NAVAL
POSTGRADUATE
SCHOOL

MONTEREY, CALIFORNIA

THESIS

RELIABILITY-BASED DESIGN OPTIMIZATION USING BUFFERED FAILURE PROBABILITY

by

Habib Gurkan Basova

June 2010

Thesis Advisor:	Johannes O. Royset
Second Reader:	R. Kevin Wood

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2010	**3. REPORT TYPE AND DATES COVERED** Master's Thesis
4. TITLE AND SUBTITLE Reliability-Based Design Optimization Using Buffered Failure Probability		**5. FUNDING NUMBERS**
6. AUTHOR(S) Habib Gurkan Basova		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number ________________.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT

Reliability-based design optimization (RBDO) seeks the best design for a structural system under uncertainty. Typically, uncertainty arises from random loads such as wind pressure and random material properties such as yield stress. A reliable design must account for uncertainties to ensure safety.

Various methods have been proposed to solve the nonlinear optimization models that RBDO uses. However, these methods are theoretically and computationally troublesome as they involve constraints on failure probability, and failure probability is difficult to handle in optimization algorithms. This thesis considers an alternative approach to RBDO that uses the "buffered failure probability," and develops four new solution algorithms based on sample-average approximations. Buffered failure probability is more conservative than failure probability and it is much easier to handle in optimization algorithms.

We test the algorithms on six engineering-design examples from the literature. The examples range from simple systems with two design variables to complicated ones with ten. Results show that the new algorithms may reduce solution time by an average factor of 560 compared to an existing algorithm. Furthermore, they can handle problem instances with two orders of magnitude larger sample sizes, which may be important for reasons of accuracy.

14. SUBJECT TERMS Reliability-based Design Optimization, Failure Probability, Buffered Failure Probability, Structural Reliability.			**15. NUMBER OF PAGES** 73
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

RELIABILITY-BASED DESIGN OPTIMIZATION USING BUFFERED FAILURE PROBABILITY

Habib Gurkan Basova
Captain, Turkish Army
B.S., Turkish Military Academy, 2000

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
June 2010

Author:	Habib Gurkan Basova
Approved by:	Johannes O. Royset Thesis Advisor
	R. Kevin Wood Second Reader
	Robert F. Dell Chairman, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Reliability-based design optimization (RBDO) seeks the best design for a structural system under uncertainty. Typically, uncertainty arises from random loads such as wind pressure and random material properties such as yield stress. A reliable design must account for uncertainties to ensure safety.

Various methods have been proposed to solve the nonlinear optimization models that RBDO uses. However, these methods are theoretically and computationally troublesome as they involve constraints on failure probability, and failure probability is difficult to handle in optimization algorithms. This thesis considers an alternative approach to RBDO that uses the "buffered failure probability," and develops four new solution algorithms based on sample-average approximations. Buffered failure probability is more conservative than failure probability and it is much easier to handle in optimization algorithms.

We test the algorithms on six engineering-design examples from the literature. The examples range from simple systems with two design variables to complicated ones with ten. Results show that the new algorithms may reduce solution time by an average factor of 560 compared to an existing algorithm. Furthermore, they can handle problem instances with two orders of magnitude larger sample sizes, which may be important for reasons of accuracy.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

LIST OF ACRONYMS AND ABBREVIATIONS

α	Reliability Level
MCS	Monte Carlo Simulation
$p(x)$	Failure Probability
$\bar{p}(x)$	Buffered Failure Probability
PMA	Performance Measure Approach
RBDO	Reliability-Based Design Optimization
RIA	Reliability Index Approach
$q_\alpha(x)$	α- quantile
$\bar{q}_\alpha(x)$	α- superquantile

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

Reliability-based design optimization (RBDO) deals with how to best design a structural system under uncertainty while considering reliability constraints. Uncertainty present in engineering systems typically arises from random loads such as wind pressure and random material properties such as buckling stress. A reliable design must account for these uncertainties in order to ensure safety.

Various methods have been proposed to solve the nonlinear optimization models that RBDO uses. However, these methods are theoretically and computationally troublesome as they involve constraints on failure probability, and failure probability is difficult to handle in nonlinear optimization algorithms. The constraints based on failure probability may yield difficult-to-solve optimization problems.

