canada (Cement Association of Canada, 1995). The building has seven bays with 6m spacing in the North-South (NS) direction and three bays (two office bays with 9m spacing and a

central 6m corridor bay) in the East-West (EW) direction. The interior columns are all 500×500mm, while the exterior columns are all 450×450mm. The beams of both NS and EW frames are 400mm wide×600mm deep for the first three storeys and 400×550mm for the top three storeys. Concrete with mean strength f_c' = 30MPa is used throughout, and the reinforcement has mean yield strength f_y = 400MPa. The Canadian Concrete Design Handbook (1995) specifies that the frame is designed as a ductile moment resisting frame with $R = 4.0$, where R is the ductility force modification factor that reflects the capacity of a structure to dissipate energy through inelastic behaviour.

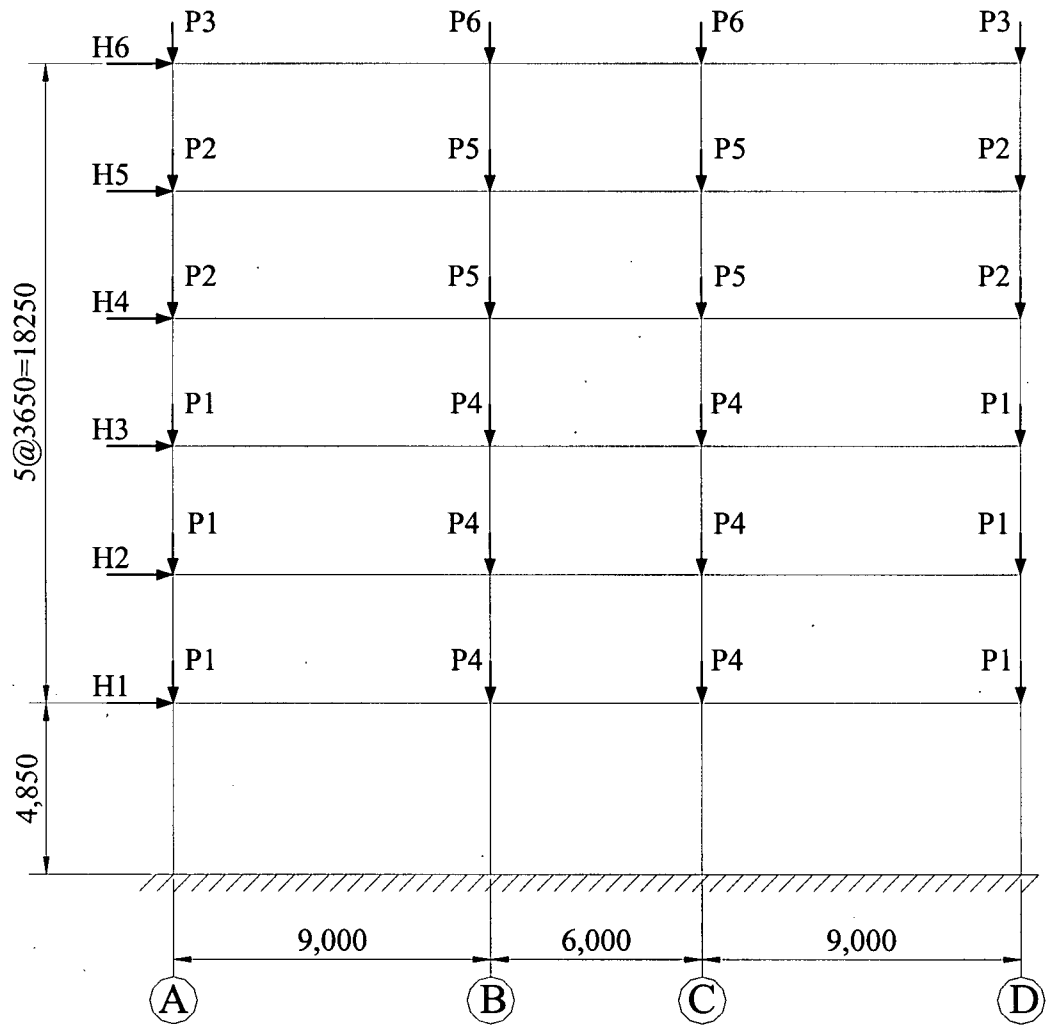

Figure 7.1 Ductile moment-resisting frame model

This thesis aims to optimize the design of the columns and beams of this ductile moment resisting frame. For this purpose, we consider linear and nonlinear pushover analyses of the second EW frame. The finite element model and the applied loads are illustrated in Figure 7.1.

The load case of "1.0×dead load + 0.5×live load + 1.0×earthquake load" is considered in the analysis. We consider dead loads and live loads as deterministic. The lateral loads from ground motion have lognormal distribution. Their means and coefficients of variation are shown in Table 7.1.

Table 7.1 Vertical loads and lateral loads (c.o.v. indicates the coefficient of variation)

Loads	Mean	c.o.v.	Type	Description
H_1	28490 kN	0.15	lognormal	random lateral load on floor 1
H_2	48950 kN	0.15	lognormal	random lateral load on floor 2
H_3	70070 kN	0.15	lognormal	random lateral load on floor 3
H_4	89100 kN	0.15	lognormal	random lateral load on floor 4
H_5	109780 kN	0.15	lognormal	random lateral load on floor 5
H_6	131890 kN	0.15	lognormal	random lateral load on roof
P_1	108000 kN	N/A	N/A	deterministic vertical load
P_2	105000 kN	N/A	N/A	deterministic vertical load
P_3	96000 kN	N/A	N/A	deterministic vertical load
P_4	184000 kN	N/A	N/A	deterministic vertical load
P_5	178000 kN	N/A	N/A	deterministic vertical load
P_6	182000 kN	N/A	N/A	deterministic vertical load

7.1.1 Case 1: Elastic Analysis using *elasticBeam* Elements

In earthquake engineering it is common to assess the structural capacity using inelastic pushover analysis. As a reference, a linear elastic analysis is also performed. In this thesis, the *elasticBeam* element of OpenSees is used for the linear elastic analysis. This type of element contains a linear elastic material model, without any yield limit. The

"equal displacement principle" shown in Figure 7.2 is employed to compute the total inelastic displacement d_e, subject to the lateral seismic force V. The solid line in the figure denotes the inelastic response. The corresponding linear system, signified by the dashed line, applies equivalent lateral seismic force $V_e = V \times R$ and results in the same displacement d_e.

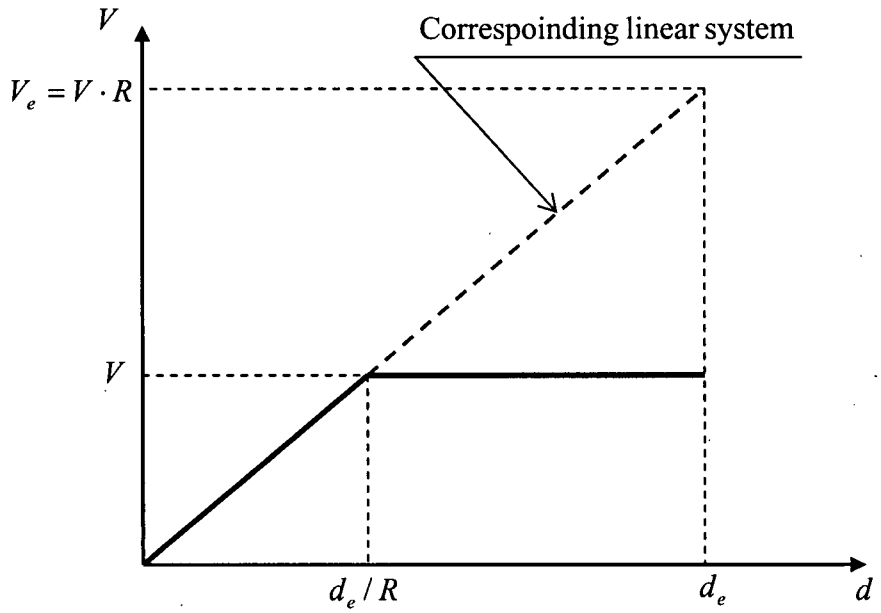

Figure 7.2 Equal displacement principle

In the linear case, 12 design variables are collected in the vector $\mathbf{x} = (b_1, h_1, b_2, h_2, b_3, h_3, b_4, h_4, b_5, h_5, b_6, h_6)$, as defined in Table 7.2. A total of 48 random variables describing the loading and material properties are collected in the vector $\mathbf{v} = (H_1, H_2, H_3, H_4, H_5, H_6, E_1, \cdots, E_{42})$, where H_1, H_2, H_3, H_4, H_5, and H_6 are the equivalent lateral loads from the first storey to the roof, respectively. E_1 to E_{42} represent the modulus of elasticity of the concrete material for all 42 elements. We assume that all random variables are lognormally distributed with the means and coefficients of variation listed in Table 7.3. Random variables H_1 to H_6 are correlated with the correlation coefficient of 0.7, while random variables E_1 to E_{42} are correlated with the correlation coefficient of 0.7.

Table 7.2 Definition and initial values of design variables for Cases 1 and 2

Variable	Initial Value	Description
$b_1 \times h_1$	0.45×0.45m	width and depth of exterior columns of first three stories
$b_2 \times h_2$	0.45×0.45m	width and depth of exterior columns of top three stories
$b_3 \times h_3$	0.50×0.50m	width and depth of interior columns of first three stories
$b_4 \times h_4$	0.50×0.50m	width and depth of interior columns of top three stories
$b_5 \times h_5$	0.40×0.60m	width and depth of first three stories' beams
$b_6 \times h_6$	0.40×0.55m	width and depth of top three stories' beams

Table 7.3 Statistics of random variables in Case 1 (c.o.v. indicates the coefficient of variation, and c.c. indicates the correlation coefficient)

Variable	Mean	c.o.v.	c.c.	Type	Description
H_1	4×28490 kN	0.15	0.7	lognormal	equivalent lateral load on floor 1
H_2	4×48950 kN	0.15		lognormal	equivalent lateral load on floor 2
H_3	4×70070 kN	0.15		lognormal	equivalent lateral load on floor 3
H_4	4×89100 kN	0.15		lognormal	equivalent lateral load on floor 4
H_5	4×109780 kN	0.15		lognormal	equivalent lateral load on floor 5
H_6	4×131890 kN	0.15		lognormal	equivalent lateral load on roof
$E_1 \sim E_{42}$	24648 MPa	0.15	0.7	lognormal	modulus of elasticity of concrete

The reliability problem for the frame is defined in terms of the limit-state function

$$g(\mathbf{d}(\mathbf{x}, \mathbf{v})) = 23.1 \times 0.02 - d_{\text{roof}} \qquad (7.1)$$

where 23.1m is the height of the frame, 0.02 is the maximum limit of the drift ratio, and d_{roof} is the roof displacement.

In this thesis, our objective is to achieve a frame design that minimizes the total expected cost, given specific constraints. For this purpose, we model the initial cost of design and the cost of failure in terms of the total volume of the members. The cost of

failure is assumed to be five times the volume of the members. This leads to the following objective function:

$$\sum_{i=1}^{6} b_i h_i \cdot L_i + p_f(\mathbf{x}) \cdot 5 \cdot \sum_{i=1}^{6} b_i h_i \cdot L_i \tag{7.2}$$

where L_i represent the total length of the members in each of the six categories identified in the design vector, while b_i and h_i are cross-sectional dimensions. The reliability constraint is prescribed as $p_f(\mathbf{x}) \leq 0.00135$, which implies a minimum reliability index of 3.0. The structural constraints are prescribed to be $0 \leq b_i$, h_i and $0.5 \leq b_i/h_i \leq 2$ to ensure positive dimensions and appropriate aspect ratios, where $i = [1, 2, 3, 4, 5, 6]$.

A stand-alone finite element reliability analysis using *elasticBeam* elements was performed. For initial values of the design variables in Table 7.2 and mean realizations of the random variables in Table 7.3, the lateral displacement at the roof level was 238mm. The corresponding drift ratio was 238/23100 = 1.03%, which was less than the limit of 2%. A first-order reliability analysis (FORM) resulted in a reliability index $\beta = 3.646$ and corresponding failure probability $p_f(\mathbf{x}_0) = 0.000133$, which is acceptable according to the prescribed reliability constraint. The total expected cost of the initial design in terms of volume was 54.062m^3.