This thesis considers an alternative approach to RBDO that uses "buffered failure probability," and considers five solution algorithms based on sample-average approximations. The buffered failure probability approach is more conservative than the traditional approach based on the failure probability, meaning that a design that satisfies a reliability constraint based on buffered failure probability is guaranteed to satisfy one based on failure probability. The buffered failure probability is also much easier to handle in nonlinear optimization algorithms.

The first three algorithms each solve models that incorporate a single sample-average approximation of the buffered failure probability constraint. Algorithm 1 is a well-known method based on a model reformulation and solves a single, large nonlinear program. Algorithms 2–5 are new algorithms developed in this thesis. Algorithm 2 is an active-set implementation of Algorithm 1. Algorithm 3 uses exponential smoothing of a max-function to avoid the large-scale reformulation of Algorithm 1. Algorithm 4 approximately solves a sequence of sample-average approximations within an adaptive sample-adjustment scheme that ensures the sample size is gradually increased to infinity. Algorithm 5 is similar to Algorithm 4, but includes an active-set strategy.

We test the algorithms on six engineering-design examples from the literature. The examples range from simple systems with two design variables to complicated ones with ten design variables. Results from Algorithm 2 show an average speed-up in solution time by a factor of 560 over Algorithm 1. Algorithm 3 exhibits a speed-up by a factor of 31 over Algorithm 1. Algorithms 2 and 3 can handle problem instances with large sample sizes, in fact, one and two orders of magnitude larger than that of Algorithm 1, respectively. The ability to handle large sample sizes is important, for reasons of accuracy. Algorithms 4 and 5 obtain high-quality solutions in minutes without the need for a user to specify a sample size, which may be difficult in practice.

The results in this thesis show that Algorithms 2–5 have significantly improved engineers' ability to relatively quickly generate cost efficient designs that satisfy a failure probability constraint.

ACKNOWLEDGMENTS

I would like to extend my personal thanks to Professor Johannes O. Royset, for his clear guidance and constant availability. With his unwavering patience this difficult process has become an exceptional learning opportunity. I could not have completed this research without his tireless efforts.

I also wish to thank Professor R. Kevin Wood for his dedicated efforts to ensure this study to be genuine research writing. His editing contribution and professional insight ensured the highest standard in this thesis.

I want to thank my wife, Medalet, for her unconditional love, precious support and endless patience in the midst of this challenge and my daughter, Elif, for enduring days without dad so I could finish this work.

Finally, I would like to offer my special thanks to my country, Turkey, for providing me the unique opportunity to attend the Operations Research program at NPS.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. PROBLEM STATEMENT

This thesis considers the reliability-based design-optimization problem (RBDO) of minimizing the design cost of a structural system subject to a constraint on its reliability and other quantities. We characterize a system’s reliability in terms of the probability of failure with respect to one or more performance requirements.

We approximate this problem by replacing the failure probability constraint with a buffered failure probability constraint and develop four algorithms for solving the resulting nonlinear program.

B. MOTIVATION

Engineers aim to minimize design costs of structural systems such as aircraft wings, vehicle frames, ship hulls, bridges, and buildings. The minimization typically is subject to one or more constraints on system reliability. The parameters that describe the shape and characteristics of the system are referred to as design variables. These variables are controlled by the engineer and are manipulated so that the cost of the system is minimized and the reliability is sufficiently large. Therefore, the challenge for an engineer is to reduce the design cost while satisfying requirements for system performance and reliability.

A system’s reliability accounts for uncertain loads that act on the system, and the uncertain capacity of the system to withstand these loads. There are a variety of uncertainties that engineers need to consider. A system’s performance depends on the type and magnitude of loads and the strength of the system, which is related to properties of materials used in the design. The loads and system strength can be described by random (uncontrollable) variables. As an example, for a building, wind pressure is an uncontrollable load that can be modeled using random variables. The loads may lead the system to not being able to meet functional and safety requirements. The result can be loss of serviceability, or a complete destruction of the system. The functional requirements are called limit states of the system and, when they are exceeded, we say that the system is failed.

Various methods have been proposed for solving the nonlinear programs arising in RBDO. However, these methods are theoretically and computationally troublesome as they involve the failure probability, which may be nonsmooth and nonconvex and therefore difficult to handle by standard nonlinear programming solvers such as SNOPT (Gill, Murray, & Saunders, 1998), LANCELOT (Conn, Gould, & Toint, 1992), and NLPQL (Schittkowski, 1985). This thesis explores another approach to RBDO based on buffered failure probability. The buffered failure probability approach is more conservative than the traditional approach based on the failure probability (Rockafellar & Royset, 2010). Thus, a design that satisfies the reliability constraint based on the buffered failure probability also satisfies one based on the failure probability. The buffered failure probability is computationally easier to handle due to the algorithmic advances of this thesis. Rockafellar and Royset (2010) also discuss other potential advantages that buffered failure probability has over failure probability in this context. However, this thesis shows that the computational advantages alone are sufficient to warrant a preference for the buffered failure probability approach.

C. SCOPE AND LIMITATIONS

We represent uncertainties as random variables. In principle, the random variables can be either discrete or continuous. However, we only consider continuous random variables in this thesis.

We assume that we know the joint distribution of the random variables and that the corresponding cumulative distribution function is strictly increasing.

A real-life structural system generally is composed of several components. If failure of any one of these components constitutes failure for the entire system, then we call the structure a series system. In contrast, parallel systems are those that need the failure of all the components for a system failure. In this thesis, we focus on series systems with, consequently, one failure probability constraint.

D. LITERATURE REVIEW

The classical methods for solving the RBDO perform a double-loop approach: an outer optimization loop and an inner reliability assessment loop; see Du and Chen (2004). The reliability assessment is performed by two different methods: Reliability Index Approach (RIA) (Lee, Yang, & Ruy, 2002) or Performance Measure Approach (PMA) (Tu, Choi, & Park, 1999). Both of these methods approximate the failure probability by surrogates of unknown accuracy and hence cannot guarantee the convergence to globally, locally, or stationary solutions of RBDO problems. A single-loop RBDO approach is proposed by Liang, Mourelatos, and Tu (2008). This method utilizes the fact that a surrogate for the failure probability can be evaluated by solving a nonlinear program. Hence, the RBDO problem with a failure probability constraint can be approximated by an optimization model with equilibrium constraint. However, the resulting model may be difficult to solve due to nonconvexity, nonsmoothness, and/or lack of a constraint qualification. Moreover, as the accuracy of the surrogate for the failure probability is unknown, the computed design may be nonoptimal and violate failure probability constraints. We refer to Rockafellar and Royset (2010) and Royset, Der Kiureghian, and Polak (2006) for a more comprehensive literature review and difficulties associated with current approaches.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND AND PROBLEM DEFINITION

A. LIMIT-STATE FUNCTIONS AND FAILURE PROBABILITY

A system response relative to a functional requirement, viewed as a function of the design and random variables, is referred to as a limit-state function and denoted by $g(x,V)$. Here $x = (x_1, x_2, ..., x_n)'$ is a vector of design variables (where prime $'$ denotes the transpose of a vector) and $V = (V_1, V_2, ..., V_m)'$ is a vector of random variables. The joint probability distribution of the random variables is known. We denote realizations of V by v. For a given x, $g(x,v)$ represents the performance of the system under realization v of V. An unsatisfactory performance, i.e., "failure," occurs when $g(x,v) > 0$. If $g(x,v) \le 0$ then the system performance is acceptable.

In complex systems, there may be multiple limit-state functions; for example, see Example 5 in the Appendix, taken from Rao (2009, pp. 472–473). We denote these functions as $g_k(x,v)$, $k \in K$, where $K = \{1, 2, ..., m\}$. We also define

$$g(x,v) = \max_{k \in K} \left\{ g_k(x,v) \right\}. \tag{1}$$

We can characterize the reliability of the system by its failure probability defined by

$$p(x) = Prob\left[g(x,V) > 0 \right]. \tag{2}$$

The following simplified example illustrates a design process based on the failure probability. (Note: For clarity, this example uses discrete x, but we note that this thesis develops solution methods only for continuous x.)