The first optimization analysis was performed using the DSA-MOOA approach. This approach starts from the semi-infinite optimization analysis (task A1), which iteratively updates the constraints represented by ψ and obtains improved designs for $t = 1$. These were identified earlier as tasks B1 to B3. In this case, the limit-state function was linear, since a linear relationship exists between the random variables and the response quantity d_{roof}. Convergence was achieved within 1 to 3 iterations for task B1, and within 1 to 10 iterations for task B3. After discretizing the ball using the method of outer approximation (MOOA) algorithm by 75 points—after 75 loops of task B1 to B3—the algorithm repeatedly produced the same design. This was taken to indicate convergence. At the optimal design there were five reliability constraints. The tolerance of this solution to the "true" converged solution was $\sigma_N = 0.1/75^2 = 1.78 \times 10^{-5}$. The total expected cost was

reduced from 54.062m^3 to 38.701m^3, while the failure probability was 0.00135, which satisfied the reliability constraint. Then, an importance sampling analysis based on the new design variables was performed in task A2 to get the "real" failure probability with a 2% coefficient of variation of the sampling result. The results were 38.711m^3 for the total cost and 0.00140 for the failure probability. This difference was acceptable, and the RBDO was stopped after one top level of iteration (tasks A1 and A2). This was due to the linear nature of the problem.

The second optimization analysis was performed using the DSA-S approach. The algorithm started with task C1, which sequentially completed a deterministic optimization analysis and then found a new reliability constraint for t = 1. These were identified earlier as tasks D1 and D2. Similar to the DSA-MOOA approach, the convergences within tasks D1 and D2 was achieved quickly—within 1 to 30 iterations for task D1 and within 1 to 6 iterations for task D2—since the limit-state function was linear. We used the same tolerance as with the DSA-MOOA to judge whether convergence was achieved (*i.e.*, $\varepsilon = 0.1/75^2 = 1.78\times10^{-5}$). The optimal design was achieved after three loops of tasks D1 and D2. The DSA-S approach and the DSA-MOOA approach gave the same design: the total expected cost was 38.701m^3 and the failure probability was 0.00135. In the next task, C2, we got the same solution as in task A2 in the DSA-MOOA approach using importance sampling with a 2% coefficient of variation of the sampling result. We accepted this as the optimal design and terminated the analysis.

Table 7.4a Results from RBDO analysis for Case 1

t	b_1	h_1	b_2	h_2	b_3	h_3	b_4	h_4	b_5	h_5
1.000	0.45	0.45	0.45	0.45	0.5	0.5	0.5	0.5	0.4	0.6
	0.236	0.471	0.213	0.426	0.358	0.716	0.294	0.588	0.311	0.622

Table 7.4b Results from RBDO analysis for Case 1 (continued)

t	b_6	h_6	a	$\tilde{p}$	$c_0 + c_f a$	$c_0 + c_f \tilde{p}$
1.000	0.4	0.55	0.000133	0.000141	54.062	54.064
	0.261	0.522	0.00135	0.00140	38.701	38.711

Tables 7.4a and 7.4b show the results obtained from the two implemented approaches. The presented results include the value of the 12 design variables, the auxiliary parameter a, the failure probability $\tilde{p}_f$ from the importance sampling with a 2% coefficient of variation, and the total expected costs corresponding to a and $\tilde{p}_f$. The first rows show the values of the initial design, while the second rows show values of the optimal design. The two approaches produced the same solution, although the DSA-MOOA approach guaranteed convergence with a first order approximation, while the DSA-S approach did not.

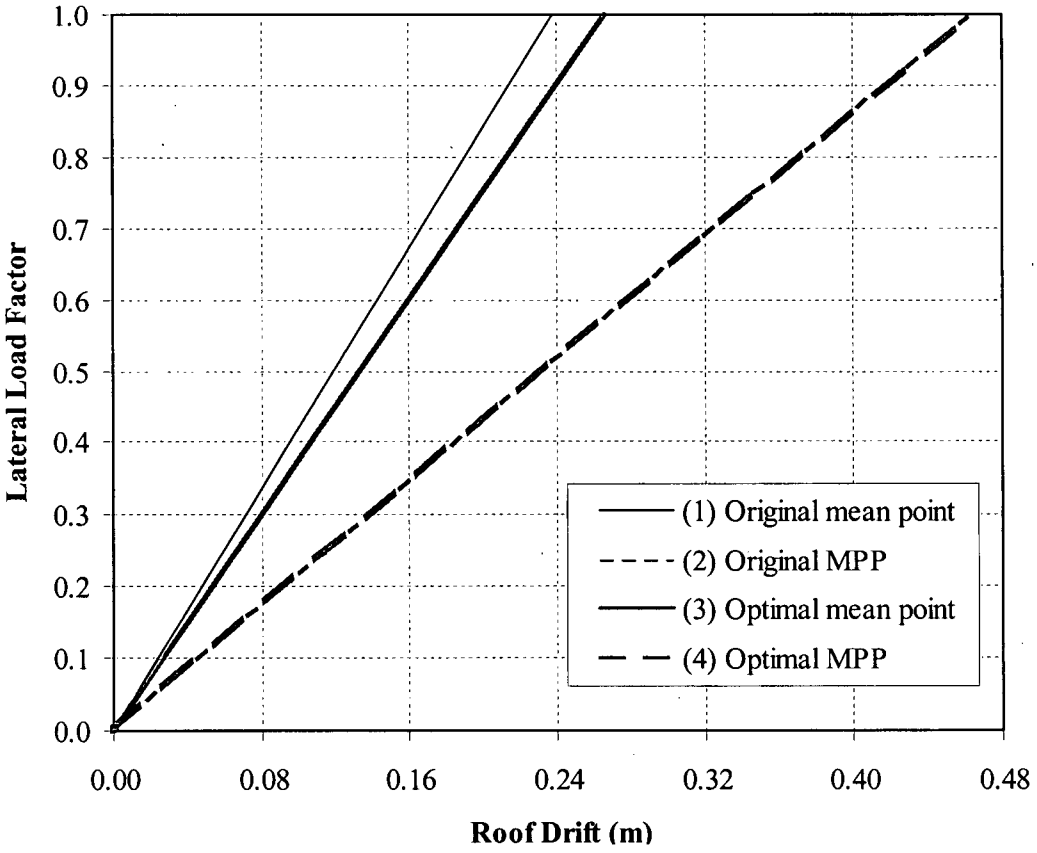

Figure 7.3 Structural responses for Case 1 (load factor versus roof displacement) at: (1) the mean point of the initial design; (2) the MPP of the initial design; (3) the mean point of the optimal design; and (4) the MPP of the optimal design.

Figure 7.3 shows the structural response of four characteristic realizations of design variables and random variables. The response at the mean realization of the random variables for the initial (original) design is shown as the thin solid line. As expected, this response is linear. At the most probable failure point (MPP) of the initial design, the structural response is still linear, but reaches the 2% of drift limit 0.462m, as prescribed by the limit-state function. The structural response at the mean realization of random variables for the optimal design is shown as the thick solid line. The response is linear, but has a larger displacement than in the initial design, which is consistent with the failure probability of the structure increasing from 0.000133 to 0.00135. Finally, the structural response at the MPP of the optimal design is linear and reaches 2% of the drift limit. The optimal design has an acceptable reliability, but a lower total expected cost. This serves as an indication of the usefulness of the RBDO approach.

Table 7.5 Comparison of computational time for Case 1

	DSA-MOOA		DSA-S	
	DDM	FDM	DDM	FDM
g	555	8559	147	1227
g with $\partial g/\partial \mathbf{u}$	80	N/A	8	N/A
g with $\partial g/\partial \mathbf{x}$	347	N/A	58	N/A
Importance Sampling	112486		112607	

It is of interest to compare the computational cost of two implemented approaches, particularly with respect to which method is being used to compute the response gradients, the direct differentiation method (DDM) or the finite difference method (FDM). For this purpose, Table 7.5 lists the number of calls to the limit state function, which is a key indicator when studying computation time. The DDM method is much more efficient in computing gradients than the FDM method. For example, the DSA-MOOA approach requires 555 single limit-state function calls, 80 limit-state function calls together with the computation of $\partial g / \partial \mathbf{u}$, and 347 limit-state function calls together

with the computations of $\partial g / \partial \mathbf{x}$. Thus, altogether 982 limit-state function calls are required for the DDM method. On the other hand, the FDM method computes the gradient using one extra limit-state function evaluation for each random variable and design variable. Thus, the total number of limit-state function calls required for the FDM method is 555, as well as 80 times the number of random variables (48) and 347 times the number of design variables (12). Finally, 8,559 limit-state function calls are required, which is much more than needed for the DDM method. The two approaches require a similar number of simulations to compute the failure probability using importance sampling, shown in the last row of Table 7.5.

The DSA-S approach, which requires 213 (147+8+58) limit-state function calls, appears to be more efficient than the DSA-MOOA approach, which requires 982 limit-state function calls for the DDM method. The key reason for this is that only one reliability constraint is maintained in the DSA-S approach, while the DSA-MOOA approach expands reliability constraints step by step when discretizing the ball with more points.

7.1.2 Case 2: Nonlinear Analysis using *beamWithHinges* Elements

In this section we perform a nonlinear pushover analysis by using the *beamWithHinges* element of OpenSees. We consider the plasticity to be concentrated at over 10% of the element length at the element ends. The elastic properties are integrated over the beam

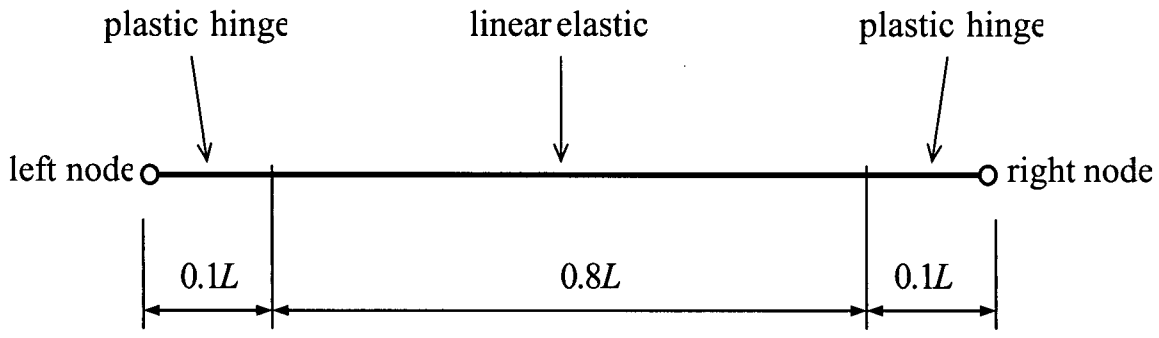

Figure 7.4 *beamWithHinges* element

interior, which is considered to be linear elastic. Forces and deformations of the inelastic region are sampled at the hinge midpoints. A bi-linear or smooth uniaxial material is used in the plastic hinge region to model the moment-rotation relationship. A typical *beamWithHinges* element used in this thesis is illustrated in Figure 7.4.

As in the linear case, 12 design variables shown in Figure 7.2 are collected in the vector $\mathbf{x} = (b_1, h_1, b_2, h_2, b_3, h_3, b_4, h_4, b_5, h_5, b_6, h_6)$. 48 random variables are collected in the vector $\mathbf{v} = (H_1, H_2, H_3, H_4, H_5, H_6, E_1, \cdots, E_{42})$ to describe loading and material properties. We assume that all random variables are lognormally distributed with the means and coefficients of variation listed in Table 7.6. Random variables H_1 to H_6 are correlated with the correlation coefficient of 0.7, and random variables E_1 to E_{42} are correlated with the correlation coefficient of 0.7. The limit-state function and the objective function are as defined in Eqs. (7.1) and (7.2). The reliability constraint and structural constraints are as prescribed for the linear structure.

Table 7.6 Statistics of random variables for Case 2 (c.o.v. indicates the coefficient of variation, and c.c. indicates the correlation coefficient)

Variable	Mean	c.o.v.	c.c.	Type	Description
H_1	28490 kN	0.15	0.7	lognormal	lateral load on floor 1
H_2	48950 kN	0.15		lognormal	lateral load on floor 2
H_3	70070 kN	0.15		lognormal	lateral load on floor 3
H_4	89100 kN	0.15		lognormal	lateral load on floor 4
H_5	109780 kN	0.15		lognormal	lateral load on floor 5
H_6	131890 kN	0.15		lognormal	lateral load on roof
$E_1 \sim E_{42}$	11097 MPa	0.15	0.7	lognormal	modulus of elasticity of concrete

A stand-alone finite element reliability analysis was performed. The bi-linear material model was employed to model the plastic hinges. The stiffness of cross-section was evaluated by $EI = Ebh^3/12$, where b and h were the width and the depth of the section,

and the value of E was smaller than in the linear case because of considering concrete cracking. For all columns, the yield stain $\varepsilon_y = 0.84$, and the strain hardening factor $\alpha = 0.5$. For all the beams, the yield stain $\varepsilon_y = 0.52$, and the strain hardening factor $\alpha = 0.3$. At the mean realization of the random variables in Table 7.6 and with the initial design in Table 7.2, the lateral displacement at the roof level was 131mm. The corresponding drift ratio was 131/23100 = 0.57%, which was less than the limit of 2%. A reliability analysis using the FORM resulted in a reliability index β = 3.536 and the corresponding failure probability $p_f(\mathbf{x}_0) = 0.000203$, which satisfied the prescribed reliability constraint. The total expected cost of the initial design was 54.080m^3.