Example 1. A TOW missile is a heavy anti-tank missile. One component of the missile's launcher is an optical system. Suppose that two different optical systems, 1 and 2, are available for incorporation in a final design: setting $x_i = 1$ means system i is selected and $x_i = 0$, otherwise. The decision maker plans to procure and incorporate the system with lower failure probability. Systems 1 and 2 have an operational capability down to -28 °F and -30 °F, respectively. The procurement cost of system 2 is higher than system 1. The missile will be operated under severe conditions in a certain mission area. Let V represent the random temperature in Fahrenheit degrees, which is known to be normally distributed with mean -20 °F and standard deviation 3 °F in that area. Engineers have characterized the reliability of the system by the following limit-state function

$$g(x,v) = -0.9v - 28x_1 - 30x_2, \tag{3}$$

where $x_1 + x_2 = 1$ and $x_1, x_2 \in \{0,1\}$. The coefficient 0.9 is a scaling factor and represents the impact of the current temperature on the system. If $g(x,v)$ is positive for a given temperature v, the system fails. We compute that designs 1 and 2 have failure probabilities of 1.06×10^{-4} and 4.406×10^{-6}, respectively. Thus, system 2 will be procured.

For a given design x, the computation of failure probability requires the evaluation of a high-dimensional integral. That is,

$$p(x) = \int\ldots\int I\left(g(x,v)\right) f_V(V)dv_1\ldots dv_m, \tag{4}$$

where $f_V(V)$ is the joint probability density function for the random vector V, and $I(z)=1$ if $z>0$, and $I(z)=0$ otherwise.

B. DESIGN OPTIMIZATION OF A SYSTEM SUBJECT TO A FAILURE PROBABILITY CONSTRAINT

Engineers often seek to solve the design-optimization problem (Rockafellar & Royset, 2010)

$$\begin{aligned} \mathbf{P}: \quad & \min_{x} \ f(x) \\ & \text{s.t.} \quad p(x) \le 1-\alpha \\ & \qquad x \in X, \end{aligned} \tag{5}$$

where $f(x)$ is a deterministic and continuously differentiable cost function for the system, X is a continuous region of allowable designs, and α is a desired reliability level in (0,1]. We also assume that X is convex.

Problem **P** represents the current approach to RBDO. Typically, **P** is difficult to solve even approximately because the computation of the failure probability in (4) is computationally challenging. Furthermore, the integrand in (4) is non-smooth, i.e., is not differentiable at every point, and this may cause difficulties during optimization. Moreover, $p(x)$ is generally nonconvex. With lack of convexity, a solution algorithm may yield only a locally optimal solution.

Solving **P** is also difficult because computation of the failure probability requires the evaluation of the high-dimensional integral in (4). For a real-world system, the limit-state functions are highly complex and involve many different random variables. (See Example 6 in the Appendix, taken from Samson, Thoomu, Fadel, & Reneke 2009). Consequently, failure reliability can only be estimated: the standard approach uses Monte Carlo simulation (MCS) (Melchers, 1999; Choi, Grandhi, & Canfield, 2007).

MCS is a sampling method that estimates expectation and probability. When estimating the probability of failure, MCS computes the following estimate of the probability of failure in (4):

$$\hat{p}(x) = \frac{1}{N}\sum_{j=1}^{N} I(g(x, v^j)), \tag{6}$$

where N is the number of random, sampled, vector realizations v^j. The function $\hat{p}(x)$ is an unbiased estimator of $p(x)$ (Rubinstein & Kroese, 2008). The standard deviation of $\hat{p}(x)$ is inversely proportional to the square root of the sample size N. Hence, the accuracy of failure probability estimates is poor for small samples, especially for small failure probabilities. Therefore, replacing the failure probability by its estimator in (6) computed with a large N leads to long solution times for sampled approximations to **P**. Furthermore, (6) is not differentiable because of the indicator function, and this may cause convergence difficulties for standard nonlinear optimization algorithms. We refer to Rockafellar and Royset (2010) for more detailed explanations of difficulties associated with $p(x)$.

C. BUFFERED FAILURE PROBABILITY

As discussed above, the failure probability has several undesirable properties that may cause difficulties for a standard nonlinear optimization algorithm. We now introduce an alternative approach to the design-optimization problem using another measure of failure, the *buffered failure probability* (Rockafellar & Royset, 2010). The buffered failure probability approach is based on the conditional value-at-risk (Rockafellar & Uryasev, 2000; Rockafellar & Uryasev, 2002), which is used in financial engineering to compute optimal investment portfolios. We next define the buffered failure probability.

Suppose we wish to solve **P** with a failure probability level $1-\alpha$. Instead of imposing the constraint $p(x) \le 1-\alpha,$ we introduce $\bar{p}(x) \le 1-\alpha$ as an alternative, where $\bar{p}(x)$ denotes the buffered failure probability (Rockafellar & Royset, 2010). A design that satisfies the buffered failure probability constraint also satisfies the failure probability requirements, as we describe later, and the constraint $\bar{p}(x) \le 1-\alpha$ is easier to handle computationally. The following description follows Rockafellar and Royset (2010) closely.

Let the cumulative distribution function of $g(x,V)$ be denoted by

$$F_x(\gamma) = Prob\big(g(x,V) \le \gamma\big), \tag{7}$$

which we assume to be continuous and strictly increasing. We next define the α*- quantile function*, or simply α*- quantile,* which is denoted as $q_\alpha(x).$ For a probability level $\alpha,$

$$q_\alpha(x) = F_x^{-1}(\alpha). \tag{8}$$

In addition, we define the α*- superquantile function*, or simply α*- superquantile,* by

$$\bar{q}_\alpha(x) = E\big[g(x,V) \mid g(x,V) \ge q_\alpha(x)\big]. \tag{9}$$

Integration is also useful in finding the superquantile. That is,

$$\bar{q}_\alpha(x) = \frac{1}{1-\alpha}\int_\alpha^1 q_\beta(x)d\beta. \tag{10}$$

The superquantile can be interpreted as the value of $g(x,V)$ that splits the area under the probability density function in the interval $[q_\alpha(x),\infty)$ into two balancing parts. That is, it represents the average value of the limit-state function, conditioned on $g(x,V)$ being no less than the α- quantile. The superquantile is identical to the conditional value-at-risk, but we follow Rockafellar and Royset (2010) here and use the term superquantile. When the superquantile equals zero at a probability level α, then the buffered failure probability $\overline{p}(x)$ is equal to $1-\alpha$. Hence, the buffered failure probability may be defined as

$$\overline{p}(x) = Prob\left[g(x,V) \geq q_{\alpha_0}(x) \right], \tag{11}$$

where α_0 is such as $\overline{q}_{\alpha_0}(x)=0$ (Rockafellar & Royset, 2010).

Figure 1 illustrates the probability density function for an example limit-state function at a fixed $x=\hat{x}$, where α_0 has been identified such that $\overline{q}_{\alpha_0}(\hat{x})=0$. $\overline{q}_{\alpha_0}(\hat{x})$ splits the area under the probability density function to the right of -2 into two balancing parts. Hence, the quantile $q_{\alpha_0}(\hat{x})=-2$. We also see in Figure 2, which represents the cumulative distribution function for the same limit-state function, that α_0 therefore must equal 0.85. Thus, we calculate the buffered failure probability for this example as $1-0.85$, which is 0.15.