The first optimization analysis was performed using the DSA-MOOA approach with the bi-linear material model. As outlined previously, the algorithm starts with the semi-infinite optimization analysis (task A1). In this case, the convergence within task B3 was achieved for the first few iterations, namely when the number of constraints represented by ψ was low. However, the algorithm in task B3, the *PolakHeNonlinMultiIneqOpt*, exhibited progressively slower convergence as the number of constraints increased. In fact, this problem made the algorithm grind to a halt. The presence of gradient discontinuities due to sudden yielding events of the bi-linear material models was taken to be the reason for this problem.

As a remedy to the above problem, a smoothed version of the bi-linear model introduced in Chapter 5 was substituted. A circular segment in a normalized stress-strain plane that started at 80% of the yield strength, $\gamma = 0.8$, was employed to smooth the bi-linear material as illustrated in Figure 5.2. Remarkably, the analysis proceeded without any of the convergence problems described above. Convergence was achieved within 1 to 6 iterations for task B1, and within 1 to 65 iterations for task B3. This led us to conclude that the presence of a non-smooth response surface due to sudden yielding events was a serious impediment to the performance of the algorithm. Similar problems were also observed in the stand-alone reliability analysis. However, in our experience the problem was significantly amplified in the optimization analysis context.

After discretizing the ball in the MOOA algorithm by 75 points, or after 75 loops of tasks B1 to B3, the algorithm repeatedly produced the same design. At the design point, there were 12 reliability constraints. The tolerance of this solution to the "true" converged point was $\sigma_N = 0.1/75^2 = 1.78\times10^{-5}$. The total cost was reduced from 54.080m^3 to 37.108m^3, and the failure probability was 0.00135, which satisfied the reliability constraint. In the next task, A2, importance sampling based on the new design variables was performed to get the "real" failure probability with a 2% coefficient of variation of the sampling result. The results were 37.127m^3 as the total cost and 0.00146 as the failure probability. The difference between the two failure probabilities (0.00135 from task A1 and 0.00146 from task A2) shows the nonlinearity of the structure. In task A3, parameter t was updated, and the top level of the iteration (tasks A1 to A3) was repeated. After two more loops of tasks A1 to A3, the differences of failure probabilities between task A1 and A2 were reduced and accepted, and the RBDO was stopped. The final total cost was 37.197m^3, and the failure probability was 0.00135.

The second optimization analysis was performed using the DSA-S approach with the smooth material model. The algorithm started with task C1. Convergence was achieved within 1 to 109 iterations for task D1, and within 1 to 13 iterations for task D2. We used the same tolerance to judge the consistent design (*i.e.*, $\varepsilon = 0.1/75^2 = 1.78\times10^{-5}$). The consistent design was achieved after four loops of tasks D1 and D2. The DSA-S approach and the DSA-MOOA approach gave the same design: the total cost was 37.108m^3 and the failure probability was 0.00135. In the next task, C2, the DSA-S approach produced the same solution as task A2 in the DSA-MOOA approach using importance sampling with a 2% coefficient of variation. As in the DSA-MOOA approach, the top level of iteration (tasks C1 to C3) was repeated for two more loops, producing consistent designs. The optimization procedure was stopped at the total cost of 37.197m^3 and the failure probability of 0.00135.

Tables 7.7a and 7.7b show the results obtained from the two implemented approaches. The presented results include the value of 12 design variables, the auxiliary parameter a, the failure probability $\tilde{p}$ from the importance sampling with a 2% coefficient of

variation, and the total expected cost corresponding to a and $\tilde{p}$. The first rows show the values of the initial design, while the following rows show values of the optimal design. In each of these iterations, the parameter t is updated to account for nonlinearities in the limit-state function. After the first iteration, the value of t was updated as $1.0 \times \Phi^{-1}(0.00135)/\Phi^{-1}(0.00146) = 1.0077$. The analysis was carried out for two more iterations. No appreciable difference in the design was observed. In the last row of the table, a and $\tilde{p}$ converge to the same acceptable value, 0.00135. In effect, the objective functions have reached the minimum value: 37.197m^3. Hence, the design variables in the last row of the table can be accepted as the optimal design.

Table 7.7a Results from RBDO analysis for Case 2

t	b_1	h_1	b_2	h_2	b_3	h_3	b_4	h_4	b_5	h_5
1.000	0.45	0.45	0.45	0.45	0.5	0.5	0.5	0.5	0.4	0.6
1.0077	0.246	0.491	0.200	0.399	0.316	0.631	0.289	0.577	0.323	0.645
1.0089	0.246	0.492	0.200	0.400	0.316	0.632	0.289	0.578	0.323	0.646
1.0086	0.246	0.492	0.200	0.400	0.316	0.632	0.289	0.578	0.323	0.646

Table 7.7b Results from RBDO analysis for Case 2 (continued)

t	b_6	h_6	a	$\tilde{p}$	$c_0 + c_f a$	$c_0 + c_f \tilde{p}$
1.000	0.4	0.55	0.000203	0.000219	54.080	54.085
1.0077	0.246	0.492	0.00135	0.00146	37.108	37.127
1.0089	0.246	0.492	0.00135	0.00137	37.186	37.189
1.0086	0.246	0.492	0.00135	0.00135	37.197	37.197

It is observed that the two approaches obtain an improved design that is close to the final solution already after the first top-level iteration. This can also be seen in Figures 7.5, where the total expected costs (objective function) are plotted as the function of the iteration number. This phenomenon shows that the structure is not highly nonlinear and that the implemented approaches are effective in dealing with the nonlinear problem.

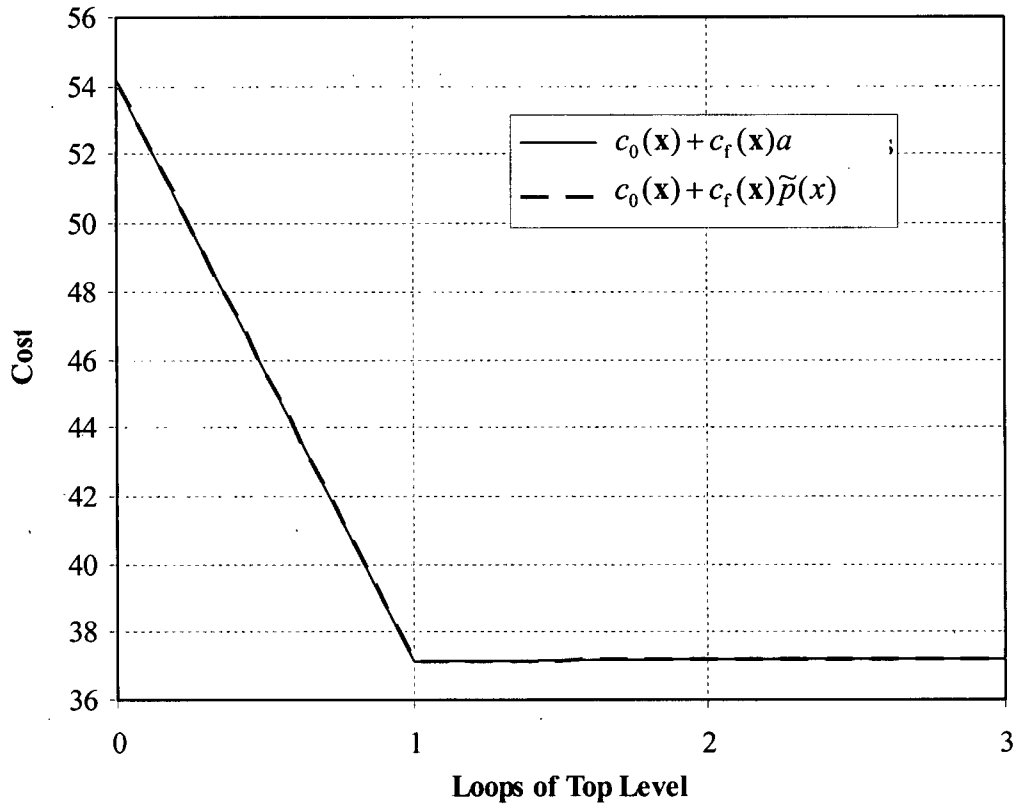

Figure 7.5 Evolution of the total expected cost for objective functions for Case 2 (1) $c_0(\mathbf{x})+c_f(\mathbf{x})a$ and (2) $c_0(\mathbf{x})+c_f(\mathbf{x})\tilde{p}(x)$

Figure 7.6 shows the structural response for four characteristic realizations of design variables and random variables. The response at the mean realization of random variables of the initial (original) design is shown as the thin solid line. As expected, this response is close to linear, because no significant damage (yielding) is anticipated for this realization. At the MPP of the initial design, however, substantial yielding occurs. This is reasonable, since this realization represents failure. Third, the structural response at the mean realization of random variables for the optimal design is shown as the thick solid line. This response has larger displacement than the initial design. Finally, the structural response at the MPP of the optimal design is also shown. Again, significant nonlinearity in the finite element response is observed. This response is similar to that of the initial design, but it is not equal to it. This is reasonable, because the limit-state function is

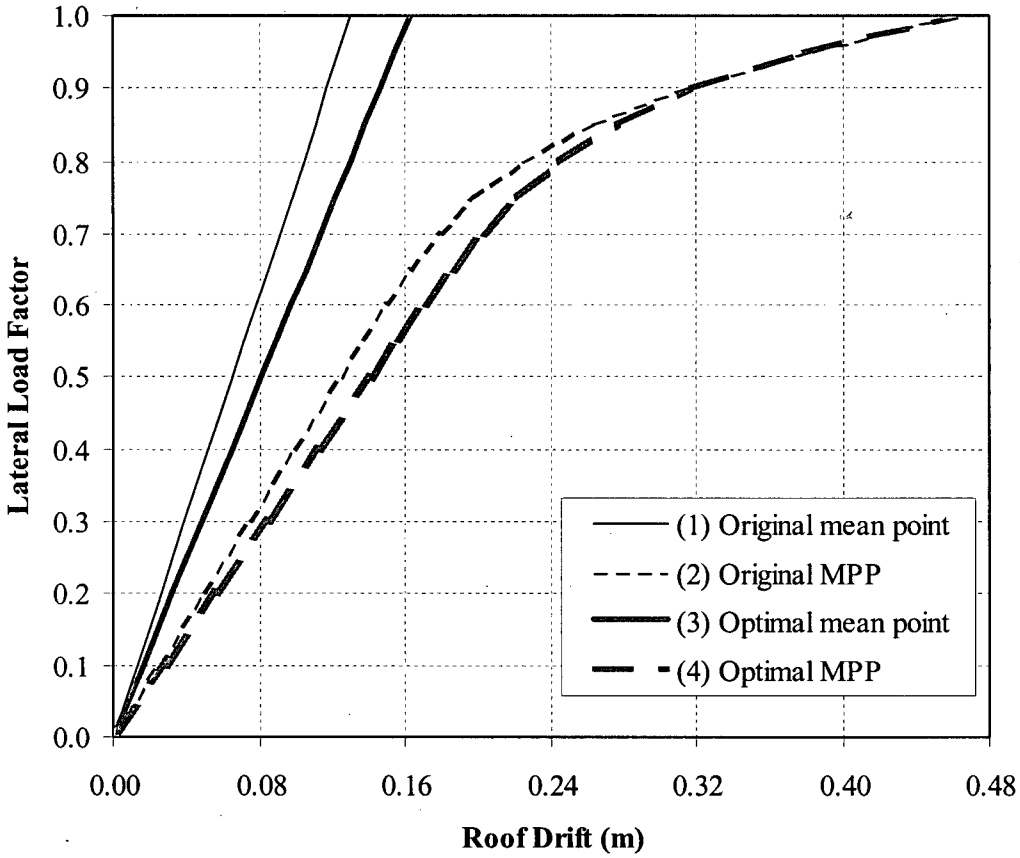

Figure 7.6 Structural responses for Case 2 (load factor versus roof displacement) at: (1) the mean point of the initial design; (2) the MPP of the initial design; (3) the mean point of the optimal design; and (4) the MPP of the optimal design.

altered by changes in the structural design. The apparent lower value of the stiffness at the MPP of the optimal design is explained as follows: for the optimal design a greater reduction of the stiffness is required to "achieve" failure (*i.e.*, to obtain the MPP). Again, we observe that the optimal design has an acceptable reliability and a reduced total expected cost. This serves as an indication of the usefulness of the RBDO approach.

Table 7.8 compares the computational cost of the two implemented approaches. The data in the table come from the first iteration, and are almost the same as the data from the second and third iterations. The DSA-S approach, requiring 6,195 limit-state function calls, appears to be more efficient than the DSA-MOOA approach, requiring 22,730 limit-state function calls using the FDM method. We have observed that the nonlinear

case requires significantly more effort than the linear case, which only requires 1,227 and 8,559 limit-state function calls. For the linear case, updates in design do not dramatically change the corresponding MPP of the reliability analysis. There are only five reliability constraints after 75 loops of tasks B1 to B3. For the nonlinear cases, however, the MPP of the reliability analysis clearly changes when a new design is found. In addition, in nonlinear cases there are 12 reliability constraints after 75 loops of tasks B1 to B3.

Table 7.8 Comparison of computational time for Case 2

	DSA-MOOA by FDM	DSA-S by FDM
g	22730	6195
Importance Sampling	109436	109342

7.1.3 Case 3: Nonlinear Analysis using *dispBeamColumn* Elements and *fibre* Sections

In this section we perform a nonlinear pushover analysis by using the *dispBeamColumn* element and the *fibre* section of OpenSees. To describe a better curvature distribution along the element, one original element was divided into four elements, with four integration points along each element. Each column and beam section was discretized into about 20 fibres to give an "approximately continuous" structural response. All of the fibres were described using the bi-linear concrete material. In this case, an *elasticPerfectlyPlastic* material is used as the concrete material by setting *FyP* as 0, as shown in Figure 4.4. For the reinforced bars of these sections, the smooth steel material was used, as shown in Figure 5.2. Typical fibre sections of columns and beams are illustrated in Figure 7.7.

In this nonlinear case, 18 design variables are collected in the vector $\mathbf{x} = (b_1, h_1, b_2, h_2, b_3, h_3, b_4, h_4, b_5, h_5, b_6, h_6, A_1, A_2, A_3, A_4, A_5, A_6)$. In addition to b and h

defined in Cases 1 and 2, this case has the area of steel bars A as design variables. The definitions and initial values of b, h, and A are described in Table 7.9.

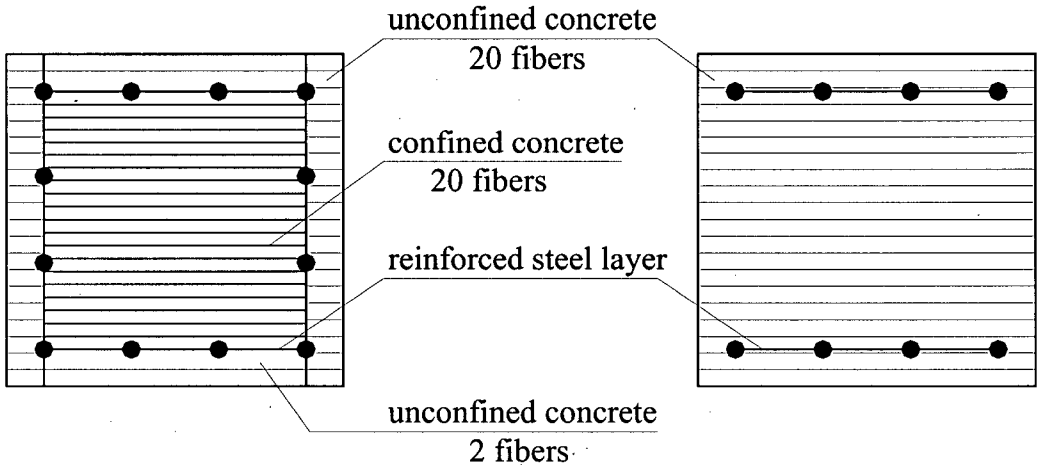

Figure 7.7 Typical fibre sections for columns and beams

78 random variables for the loading and material properties are collected in the vector $\mathbf{v} = (H_1, \cdots, H_6, f'_{cc1}, \cdots, f'_{cc8}, E_{cc1}, \cdots, E_{cc8}, f'_{c1}, \cdots, f'_{c14}, E_{c1}, \cdots, E_{c14}, f_{y1}, \cdots, f_{y14}, E_1, \cdots, E_{14})$. We assume that all random variables are lognormally distributed with the means and coefficients of variation listed in Table 7.10. The random variables are correlated with the correlation coefficient of 0.7 in several groups. More specifically, we have eight random variables for confined concrete strength f'_{cc} and eight random variables for modulus of elasticity of confined concrete E_{cc}. They are assigned to eight types of columns: first three-storey columns and top three-storey columns on four axes A, B, C, and D. We also have 14 random variables for unconfined concrete strength f'_c and 14 random variables for modulus of elasticity of unconfined concrete E_c. They are assigned to eight types of columns and six types of beams: first two-storey beams, middle two-storey beams, and top two-storey beams. In addition, we have 14 random variables for steel bars strength f_y and 14 random variables for modulus of elasticity of steel E assigned to eight types of columns and six types of beams.

Table 7.9 Definition and initial values of design variables for Case 3

Variable	Initial Value	Description
$b_1 \times h_1$	0.45×0.45m	width and depth of exterior columns of first three stories
A_1	0.003m^2	half of the area of reinforced bars of exterior columns of first three stories
$b_2 \times h_2$	0.45×0.45m	width and depth of exterior columns of top three stories
A_2	0.003m^2	half of the area of reinforced bars of exterior columns of top three stories
$b_3 \times h_3$	0.50×0.50m	width and depth of interior columns of first three stories
A_3	0.003m^2	half of the area of reinforced bars of interior columns of first three stories
$b_4 \times h_4$	0.50×0.50m	width and depth of interior columns of top three stories
A_4	0.003m^2	half of the area of reinforced bars of interior columns of top three stories
$b_5 \times h_5$	0.40×0.60m	width and depth of exterior columns of first three stories
A_5	0.0024m^2	area of reinforced bars of first three stories' beams
$b_4 \times h_4$	0.40×0.55m	width and depth of exterior columns of top three stories
A_6	0.0024m^2	area of reinforced bars of top three stories' beams

The limit-state function was as defined in the same way as in Eq. (7.1). The objective function was described in terms of the total volume of the members. Because of the price difference between two materials in the current market (the price of steel bars per cubic meter was 100 times the price of the concrete per cubic meter), the volume of steel bars was accounted for by using its equivalent concrete volume, which was equal to 100 times the actual volume of the steel bars. Again, the cost of failure was assumed to be five times the initial volume. This led to the following objective function:

$$\begin{aligned} &\left(\sum_{i=1}^{4} (b_i h_i + 100 \cdot 2 \cdot A_i) \cdot L_i + \sum_{i=5}^{6} (b_i h_i + 100 \cdot A_i) \cdot L_i \right) \\ &+ p_f(\mathbf{x}) \cdot 5 \cdot \left(\sum_{i=1}^{4} (b_i h_i + 100 \cdot 2 \cdot A_i) \cdot L_i + \sum_{i=5}^{6} (b_i h_i + 100 \cdot A_i) \cdot L_i \right) \end{aligned} \tag{7.3}$$

where L_i represents the total length of the members in each of the six categories identified in the design vector. The reliability constraint was still prescribed as $p_f(\mathbf{x}) \le 0.00135$.

The structural constraints were prescribed as $0 \le b_i$, h_i and $0.5 \le b_i/h_i, \le 2$ to ensure positive dimensions and appropriate aspect ratios, where i = [1, 2, 3, 4, 5, 6]. The structural constraints for the area of steel bars were $0.01 \cdot b_i h_i \le A_i \le 0.02 \cdot b_i h_i$ for columns, where i = [1, 2, 3, 4], and $0.008 \cdot b_i h_i \le A_i \le 0.02 \cdot b_i h_i$ for beams, where i = [5, 6], to ensure appropriate reinforced bar ratios.

Table 7.10 Statistics of random variables for Case 3 (c.o.v. indicates the coefficient of variation, and c.c. indicates the correlation coefficient)

Variable	Mean	c.o.v.	c.c.	Type	Description
H_1	28490 kN	0.15	0.7	lognormal	lateral load on floor 1
H_2	48950 kN	0.15		lognormal	lateral load on floor 2
H_3	70070 kN	0.15		lognormal	lateral load on floor 3
H_4	89100 kN	0.15		lognormal	lateral load on floor 4
H_5	109780 kN	0.15		lognormal	lateral load on floor 5
H_6	131890 kN	0.15		lognormal	lateral load on roof
$f'_{cc1} \cdots f'_{cc8}$	39 MPa	0.15	0.7	lognormal	confined concrete strength
$E_{cc1} \cdots E_{cc8}$	9750 MPa	0.10	0.7	lognormal	modulus of elasticity of confined concrete
$f'_{c1} \cdots f'_{c14}$	30 MPa	0.15	0.7	lognormal	unconfined concrete strength
$E_{c1} \cdots E_{c14}$	15000 MPa	0.10	0.7	lognormal	modulus of elasticity of unconfined concrete
$f_{y1} \cdots f_{y14}$	400 MPa	0.15	0.7	lognormal	steel bars strength
$E_1 \cdots E_{14}$	200000 MPa	0.05	0.7	lognormal	modulus of elasticity of steel

A stand-alone finite element reliability analysis was performed. At the mean realization of the random variables in Table 7.10, with the initial design in Table 7.9, the lateral displacement at the roof level was 168mm. The corresponding drift ratio was 168/23100 = 0.73%, which is less than the limit of 2%. A reliability analysis by the FORM resulted in a reliability index β = 3.120 and the corresponding failure probability

$p_f(\mathbf{x}_0) = 0.000903$, which satisfied the prescribed reliability constraint. The total expected cost of the initial design was 130.753m^3, which was larger than Case 1 and 2, since in this case we considered the area of reinforced bars.

With the experience of solving convergence problem in Case 2, we were confident in solving the nonlinear problem using *dispBeamColumn* elements and discretized *fibre* sections. First, we performed an optimization analysis using the DSA-MOOA approach. The semi-infinite optimization analysis (task *A1*) converges quickly: within 1 to 5 iterations for task B1, and within 1 to 21 iterations for task B3. The results show that discretized concrete fibre sections, together with the smooth steel material, can achieve a "continuous" structural response. After discretizing the ball by 75 points, or after 75 loops of task B1 to B3, the algorithm repeatedly produced the same design. At the optimal design there were 17 reliability constraints. The tolerance of this solution to the "true" converged point was $\sigma_N = 0.1/75^2 = 1.78\times10^{-5}$. The total cost was reduced from 130.753m^3 to 85.221m^3, and the failure probability was 0.00135, which satisfied the reliability constraint. Next, an importance sampling based on the new design variables was performed to get the "real" failure probability with a 2% coefficient of variation (task A2). The results were 85.327m^3 for the total cost and 0.00160 for the failure probability. This difference between the two failure probabilities (0.00135 from task A1 and 0.00160 from task A2) shows the nonlinearity of the structure. The parameter t was updated in task A3, and the top level of iteration (task A1 to A3) was repeated. After two more loops of tasks A1 to A3, the RBDO was stopped when the differences in failure probabilities between tasks A1 and A2 were reduced to an accepted level. The final total cost was 85.663m^3, and the failure probability was 0.00135.

The second optimization analysis was performed using the DSA-S approach. The approach began from task C1. Convergence was achieved within 1 to 88 iterations for task D1, and within 1 to 17 iterations for task D2. We used the same tolerance as in the DSA-MOOA approach to judge the consistent design (*i.e.*, $\varepsilon = 0.1/75^2 = 1.78\times10^{-5}$). An optimal design was achieved after five loops of tasks D1 and D2. The DSA-S approach and the DSA-MOOA approach produced the same design. The total cost was 85.221m^3,

and the failure probability was 0.00135. In the next task, C2, the DSA-S approach produced the same solution as task A2 of the DSA-MOOA approach using importance sampling with a 2% coefficient of variation. As the DSA-MOOA approach, the top level of iteration (tasks C1 to C3) was repeated for two more loops and the designs were consistent. The entire optimization procedure was stopped at the total cost of 85.663m^3 and failure probability of 0.00135.