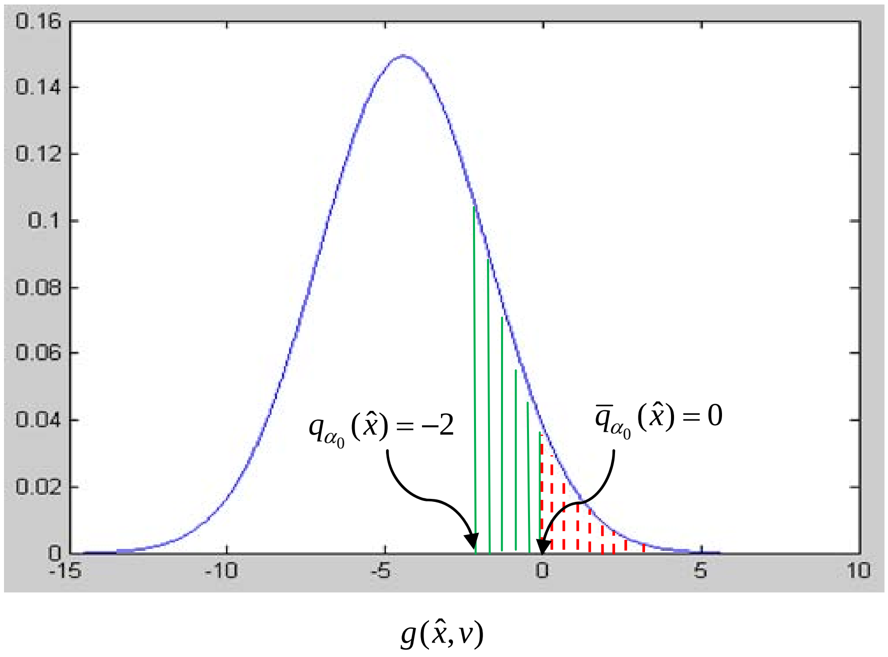

Figure 1. Example Probability Density Function (pdf) for $g(\hat{x},V)$ when That Function is Normally Distributed

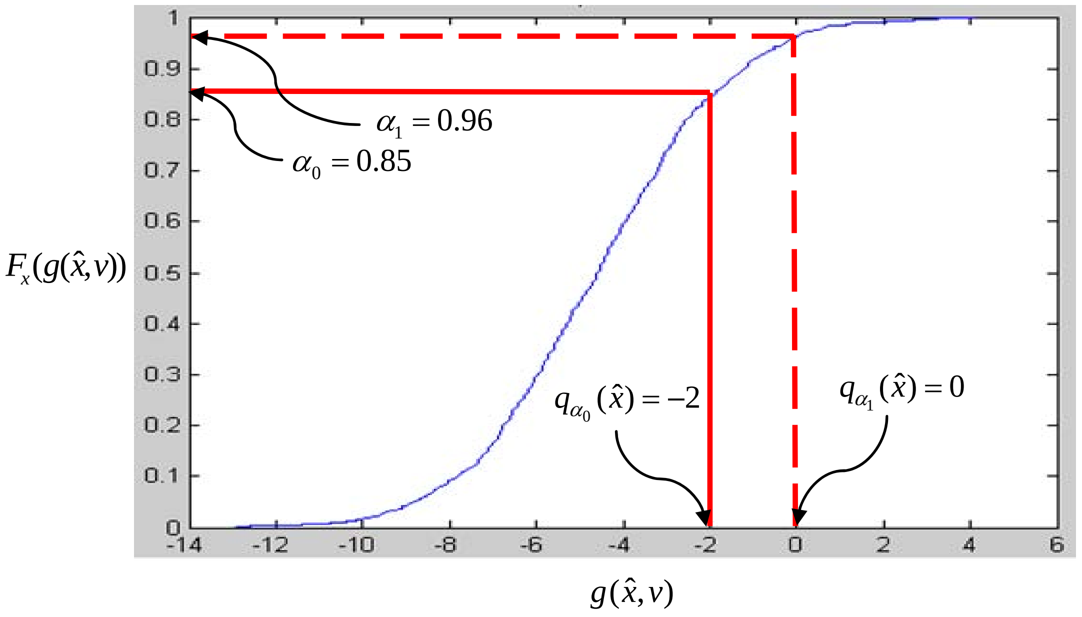

Figure 2. Example Cumulative Distribution Function (cdf) for $g(\hat{x},V)$ when That Function is Normally Distributed