Table 7.11a Results from RBDO analysis for Case 3

t	b_1	h_1	b_2	h_2	b_3	h_3	b_4	h_4	b_5	h_5
1.000	0.45	0.45	0.45	0.45	0.5	0.5	0.5	0.5	0.4	0.6
1.0177	0.224	0.449	0.181	0.362	0.306	0.611	0.286	0.572	0.334	0.668
1.0182	0.225	0.450	0.182	0.363	0.306	0.612	0.287	0.573	0.335	0.670
1.0182	0.225	0.450	0.182	0.363	0.306	0.612	0.287	0.573	0.335	0.670

Table 7.11b Results from RBDO analysis for Case 3 (continued)

t	b_6	h_6	A_1	A_2	A_3	A_4	A_5	A_6
1.000	0.4	0.55	0.003	0.003	0.003	0.003	0.0024	0.0024
1.0177	0.334	0.524	0.0010	0.0007	0.0019	0.0016	0.0029	0.0014
1.0182	0.335	0.526	0.0010	0.0007	0.0019	0.0016	0.0029	0.0014
1.0182	0.335	0.526	0.0010	0.0007	0.0019	0.0016	0.0029	0.0014

Table 7.11c Results from RBDO analysis for Case 3 (continued)

t	a	$\tilde{p}$	$c_0 + c_f a$	$c_0 + c_f \tilde{p}$
1.000	0.000903	0.000941	130.753	130.778
1.0177	0.00135	0.00160	85.221	85.327
1.0182	0.00135	0.00136	85.652	85.655
1.0182	0.00135	0.00135	85.663	85.663

Tables 7.11a, 7.11b, and 7.11c show the results obtained from the two implemented approaches. The presented results include the value of 18 design variables, the auxiliary parameter a, the failure probability $\tilde{p}_f$ from the importance sampling with a 2% coefficient of variation, and the total expected costs corresponding to a and $\tilde{p}_f$. The first rows show the values of the initial design, while the following rows show the values of the optimal design. In each of these iterations, parameter t was updated to account for nonlinearities in the limit-state function. After the first iteration, the value of t was updated as $1.0 \times \Phi^{-1}(0.00135)/\Phi^{-1}(0.00160) = 1.0177$. The analysis was carried out for two more iterations. No appreciable difference in the design were observed. In the last row, a and $\tilde{p}_f$ converge to the same acceptable value 0.00135. In effect, the objective functions have reached the minimum value, 85.663m^3. Hence, the design variables in the last row were accepted as the optimal design.

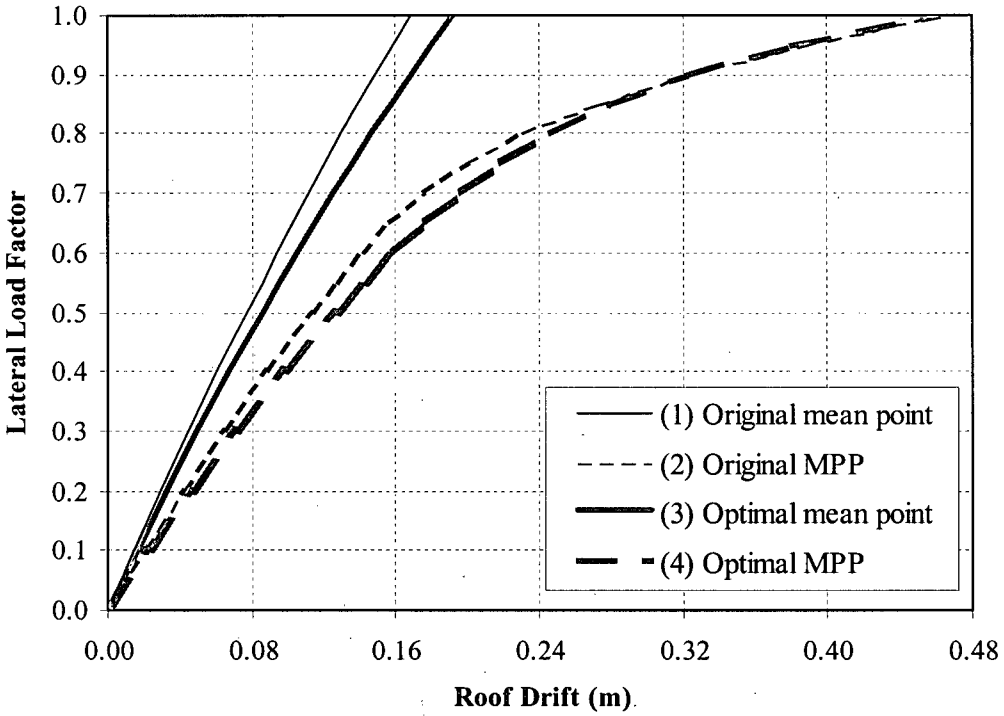

Figure 7.8 Structural responses for Case 3 (load factor versus roof displacement) at: (1) the mean point of the initial design; (2) the MPP of the initial design; (3) the mean point of the optimal design; and (4) the MPP of the optimal design.

Figure 7.8 shows the structural response of four characteristic realizations of the design variables and random variables. The responses at the mean realization of the random variables and at the MPP of the random variables for the initial (original) design are shown as the thin solid line and the thin dashed line, respectively. The structural responses at the mean realization of the random variables and at the MPP of the random variables for the optimal design are shown as the thick solid line and the thick dashed line, respectively. The figure shows similar properties to the nonlinear case, using *beamWithHinges* elements in Case 2.

Table 7.12 Comparison of computational time for Case 3

	DSA-MOOA		DSA-S	
	DDM	FDM	DDM	FDM
g	5471	40415	1300	7618
g with $\partial g/\partial \mathbf{u}$	91	N/A	33	N/A
g with $\partial g/\partial \mathbf{x}$	1547	N/A	208	N/A
Importance Sampling	109436		101848	

Table 7.12 compares the efficiency of the FDM and DDM methods, as well as the computation cost of the two implemented approaches, by measuring the number of calls to the limit state function. We came to the same conclusion as in the linear case: using the DDM method to compute the gradients is much more efficient than using the FDM method, regardless of which optimization approach is adopted.

As shown in Table 7.12, the DSA-S approach, requiring 1,541 (1,300+33+208) limit-state function calls, appears to be more efficient than the DSA-MOOA approach, requiring 7,109 (5,471+91+1547) limit-state function calls using the DDM method. This nonlinear case requires much more computational effort than the linear case, 213 and 982 limit-state function calls, respectively. However, the number of limit-state function calls is almost the same as that required in the nonlinear Case 2.

7.2 Practical Experience from Case Studies

This section describes in further detail the observations and practical experiences that have been gained from the case studies presented above. Comparisons are made between two implemented optimization approaches (DSA-MOOA and DSA-S), between the FDM and the DDM methods, and between linear and nonlinear pushover analyses. We also make the observation that the convergence of the optimization procedure is significantly improved by removing inactive constraints or by properly scaling the functions involved.

7.2.1 Comparison of Two Optimization Approaches

Both the DSA-MOOA and the DSA-S approaches are gradient-based algorithms and decoupled sequential optimization approaches. The reliability analysis and the optimization analysis are decoupled in them, so the users have the flexibility to choose any available reliability methods and optimization algorithms according to their requirements. However, the two approaches have different behaviours with regards to convergence performance and computational time.

The two approaches use the same problem reformulation. The original problem and the reformulated problem are proved to be identical in the first-order approximation. The DSA-MOOA approach considers the reformulated problem as a semi-infinite optimization problem and solves it using the MOOA algorithm, which has a converged solution when using an infinite number of reliability constraints. On the other hand, the DSA-S approach considers the reformulated problem as an inequality constraint optimization problem and solves it using the Polak-He algorithm, which can only find a consistent design without the proof of convergence. According to case study results, the two approaches can achieve the same solution if the analysis is stopped at the same tolerance.

When the two approaches' convergence speed is compared, it can be seen that the DSA-S approach is much faster than the DSA-MOOA approach: the former only needs

~20% of limit-state function calls of the latter. The DSA-S approach solves the final optimization problem using a single reliability constraint, while the DSA-MOOA approach expands the reliability constraints step by step by discretizing the ball with progressively more points to achieve a gradually precise solution. Hence, 80% of the computational time in the DSA-MOOA approach is used to deal with the discretization of points and a progressively larger reformulated problem.

In conclusion, the DSA-S approach is effective and accurate enough. However, if this approach fails in the converge procedure, the user has to rely on the DSA-MOOA approach, which is reliable but slow.

7.2.2 Comparison of Two Gradient Computation Methods

Two methods of computing response sensitivities in OpenSees are used in case studies: the FDM and the DDM. This section compares the two methods in light of three requirements: consistency, accuracy, and efficiency.

Consistency refers to the computed sensitivities being consistent with the approximations made in computing the response itself. In the DDM, consistency is ensured through differentiating time- and space-discretized finite element response equations (Haukaas & Der Kiureghian, 2005). The computation of the structural response and the response gradient are both conducted in the finite element analysis. On the other hand, the FDM simply computes the ratio of the structural response difference and perturbation.

Accuracy is important to response sensitivity, since the convergence of reliability and optimization algorithms depend on it. The sensitivities computed by ordinary finite difference may not be sufficiently accurate to guarantee convergence of the solution algorithms (Haukaas & Der Kiureghian, 2005). The DDM ensures better accuracy than the FDM, since the DDM evaluates the exact derivatives of the approximate finite element response.

Efficiency is an important requirement in computation, since sensitivities are repeatedly computed in the solution algorithms. In the DDM, the response sensitivities for each parameter are obtained from a linear equation upon convergence of the finite element response (Haukaas & Der Kiureghian, 2005). The additional time used by the gradient calculation in the DDM is less than the time required by another nonlinear finite element analysis. Instead, the FDM method requires one more nonlinear finite element analysis with perturbed parameter values for each random variable and design variable. Hence, the computational time by the FDM equals the number of random/design variables multiplied by the computational time for a single finite element analysis.

The DDM method requires a one-time consolidated effort to derive differentiation equations and implement them in the finite element program. However, once we have it, the DDM method is more accurate and efficient than the FDM method.

7.2.3 Comparison of Linear and Nonlinear Analyses

Usually the users choose between linear or nonlinear pushover analyses according to their requirements and their analysis ability. The comparison in this section shows the possible results and computational cost for each selected case. This comparison can serve to guide the users when making the decision about which analysis method to choose.

Table 7.13 presents the comparison between initial and optimal reliability indexes for the three cases. All three initial reliability indexes are greater than 3.0 regardless of the analysis model. This implies that the initial design is safe but may not be optimal. Following the RBDO analysis, the reliability indexes go down to 3.0, which is the lower bound of the reliability constraint.

The optimal total costs of nonlinear cases are lower than those for the linear case. This is reasonable, since the linear analysis is based on the "equal displacement principle" and results in equivalent results, while nonlinear analyses offer more "exact" results. However, from the structural design point of view, the results of the linear analysis are also acceptable.

Table 7.13 Comparison of linear and nonlinear cases

	Case 1: Linear	Case 2: Nonlinear (*beamWithHinges*)	Case 3: Nonlinear (*dispBeamColumn* + *fibre*)
Initial reliability index	3.646	3.536	3.121
Optimal reliability index	3.0	3.0	3.0
Initial total cost	1.0	1.0	1.0
Optimal total cost	0.715	0.688	0.651
Number of top level of iteration	1	3	3
Number of limit-state function calls in first iteration	982	7047	7109

We have also compared computational costs for linear and nonlinear cases. In the linear case, the DSA-MOOA and the DSA-S analyses require only one iteration to achieve an acceptable design, while in the nonlinear cases three iterations are required. In the first iteration the linear case calls 982 limit-state function calculations, which is about 15% of the number of limit-state function calls required by the nonlinear analysis (about 7,000 calls). Hence, the linear analysis is much more effective than the nonlinear analysis. For the linear case, with the updating of the design the corresponding MPP of the reliability analysis does not change dramatically. In addition, there are only five reliability constraints after 75 loops of tasks B1 to B3. On the other hand, the MPP of the reliability analysis for the nonlinear cases changes apparently when a new design is found. In addition, there are 12 to 17 reliability constraints after 75 loops of tasks B1 to B3.

In conclusion, nonlinear analyses produce "exact" and trustworthy optimal designs, while the linear case is effective and its results are also acceptable. It is advisable to conduct a linear analysis for an optimal design first. If the user really needs a more "exact" design, the nonlinear analysis can start from the results of the linear analysis.

7.2.4 Active and Inactive Constraints

There are three categories of constraints in the reformulated optimization problem: the deterministic constraints $\mathbf{f}(\mathbf{x}) \le 0$, reliability constraints $\psi \le 0$, and constraints $p_f \le \hat{p}_f$. At the optimal design point, the limit-state function is $g(\mathbf{d}(\mathbf{x},\mathbf{u})) = 0$, which satisfies the reliability constraints $\psi = -g \le 0$, making these reliability constraints active. In Tables 7.4, 7.7, and 7.11, all failure probabilities at the optimal design reach the upper bound $\hat{p}_f$ =0.00135, so this constraint is also active.

However, by observing the optimal results, we find that some of the deterministic constraints are not active. For example, in the nonlinear case using *dispBeamColumn* elements and *fibre* sections only two types of constraints are active in six types of constraints. In this case, we define the following six constraints:

$$\begin{array}{ll} b > 0 & h > 0 \\ b/h \le 2 & b/h \ge 0.5 \\ As \ge \rho_{\min} bh & As \le \rho_{\max} bh \end{array} \qquad (7.4)$$

where $\rho_{\min}$ and $\rho_{\max}$ are the lower and upper bounds of longitudinal reinforcement ratios. Only two categories, $b/h \ge 0.5$ and $As \ge \rho_{\min} bh$, are active. When all of the inactive deterministic constraints are removed, the final results are the same as those of the full constraints, but the computational time is reduced to about 60-80% of the original time. The reduction in time stems from the reduced size of vector **g** and matrix **G** in the LSSOL analysis. In the two implemented approaches, the time is saved in tasks B3 and D1.

In summary, removing inactive constraints can speed up the optimization procedure. Once the user finds that some inactive constraints are violated in the analysis, the user can stop the analysis and add the constraints back into the optimization to make sure that the solutions are correct.

7.2.5 Acceleration of Convergence Procedure by Proper Scaling

Both tasks B1/D2 and B3/D1 apply the Polak-He algorithm to solve an optimization problem. The Polak-He algorithm requires the computation of the values and gradients of the objective function, deterministic constraints, and reliability constraints. It has been observed that the different ways to define these functions can affect the convergence speed.

Without scaling, the objective function is about $50m^3$ to $130m^3$, the deterministic constraints are between 0.5 and 2, and the reliability constraints are about 0.2 to 1.0. These values are not in the same order of magnitude. For the nonlinear analysis, task B1/D2 requires about 1 to 10 iterations to converge, while task B3/D1 requires several hundred or even thousand iterations to converge. This convergence speed is not acceptable. If a scaling is applied (scaling all involved functions—objective function, deterministic constrains, and reliability constraints) to approximately the same order 10^0 = 1.0, the new convergence performance in task B1/D2 remains same, but the computational cost of task B3/D1 is reduced to less than one hundred iterations (and often less than 10 iterations). Originally, the functions involved in task B1/D2 had the same order, so scaling did not affect these tasks. The benefit of task B3/D1 is apparent because only 10% of the original computational time is required.

The gradients of objective and constraint functions cannot be scaled directly. It may not be possible to conduct proper scaling for both the functions and their gradients. Scaling the values of functions to 1.0 can approximately scale the gradient in the order of 10^0. This is better than the value of function in the order 10^2 and the gradient in the order 10^4. The bigger the difference between the vector and the matrix cells in LSSOL, the more "ill-conditioned" the problem becomes. This is a general problem that cannot be fixed easily. It is recommended to do scaling in the beginning of the process, when defining all of the functions.

As mentioned in Chapter 6, another scaling skill can be used to speed up the convergence procedure. Parameter $b = -\Phi^{-1}(a)$ is used in place of parameter a. With

reference to Eq. (2.9), parameter b is a substitute for the reliability index β, in the same way as parameter a is a substitute for failure probability $p_f(\mathbf{x})$. This parameter replacement avoids numerical difficulties caused by the difference in orders of magnitude between a and other design variables $\mathbf{x}$. Hence, the optimization in the DSA-MOOA approach is over the design vector $(\mathbf{x}, b)$.

In summary, the convergence procedure in the RBDO can be accelerated by properly scaling the functions involved and by using the reliability index to take the place of the failure probability in the optimization. The benefit of this scaling is apparent, because it only takes 10% of the original time.

Chapter 8 Conclusions

8.1 Summary of Major Findings

In this thesis we implement two reliability-based design optimization (RBDO) approaches in the object-oriented software framework OpenSees. The total expected cost is minimized in the optimization process subject to constraints on structural properties and component structural reliability. The implementations comprise a merger between reliability, optimization, and finite element techniques. This enables the RBDO of comprehensive real-world structures that exhibit nonlinear behaviour. Our work provides a tool for engineers in making rational decisions based on the balance between cost and safety in engineering practice.

The fact that the failure probability in terms of a high-dimensional integral is involved in the objective and constraint functions violates two basic requirements of the standard nonlinear optimization solver: all involved functions must be evaluated in a finite time and must be continuously differentiable. In this thesis, the first requirement is satisfied by evaluating the probability of failure using efficient approximation methods such as the first-order reliability method and importance sampling. The second requirement is satisfied by making use of smooth material model and discretized cross-sections.

This thesis proposes two decoupled optimization approaches: the DSA-MOOA and the DSA-S. These are efficient, robust, and versatile tools for solving RBDO problems. In both of them, the required reliability and optimization calculations are decoupled, thus allowing flexibility in choosing optimization algorithms and reliability computation methods. The original optimization problem is reformulated as a deterministic optimization problem, which is identical to the original problem in the first order case. The DSA-MOOA considers the reformulated problem as a semi-infinite optimization problem and solves it using the method of outer approximation. The DSA-S considers the reformulated problem as an inequality constrained optimization problem and solves it using the Polak-He algorithm. This simplified algorithm is demonstrated to provide the same optimal design as the DSA-MOOA approach for the cases considered.

The solutions of the two approaches are based on the first-order approximation to the failure probability. If the users accept this failure probability, the analysis is stopped, resulting in an optimal design. However, for the problem with nonlinear limit-state function, the new design and failure probabilities are updated using a higher-order reliability method (such as importance sampling) to take into account the nonlinear structural behaviour. The parameter t is employed to fulfill the nonlinearity approximation process. Starting from unity, the parameter t is updated during the optimization analysis to account for the nonlinearity in the limit-state function.

An effective, accurate, and consistent response sensitivity analysis is essential in gradient-based optimization algorithms. For the finite element analysis, we need the gradients of the structural response with respect to model parameters. Two gradient evaluation methods, the finite difference method (FDM) and the direct differentiation method (DDM), are employed in this thesis. The DDM method requires the derivation and implementation of analytical derivatives of the finite element response. Once we have the DDM implementation in the finite element software, the DDM method is more accurate and efficient than the FDM method, which requires an additional nonlinear finite element analysis for each random variable and design variable.

The key difficulty resolved in the implementation is the negative effect of response gradient discontinuities due to sudden yielding events. The possible response gradient discontinuities for nonlinear structures cause non-convergence or slow convergence in the optimization analysis as well as in the first order reliability analysis. Two remedies are applied in this thesis: the smooth material model builds a "continuously differentiable" response, and the section discretization scheme results in an "approximately continuously differentiable" response. Hence, the requirement of standard nonlinear optimization algorithms is satisfied and the nonconvergence problem is avoided.

The new implementations results in a modern and comprehensive software, OpenSees, with RBDO capacities. Object-oriented programming was utilized when extending OpenSees' reliability, optimization, and sensitivity capabilities. The superior extensibility

and maintainability features of this programming type are emphasized. Originally, OpenSees has employed four principal objects—*ModelBuilder*, *Domain*, *Analysis,* and *Recorder*—to perform the finite element analysis. OpenSees is then extended with the *ReliabilityDomain* and the *ReliabilityAnalysis* object to perform reliability analysis. This thesis further extends OpenSees with optimization capacities by adding several objects to *ReliabilityDomain* and establishing two new analysis types, *DSA-MOOAAnalysis* and *DSA-SAnalysis* objects. The reliability domain contains all functions involved in the optimization problem and maps the design variables into the finite element model. The analysis part includes two RBDO approaches and several analysis tools. The extended OpenSees has the capacity to perform finite element analysis, reliability and sensitivity analyses, and the optimization analysis for comprehensive real-world structures exhibiting nonlinear behaviour.

A numerical example involving a nonlinear finite element analysis of a three-bay, six-storey building is used to demonstrate the implementations. In particular, the need for a continuously differentiable response with respect to the finite element model parameters is emphasized. The linear pushover analysis using *elasticBeam* elements does not encounter any convergence problems in optimization. The nonlinear pushover analysis using *beamWithHinges* elements cannot converge using the traditional bi-linear steel material model. This issue is cured by using the smooth steel material model, in which a circular segment starting at 80% of the yield strength is employed to smooth the bi-linear material. The nonlinear pushover analysis using *dispBeamColumn* elements with *fibre* sections avoids the non-convergence issue by utilizing smooth steel materials and discretized concrete fibre sections.

The observations and practical experiences are summarized following numerical case studies. It was found that the DSA-MOOA and the DSA-S can achieve the same solution. Yet, while the DSA-MOOA can theoretically prove its convergence, the DSA-S cannot. When convergence speeds were compared, it was found that the DSA-S only needed about 20% of limit-state function calls required by the DSA-MOOA. Thus, the DSA-S

can find an optimal design more effectively. However, if this algorithm fails in the convergence procedure, the user must use the more reliable DSA-MOOA.

It was observed that the linear case required only 15% of limit-state function calls used by nonlinear analyses. Nonlinear analyses produced more "exact" optimal designs, but the linear case was more effective while also producing acceptable results. It is thus suggested that the linear analysis be conducted first to find an optimal design. If a more "exact" design is then required, the nonlinear analysis can begin from the results of the linear analysis.

Some of the deterministic constraints were observed to be inactive in the optimization process. Removing them can speed up the optimization procedure and save about 20-40% of the original computational time. If the user finds that some inactive constraints are violated in the analysis, the user can stop the analysis and add these constraints back into the optimization again to make sure the solutions are correct.

Finally due to the use of the Polak-He algorithm, which only has linear convergence properties, it is necessary to scale all involved functions properly (including the objective function, deterministic constrains, and reliability constraints) to approximately the same order $10^0 = 1.0$. The benefit of this scaling is apparent in the optimization loop, since only 10% of the original computational time is used. Another scaling is also suggested, namely using the reliability index instead of the failure probability as the auxiliary variable in the analysis. This is because the failure probability is too small and does not appear in the same order of magnitude as the design variables. In summary, the convergence in the RBDO can be accelerated by properly scaling the involved functions and by using the reliability index instead of the failure probability.

8.2 Further Studies

A real-world structure is actually a general system reliability problem. Der Kiureghian and Polak (1998) first attempted to deal with the series structural system, and the RBDO for series reliability system were finally solved in Royset et al. (2001a, 2001b, 2002, &

2004a). Based on the experience obtained in this thesis, the implementation of series system problems in the reliability constraints can be achieved in the future work.

The analysis in this work is limited to static pushover finite element analysis. When considering dynamic finite element analysis, time-variant reliability analysis must be used to evaluate the failure probability. An available time-variant reliability analysis method is the mean out-crossing reliability analysis. Furthermore, cyclic loading may cause the degradation of the structural response. The application of RBDO to such problems represents an important challenge for further work.

The definitions of initial costs and future costs are here made in terms of structural volume or weight. More detailed cost computations are desirable. For instance, it is of interest to include the present cost of future events. Such realistic considerations could be an interesting future study.

In this thesis, we noticed the importance of proper scaling, which speeds up the convergence procedure and avoids the nonconvergence problem. However, we only scaled the values of the involved functions at the beginning of the analysis. In addition, we did not know how to scale the gradient properly. It is thus suggested to develop an automatic scaling scheme to scale both the value and the gradient, and scaling them at each optimization step in future work.

Bibliography

Agarwal, H. and Renaud, J. E. (2004). "Decoupled methodology for probabilistic design optimization." *Proceedings of 9th ASCE Joint Specialty Conference on Probabilistic Mechanics and Structural Reliability*, July, Albuquerque, New Mexico.

Agarwal, H., Renaud, J. E., and Mack, J. D. (2003). "A decomposition approach for reliability-based multidisciplinary design optimization." *Proceeding of the 44th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials (SDM) Conference,* Norfolk, VA.

Benjamin, J. R. and Cornell, C. A. (1970). *Probability, statistics, and decision for civil engineers.* McGraw-Hill, New York.

Bjerager, P. and Krenk, S. (1989). "Parameter sensitivity in first order reliability theory." *Journal of Engineering Mechanics*, 115(7), 1577–1582.

Breitung, K. (1984). "Asymptotic approximation for multinormal integrals." *J. Engineering Mechanics*, 110(3), 357–366.

Cement Association of Canada. (1995). *Canadian Concrete Design Handbook.* 2nd Edition, third printing November 2001.

Chen, X., Hasselman, T., and Neill, D. (1997). "Reliability-based structural design optimization for practical applications." *American Institute of Aeronautics and Astronautics*, AIAA-97-1403, 2724–2732.

Deitel, H. M. and Deitel, P. J. (1998). *C++ How to program.* Prentice Hall, Inc., Upper Saddle River, NJ.

Der Kiureghian, A., Zhang, Y. and Li, C.C. (1994). "Inverse reliability problem." *Journal of Engineering Mechanics*, ASCE 120:1154-9.

Der Kiureghian, A. and Polak, E. (1998). "Reliability-based optimal design: A decoupled approach." *Reliability and Optimization of Structural Systems*, A.S. Nowak (Ed.), Book Crafters, Chelsea, Michigan.

Ditlevsen, O. and Madsen, H. (1996). *Structural reliability methods*. Wiley, New York, New York.

Du, X. and Chen, W. (2002). "Sequential optimization and reliability assessment method for efficient probabilistic design." *ASME Design Engineering Technical Conference, 28th Design Automation Conference*. September.

Eldred, M. S., Giunta, A. A., Wojtkiewicz, S. F., and Trucano, T. G. (2002). "Formulations for surrogate-based optimization under uncertainty." *Proceedings of the 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization,* Paper AIAA-2002-5585, Atlanta, Georgia.