In general, for any α value and any $x \in X$, $q_\alpha(x) \le \bar{q}_\alpha(x)$ and thus $p(x) \le \bar{p}(x)$. Hence, the buffered failure probability is a conservative estimate of the failure probability. The relationship $p(x) \le \bar{p}(x)$ is easily obtained from the following argument. Suppose $x = \hat{x}$ is fixed, and α_0 is chosen so that $\bar{q}_{\alpha_0}(\hat{x}) = 0$. Then, by the definition of the superquantile, it follows that $q_{\alpha_0}(\hat{x}) \le 0$. Since $\bar{q}_{\alpha_0}(\hat{x}) = 0$, the buffered failure probability $\bar{p}(\hat{x})$ equals to $1-\alpha_0$. It follows from the definition of the failure probability that if $q_{\alpha_1}(\hat{x}) = 0$ for some α_1, then $p(\hat{x}) = 1-\alpha_1$. Since $q_{\alpha_0}(\hat{x}) \le q_{\alpha_1}(\hat{x})$, it must be true that $\alpha_0 \le \alpha_1$. Thus, $1-\alpha_0$ is no smaller than $1-\alpha_1$. Since the arguments above hold for any $\hat{x} \in X$ we have the result that $p(x) \le \bar{p}(x)$ for all $x \in X$. Figure 2 illustrates this situation with $\alpha_1 = 0.96$, $\alpha_0 = 0.85$, $q_{\alpha_1}(\hat{x}) = 0$, and $q_{\alpha_0}(\hat{x}) = -2$.

The computation of the buffered failure probability appears cumbersome. As we see below, however, we never directly use this definition in computations and instead use alternative expressions easily incorporated in optimization models.

The buffered failure probability constraint $\bar{p}(x) \leq 1-\alpha$ is satisfied if

$$\bar{q}_\alpha(x) \leq 0. \tag{12}$$

Rockafellar and Uryasev (2002) show that for any $x \in X$

$$\bar{q}_\alpha(x) = \min_{z_0} \phi_\alpha(z_0, x), \tag{13}$$

where z_0 is an auxiliary design variable and

$$\phi_\alpha(z_0, x) = z_0 + \frac{1}{1-\alpha} E\left[\max\left\{0, g(x,V) - z_0\right\}\right]. \tag{14}$$

Therefore, a design $x \in X$ and a value for z_0 that satisfy

$$\phi_\alpha(z_0, x) \leq 0 \tag{15}$$

also satisfy $\bar{p}(x) \leq 1-\alpha.$

Although $\bar{p}(x)$ generally overestimates $p(x),$ the numerical examples that we discuss in Chapter IV show that the difference between the two probabilities need not be great.

We can formulate a restriction of **P,** by replacing the reliability constraint in **P** with (15). Thus, a restriction to **P** using buffered failure probability takes the form

$$\begin{aligned} \mathbf{BP}: \quad & \min_{x,\, z_0} \ f(x) \\ \text{s.t.} \quad & z_0 + \frac{1}{1-\alpha} E\left[\max\left\{0, g(x,V) - z_0\right\}\right] \leq 0 \\ & x \in X. \end{aligned} \tag{16}$$

With **BP** being a restriction of **P**, the solution of **BP** is also feasible in **P**, but may not be optimal in **P**. However, the constraint (16) is easier to handle than (5) as it is convex when each $g_k(x,v),\ k \in K$ is convex in x for all v (Rockafellar & Royset, 2010). Moreover, in contrast to the nonsmoothness of the indicator function in (4), the nonsmoothnesss in (16) is easily handled (as described in Chapter III). Thus, computation of optimal values is easier in **BP** than in **P**. We also note that the buffered failure probability accounts for the tail of the distribution of $g(x,V)$ in a different way than does failure probability. We refer to Rockafellar and Royset (2010) for a discussion of why this might be an advantage for RBDO.

We next illustrate the failure probability and buffered failure probability approaches on a stochastic, continuous knapsack problem with random knapsack capacity. Let c and w be deterministic vectors of per-unit item values and item weights, respectively, let α be a user-defined reliability level, let V be a random variable describing the total capacity of the knapsack in terms of weight, and let x be a vector of decision variables that chooses the amount of each item to placed in the knapsack. We assume that V is normally distributed with mean μ and standard deviation σ. The knapsack problem then takes the form:

$$\begin{aligned}
\mathbf{KP}: \quad & \max_{x} \ c'x \\
\text{s.t.} \quad & Prob(w'x \le V) \ge \alpha \qquad (17) \\
& x \ge 0.
\end{aligned}$$