Enevoldsen, I. and Sorensen, J. (1994). "Reliability-based optimization in structural engineering." *Structural Safety*, 15(3), 169–196.

Gasser, M. and Schueller, G. (1998). "Some basic principles in reliability-based optimization (RBO) of structures and mechanical components." *Stochastic programming methods and technical applications*, K. Marti and P. Kall (Eds.), Lecture Notes in Economics and Mathematical Systems 458, Springer-Verlag, Berlin, Germany.

Gill, P.E., Hammarling, S.J., Murray, W., Saunders, M.A. and Wright, M.H. (1986). *User's Guide to LSSOL. A Fortran Package for Constrained Linear Least-Squares and Convex Quadratic Programming*. Technical Report SOL 86-1. Califorlia: Stanford Optimization Laboratory, Stanford.

Gill, P., Murray, W., Saunders, M., and Wright, M. (1998). *User's guide to NPSOL 5.0: A Fortran package for nonlinear programming*. Report No. SOL-86-1, System Optimization Laboratory, Stanford University, Stanford, California.

Gobbo, B. (1999). "Calling Fortran routines from a C++ program in Unix." *http://pccosrv1.cern.ch/compass/software/offline/software/fandc.html*.

Hasofer, A. M. and Lind, N. C. (1974). "Exact and invariant second-moment code format." *Journal of Engineering Mechanics*, 100(1), 111–121.

Haukaas, T. and Der Kiureghian, A. (2004). *Finite Element Reliability and Sensitivity Methods for Performance-Based Engineering*. Report No. PEER 2003/14. California: Pacific Earthquake Engineering Research (PEER) Center, University of California, Berkeley.

Haukaas, T. and Der Kiureghian, A. (2005). "Parameter sensitivity and importance measures in nonlinear finite element reliability analysis." Accepted for publication, *ASCE Journal of Engineering Mechanics*.

Hohenbichler, M. and Rachwitz, R. (1986). "Sensitivity and importance measures in structural reliability." *Civil engineering systems*, 3, 203-209.

Itoh, Y. and Liu, C. (1999). "Multiobjective optimization of bridge deck maintenance." *Case Studies in Optimal Design and Maintenance Planning if Civil Infrastructure Systems*, D.M. Frangopol (Ed.), ASCE, Reston, Virginia.

Kirjner-Neto, C., Polak, E., and Der Kiureghian, A. (1998). "An outer approximations approach to reliability-based optimal design of structures." *J. Optimization Theory and Application*, 98(1), 1–17.

Kuschel, N. and Rackwitz, R. (2000). "A new approach for structural optimization of series system." *Proceedings 8th Intern. Conf. On Applications of Statistics and Probability (ICASP) in Civil Engineering Reliability and Risk Analysis*, R.E. Melchers and M.G. Stewart (Eds.), Sydney, Australia.

Li, H. and Foschi, R.O. 1998. "An inverse reliability method and its application." *Structural Safety*. 20 (3) 257-270.

Liu, P.-L. and Der Kiureghian, A. (1986). "Multivariate distribution modes with prescribed marginals and covariances." *Probabilistic engineering mechanics*, 1(2), 105-112.

Madsen, H. and Friis Hansen, P. (1992). "A comparison of some algorithms for reliability-based structural optimization and sensitivity analysis." *Reliability and*

Optimization of Structural Systems, Proceedings IFIP WG 7.5, R. Rackwitz and P. Thoft-Christensen (Eds.), Springer-Verlag, Berlin, Germany.

Matlab Mathworks, Inc. (1999). *Matlab reference manual, Version 5.3, Release 11.* Math-Works, Inc., Natick, Massachusetts.

Mazzoni, S., McKenna, F., Fenves, G.L. & Scott, M.H. (2005). "OpenSees command language manual." *http://opensees.berkeley.edu/.* Pacific Earthquake Engineering Research Center, University of California, Berkeley, CA.

McKenna, F., Fenves, G.L. & Scott, M.H. (2004). "OpenSees: Open System for Earthquake Engineering Simulation." *http://opensees.berkeley.edu/.* Pacific Earthquake Engineering Research Center, University of California, Berkeley, CA.

Polak, E. (1997). *Optimization. Algorithms and consistent approximations.* Springer-Verlag, New York, New York.

Rackwitz, R. and Fiessler, B. (1978). "Structural reliability under combined load sequences." *Computers and structures*, 9, 489-494.

Royset, J., Der Kiureghian, A., and Polak, E. (2001a). "Reliability-based optimal design of series structural systems." *J. Engineering Mechanics*, 127(6), 607–614.

Royset, J., Der Kiureghian, A., and Polak, E. (2001b). "Reliability-based optimal structural design by the decoupling approach." *J. Reliability Engineering and System Safety*, 73(3), 213–221.

Royset, J.O., Der Kiureghian, A. & Polak, E. (2002). *Reliability-based Design Optimization of Series Structural Systems.* Roport No. UCB/SEMM-2002/15. California: Department of Civil and Environment Engineering, University of California, Berkeley.

Royset, J.O., Der Kiureghian, A. & Polak, E. (2004a). "Optimal Design with Probabilistic Objective and Constraints." *Journal of Engineering Mechanics*, submitted.

Royset, J.O. and Polak, E. (2004b). "Reliability-based optimal design using sample average approximations." *Probabilistic engineering mechanics*, 19 (2004), 331-343.

Schittkowski, K. (1985). *User's guide to nonlinear programming code, handbook to optimization program package NLPQL.* University of Stuttgart, Stuttgart, Germany.

Sexsmith, R. G. (1983). "Bridge risk assessment and protective design for ship collision." *IABSE Colloquium Copenhagen 1983 - Ship Collision with Bridges and Offshore Structures, Preliminary Report*, V42, 425-433, Copenhagen, Denmark

Thampan, C. and Krishnamoorthy, C. (2001). "System reliability-based structural configuration optimization of trusses." *J. Structural Engineering*, 127(8), 947–955.

Torczon, V. and Trosset, M. (1998). "Using approximations to accelerate engineering design optimization." *Proceedings of the 7th AIAA/USAF/NASA/ISSMO Symp. on Multidisciplinary Analysis and Optimization*, AIAA Paper 98-4800, St. Louis, Missouri.

Wang, L., Kodiyalam, S. (2002). "An efficient method for probabilistic and robust design with non-normal distribution." *Proceeding of the 43rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference.* April.

Welch, B. B. (2000). *Practical programming in Tcl and Tk.* Prentice Hall, Inc., Upper Saddle River, New Jersey, 3rd edition.

Zhang, Y. and Der Kiureghian, A. (1997). *Finite element reliability methods for inelastic structures.* Report No. UCB/SEMM-97/05, Dept. of Civil and Environmental Engineering, University of California, Berkeley, Berkeley, California.

Appendix A: Detailed Software Implementation

This appendix contains detailed software implementations of the reliability-based design optimization (RBDO). These implementations include ways to call Fortran routines from C++, methods of building LSSOL.LIB, and prerequisites to extending OpenSees with the RBDO capabilities.

A1: Calling Fortran Routines from C++

In the Polak-He algorithm, a Fortran 77 program (LSSOL) is used to solve a sub-optimization problem and to find the search direction. Therefore, a technique to call Fortran routines from OpenSees (in C++) is required. This section introduces several methods to implement this mix-program technique, focusing especially on how to pass and return variables and arrays between C++ and Fortran.

1. The extern "C" directive is used to declare the external Fortran subroutine LSSOL in C++.

```
#ifdef _WIN32
extern "C" void LSSOL(int *m, double *c, double *A, double *obj, double *x);
#else
extern "C" void lssol_( int *m, double *c, double *A, double *obj, double *x);
#endif
```

   where *m* is an input variable, c is an input one-dimensional array, and A is an input two-dimensional array. They are passed from C++ to Fortran. *obj* is a returned variable and x is a returned one-dimensional array. Both of them are returned from Fortran to C++. Note that variables and arrays listed above are only examples and are not complete.

2. When C++ calls Fortran, the reference to Fortran symbols is specified in lowercase letters, since C++ is a case sensitive language, but Fortran is not.

3. C++ passes the variables by value, while Fortran passes them by reference. It is necessary to specify in the C++ that the Fortran subroutines expect *call-by-reference* arguments using the *address-of* operator & (Gobbo, 1999). An example of passing the variable m = 10 from C++ to Fortran and returning the variable *obj* = 20.0 from Fortran to C++ is below:

```
// Define the variable m and obj
m = 10;
obj = 0.0;
// Call LSSOL
#ifdef _WIN32
    LSSOL(&m, &obj);
#else
    lssol_(&m, &obj);
#endif
```

As a returned variable, *obj* = 20.0 is then used directly in C++.

4. C++ passes arrays using pointers, while Fortran passes them using references. In addition, C++ stores arrays in a row-major order, whereas Fortran stores arrays in a column-major order. Finally, the lower bound for C++ is 0, but for Fortran it is 1 (Gobbo, 1999). For instance, given an array "fun," the Fortran array element fun(1,1) is the same as the C++ array element fun[0][0]; the Fortran array element fun(6,8) corresponds to the C++ array element fun[7][5]. An example of passing one-dimensional array c = $\begin{bmatrix}1.1 & 1.2\end{bmatrix}$ and two-dimensional array A = $\begin{bmatrix}2.1 & 2.3\\ 2.2 & 2.4\end{bmatrix}$ from C++ to Fortran and returning one-dimensional array x = $\begin{bmatrix}3.1 & 3.2\end{bmatrix}$ from Fortran to C++ is below:

```
// Preparing input data for LSSOL
c[0] = 1.1;     c[1] = 1.2;
A[0] = 2.1;     A[1] = 2.2;     A[2] = 2.3;     A[3] = 2.4;
x[0] = 0.0;     x[1] = 0.0;
//  Call LSSOL
#ifdef _WIN32
```

```
    LSSOL(c, A, x);
#else
    lssol_(c, A, x);
#endif
```

As a returned array, x[0] = 3.1 and x[1] = 3.2 is then used directly in C++.

A2: Building LSSOL.LIB

LSSOL is complied using Intel(R) visual Fortran compiler for Windows, standard edition, which is freely available from the Intel website *http://www.intel.com/software/products/compilers/downloads/forwin.htm*. Compiling LSSOL requires the following procedure:

1. Download and install "Microsoft Visual Studio.NET" and "Intel(R) Software Development Tools."
2. Run "Microsoft Visual Studio.NET." Set the project type, template, and name of project workspace in the following way:

   File → New → Project …
   Project Types: Intel (R) Fortran Projects
   Templates: Static library
   Name: LSSOL
3. Set the library wizard when you see "Welcome to the Fortran static library wizard." Make sure that the option "Prevent the insertion of linker directives for defaults libraries" is not selected.

   Library Settings: Additional features
   □ Prevent the insertion of linker directives for defaults libraries
4. Copy LOSSL files to folder "..\LSSOL," and add LSSOL files into the workspace.

   Solution Explorer – LSSOL
   LSSOL → Source Files → Add → Add Existing Item … → Add all files
5. Set Fortran libraries.

LSSOL → Properties → Fortran → Libraries:

Use Common Windows Libraries:	Yes
Use Portlib Library:	Yes
Disable Default Library Search Rules:	No

6. Compile LSSOL through "Build → Build LSSOL." The result is a library LSSOL.LIB (Size: 1,337 KB), which is saved in the folder "..\LSSOL\debug."
7. Backup four LIB files below from "Intel(R) Software Development Tools." The following libraries are required when compiling OpenSees.

IFCONSOL.LIB	(Size:	7 KB)
IFWIN.LIB	(Size:	30 KB)
LIBIFCORE.LIB	(Size:	955 KB)
LIBIFPORT.LIB	(Size:	412 KB)

A3: Extending OpenSees with RBDO Capacity

Install Tcl/Tk (which is needed to run OpenSees)

1. Uninstall any previous versions of Tcl
2. Download the installation file for Tcl/Tk
3. Run the installation by running the "exe" file downloaded in step 2
4. Install Tcl in the folder **C:\Program Files\Tcl**
5. Restart the computer

Install CVS software (required to download OpenSees from the CVS repository)

1. Download files cvs-1-11-5.zip and cvslogin.bat
2. Unzip cvs-1-11-5.zip and run the installation file cvs-1.11.5.exe
3. Restart the computer

Download OpenSees code from the "official" CVS repository in Berkeley

1. Open a DOS window (*e.g.*; Start > Programs > Accessories > Command Prompt)
2. "cd" into the folder where you have put the "cvslogin.bat" file
3. Execute "cvslogin" command
4. Note that steps 2 and 3 above can be replaced by issuing the following commands:

   set CVS_RSH=ssh
   set CVSROOT=:pserver:anonymous@opensees.berkeley.edu:/usr/local/cvs
   cvs login

5. When prompted, provide the password "anonymous"
6. Go to the directory where you want to put the OpenSees code
7. Give the command "cvs checkout OpenSees"

Later, when updating the code with the most recent changes in the CVS repository, you can follow steps 1 to 6 and then give the command "cvs -q update -d" (-q is used to suppress output, -d is used to check out any new directories). It may be a good idea to do this "directory by directory" in the SRC directory. The command "cvs diff" is used to list differences between local files and the CVS repository files. When doing updates, the following abbreviations are used to identify the actions taken for each file:

M -- local copy has been modified
P -- merged changes on server with the local copy
C -- conflict with what's on server and the local copy
U -- check a new file that is not part of local copy

Compile the "official" OpenSees version

1. Make sure the "include path" for *tcl.h* is correct in the projects *damage, database, domain, element, material, recorder, reliability,* and *openSees* by doing the following:

   a) Right-click on the project and choose "Settings > C/C++ > Preprocessor"

b) Select "Settings for: All Configurations"

c) In "Additional include directories" the last statement should be "c:\Program Files\tcl\include"

2. Include *tcl84.lib* in the *openSees* project by doing the following:

a) Right-click on the project and choose "Settings > Link > Input"

b) Select "Settings for: All Configurations"

c) In "Additional library paths," include "c:\Program Files\tcl\lib" (make sure to have a comma between the different paths)

Press F7 to compile. If many error messages appear, try to press F7 again to "clean up."

Add new and/or improved files from the UBC team

Put all files listed in Tables A.1a, A.1b, and A.1c into their respective directories. For new classes, remember to include the files in the appropriate project according to the "location of file" provided in the Tables A.1a, A.1b, A.1c,.

How to identify the difference between local files and the "official" version at Berkeley

There are two ways of identifying the difference between the files that have been modified by the "UBC team" and the official Berkeley files:

1. Download the files, include them in local OpenSees version, and use the "diff" feature of CVS to see the differences. (Give the command "cvs diff" in the relevant directory.)

2. Search for the text string "UBC Team." All UBC team modifications are marked with this stamp.

Include the Fortran library "LSSOL.LIB"

1. Copy the following files into the directory: Win32/lib: IFCONSOL.LIB, IFWIN.LIB, LIBIFCORE.LIB, and LIBIFPORT.LIB.
2. Copy the file LSSOL.LIB into the Win32/lib/debug and Win32/lib/release directories.
3. Add the LSSOL library to the project settings by doing the following:
   a) Right-click on the "opensees" project and choose "Settings > Link > General"
   b) Select "Settings for: All Configurations"
   c) In "Object/library modules" add the filename LSSOL.LIB

Table A.1a New and modified classes for extending RBDO (Classes that do not exist in the "official" version are marked with *)

Project	Location of file	Files
		classTags.h
OpenSees	Source	commands.cpp
	Header	commands.h
Reliability	analysis/types	DSA_MOOAOptimizationAnalysis.cpp* DSA_MOOAOptimizationAnalysis.h* DSA_SOptimizationAnalysis.cpp* DSA_SOptimizationAnalysis.h*
	domain/components	ReliabilityDomain.cpp ReliabilityDomain.h ConstraintFunction.cpp* ConstraintFunction.h* CostFunction.cpp* CostFunction.h* DesignVariable.cpp* DesignVariable.h* DesignVariablePositioner.cpp* DesignVariablePositioner.h* ObjectiveFunction.cpp* ObjectiveFunction.h*

Table A.1b New and modified classes for extending RBDO (continued) (Classes that do not exist in the "official" version are marked with *)

Project	Location of file	Files
Reliability	analysis/designPoint	NonlinSingleIneqOpt.cpp* NonlinSingleIneqOpt.h* PolakHeNonlinSingleIneqOpt.cpp* PolakHeNonlinSingleIneqOpt.h* NonlinMultiIneqOpt.cpp* NonlinMultiIneqOpt.h* PolakHeNonlinMultiIneqOpt.cpp* PolakHeNonlinMultiIneqOpt.h* LinMultiIneqOpt.cpp* LinMultiIneqOpt.h* LSSOLLinMultiIneqOpt.cpp* LSSOLLinMultiIneqOpt.h*
	analysis/gFunction	GFunEvaluator.cpp GFunEvaluator.h OpenSeesGFunEvaluator.cpp OpenSeesGFunEvaluator.h
	analysis/sensitivity	GradGEvaluator.h FiniteDifferenceGradGEvaluator.cpp FiniteDifferenceGradGEvaluator.h OpenSeesGradGEvaluator.cpp OpenSeesGradGEvaluator.h
	FEsensitivity	SensitivityAlgorithm.cpp
	tcl	TclReliabilityBuilder.cpp
Element		Information.cpp TclElementCommands.cpp
	dispBeamColumn	DispBeamColumn2d.cpp
	beamWithHinges	BeamWithHinges2d_bh.cpp* BeamWithHinges2d_bh.h* TclBeamWithHingesBuilder.cpp
	elasticBeamColumn	ElasticBeam2d_bh.cpp* ElasticBeam2d_bh.h* TclElasticBeamCommand.cpp

Table A.1c New and modified classes for extending RBDO (continued) (Classes that do not exist in the "official" version are marked with *)

Project	Location of file	Files
Material	uniaxial	Steel01_epsy.cpp*
		Steel01_epsy.h*
		SmoothSteel01_epsy.cpp*
		SmoothSteel01_epsy.h*
		ElasticPPMaterial_Fy.cpp*
		ElasticPPMaterial_Fy.h*
		SmoothElasticPPMaterial_Fy.cpp*
		SmoothElasticPPMaterial_Fy.h*
		TclModelBuilderUniaxialMaterialCommand.cpp
	section	FiberSection2d.cpp
		RCFiberSection2d.cpp*
		RCFiberSection2d.h*
		TclModelBuilderSectionCommand.cpp

Appendix B: User's Guide to Optimization Analysis

This appendix contains the user guide to new implementations of the reliability-based design optimization (RBDO). It is a complement to the user guide to reliability and sensitivity analyses in Haukaas and Der Kiureghian (2004). The optimization commands used in this section have the same format as in Haukaas and Der Kiureghian (2004). An example of a command is:

commandName arg1? arg2? arg3? <arg4? ...>

A question mark after an argument indicates that an integer or a floating-point number should be provided; otherwise, a character string is given. Optional arguments are enclosed in angular brackets (Haukaas & Der Kiureghian, 2004). Note that all mentioned tasks (A1-A3, B1-B3, C1-C3, and D1-D2) in this appendix are described in detail in Chapter 6.

B1: RBDO Modeling

This section describes how to define design variables and functions involved in the RBDO analysis. The object mapping design variables into the finite element domain is also introduced.

A design variable object defines design variables by giving their start points through the following command:

designVariable tag? startPt?

The **tag** argument indicates the identification number of the design variable. These objects must be ordered in a consecutive and uninterrupted manner. The **startPt** argument allows the user to specify a value for the design variable to be used as the start point in the search for the design point (Haukaas & Der Kiureghian, 2004).

A **design variable positioner** object is used to map the design variables into structural properties in the finite element model through the following command:

designVariablePositioner tag? -dvNum dvNum? (...parameter identification...)

The **tag** argument indicates the identification number of the design variable positioner. The **dvNum** argument indicates the identification number of the pre-defined design variable. The parameter identification alternatives in the command are exactly the same as in the random variable positioner command in Haukaas and Der Kiureghian (2004).

A **constraint function** object defines constraint functions using user-defined expressions through the following command:

constraintFunction tag? "expression"

The **tag** argument indicates the identification number of the constraint function. The **expression** must be enclosed in double quotes and can be any analytical expression that can be evaluated by the Tcl interpreter (Welch, 2000). This function may be expressed by various quantities including random variables, design variables, structural response quantities from an OpenSees finite element analysis, and parameters defined in the Tcl interpreter (Haukaas & Der Kiureghian, 2004). The syntax used in this command is the same as that in the performance function command in Haukaas and Der Kiureghian (2004). An example of syntax for design variables is **{d_1}**, which means the first design variable.

A **cost function** object defines cost functions using user-defined expressions through the following command:

costFunction tag? "expression"

The **tag** argument indicates the identification number of the cost function. The **expression** has the same properties as that in the constraint function command. However, only design variables and parameters defined in the Tcl interpreter are employed as quantities in this expression.

An **objective function** object defines objective functions through combing the previously defined cost functions. Currently, the **standard** type of objective function is available for this object in OpenSees. The corresponding command reads:

objectiveFunction tag? -type standard -costFunctions costNum1? costNum2?

The **tag** argument indicates the identification number of the objective function. A standard objective function is created in the following way: objective function = 1^{st} cost function + failure probability × 2^{nd} cost function, where the failure probability is passed from the *ReliabiltyDomain* each time the objective function object is called.

B2: Analysis Tools

Before a RBDO analysis is executed the user must create an aggregation of necessary analysis components or tools. Which analysis components are needed depends on the analysis type. The order in which the tools are provided is of importance, since some tools make use of other tools. The user will be notified by an error message if dependencies are violated (Haukaas & Der Kiureghian, 2004).

A **nonlinSingleIneqOpt** object is created to be responsible for solving nonlinear single inequality constrained optimization problems. This object promises to solve the tasks B1 and D2. The corresponding command reads:

nonlinSingleIneqOpt PolakHe -alpha arg1? -beta arg2? -gamma arg3? -delta arg4?

This type of optimization problem is solved by the Polak-He algorithm. In the Polak-He algorithm, **arg1** denotes the parameter alpha (default = 0.5), **arg2** denotes the parameter beta (default = 0.8), **arg3** denotes the parameter gamma (default = 2.0), and **arg4** denotes the parameter delta (default = 1.0).

A **linMultiIneqOpt** object is responsible for solving linear multi-inequality constrained optimization problems. The corresponding command reads:

linMultiIneqOpt LSSOL

In order to find the search direction for tasks B3 and D1, a quadratic sub-optimization problem with linear constraints must be solved. This sub-optimization problem is fulfilled by the linMultiIneqOpt object. Currently, a Fortran 77 program **LSSOL** is called to solve this problem and is the available implementation of this object in OpenSees.

A **nonlinMultiIneqOpt** object is created to be responsible for solving nonlinear multi-inequality constrained optimization problems. This object promises to solve tasks B3 and D1. The corresponding command reads:

nonlinMultiIneqOpt PolakHe -alpha arg1? -beta arg2? -gamma arg3? -delta arg4?

This type of optimization problem is solved using the Polak-He algorithm. A quadratic sub-optimization problem with linear constraints must be solved in this object to find the search direction. Therefore, a linMultiIneqOpt object must be created before the nonlinMultiIneqOpt object can be instantiated. **arg1** to **arg4** are user-defined parameters used in the Polak-He algorithm and have the same definition as the parameters in the **nonlinSingleIneqOpt** object.

B3: Analysis Execution and Results

Two analysis types are available in the optimization module of OpenSees. This section describes the corresponding commands to execute them. Required analysis tools must be specified prior to using any of these commands. During the course of a RBDO analysis, status information may be printed to a file or to a computer monitor. The complete results from a successful analysis are printed to an output file whose name is specified by the user, as show below (Haukaas & Der Kiureghian, 2004).

A **DSA-MOOA analysis** object is the top-level of the DSA-MOOA approach and is responsible for obtaining the optimal design by orchestrating tasks A1 to A3. This object is executed using the following command:

runDSA_MOOAOptimizationAnalysis outputfilename -beta0 arg1? -targetCost arg2? -maxIterOuter arg3? -maxIterInner arg4? -numSimulation arg5? -targetCOV arg6?

The order of arguments is arbitrary. **arg1** denotes the lower bound of failure probability (default = 3.0), **arg2** denotes the target total expect failure cost, **arg3** denotes the maximum number of iterations on top level (A1-A3), and **arg4** denotes the maximum number of iteration in task B3. **arg5** and **arg6** are input parameters necessary for importance sampling in task A2. **arg5** denotes the maximum number of simulations (default = 10^6), while **arg6** denotes the target coefficient of variation (default = 2%). The nonlinSingleIneqOpt, linMultiIneqOpt, and nonlinMultiIneqOpt objects must be created before the DSA_MOOAAnalysis object is created. The results in the output file are self-explanatory, including the optimal design as well as reliability index and total expected failure cost.

A **DSA-S analysis** object is the top level of the DSA-S approach and is responsible for obtaining the optimal design by orchestrating tasks C1 to C3. This object is executed using the following command:

runDSA_SOptimizationAnalysis outputfilename -beta0 arg1? -targetCost arg2? -maxIterOuter arg3? -maxIterInner arg4? -numSimulation arg5? -targetCOV arg6?

The input data and the results of this analysis type are exactly same as those in the DSA-MOOA analysis, expect for **arg3** and **arg4**. **arg3** denotes the maximum number of iterations of top level (C1-C3), while **arg4** denotes the maximum number of iterations in task D1.