In our notation, this problem instance has only one limit-state function:

$$g(x,V) = w'x - V. \qquad (18)$$

Since limit-state function values greater than zero represent failure, the following equation becomes the reliability constraint:

$$Prob(w'x - V > 0) \le 1 - \alpha, \qquad (19)$$

where $w'x - V$ is normally distributed with mean $w'x - \mu$ and standard deviation σ. The reliability constraint in (19) takes the following form:

$$1 - \Phi\left(\frac{0 - (w'x - V)}{\sigma}\right) \le 1 - \alpha, \qquad (20)$$

where $\Phi(\cdot)$ represents the cumulative distribution function of the standard normal distribution. Thus, the problem $\mathbf{KP}$ is equivalent to

$$\begin{aligned}
\mathbf{KP}': \quad & \min_{x} \ -c'x \\
\text{s.t.} \quad & w'x - \mu + \sigma \ \Phi^{-1}(\alpha) \le 0 \qquad (21) \\
& x \ge 0.
\end{aligned}$$

Next, we consider the buffered failure probability approach. In this case **BP** takes the form

$$\textbf{KBP}: \quad \min_{x} \; -c'x$$
$$\text{s.t.} \quad \bar{q}_\alpha(x) \leq 0 \tag{22}$$
$$x \geq 0.$$

Since $g(x,V)$ is normally distributed with mean $w'x-\mu$ and standard deviation σ, the superquantile $\bar{q}_\alpha(x)$ is easily computed (Truncated Normal Distribution, Wikipedia, n.d.) and we obtain

$$\bar{q}_\alpha(x) = \mu + \frac{\sigma\ \phi\left(q_\alpha(x)\right)}{1-\alpha}, \tag{23}$$

where $\phi(\cdot)$ is the standard normal probability density function. Therefore, **KBP** takes the following equivalent form:

$$\textbf{KBP}': \quad \min_{x} \; -c'x$$
$$\text{s.t.} \quad w'x - \mu + \frac{\sigma\ \phi\left(\Phi^{-1}(\alpha)\right)}{1-\alpha} \leq 0 \tag{24}$$
$$x \geq 0.$$

Thus, for this knapsack problem, the difference between the reliability constraints of the failure and buffered failure probabilities corresponds to the difference between (21) and (24). Table 1 gives the near-optimal solutions of **KP**$'$ and **KBP**$'$ when V is normally distributed with mean 3.5 and standard deviation 0.1, $c' = (2,1)$, $w' = (1.1, 2.1)$, and $\alpha = 0.99$. From Table 1 we see that the buffered failure probability approach results in a conservative design, with a failure probability that is about 40% lower than required.

Problem	x_1	x_2	$-c'x$	$p(x)$
KP$'$	1.06124	1.00000	-3.12248	0.0100
KBP$'$	1.03043	1.00000	-3.06087	0.0039

Table 1. Comparison of Solutions Using Failure and Buffered Failure Probabilities for a Knapsack Problem with $V \sim N(3.5, 0.1^2)$, $c' = (2,1)$, $w' = (1.1, 2.1)$, and $\alpha = 0.99$

THIS PAGE INTENTIONALLY LEFT BLANK

III. ALGORITHMS FOR RBDO BASED ON BUFFERED FAILURE PROBABILITY

This chapter examines five algorithms that exploit the properties of buffered failure probability to compute an approximated solution of **BP** in a relatively short time and with high accuracy. Since **BP** is a restriction of **P**, each solution also represents an approximate solution of **P**. Algorithm 1 is a well-known method based on the solution of a reformulation of **BP**. Algorithms 2–5 are new algorithms developed in this thesis. Algorithm 2 is an active-set strategy implementation of Algorithm 1. Algorithm 3 implements exponential smoothing of the max-function in (16) and proceeds to solve a smoothed problem. Algorithms 1–3 approximate the expectation $E[\max\{0, g(x,v) - z_0\}]$ in (16) by its sample average for a fixed sample size N, i.e.,

$$\frac{1}{N}\sum_{j=1}^{N} \max\left\{0, g(x, v^j) - z_0\right\} \tag{25}$$

with random vector realizations v^j, $j = 1, 2, ..., N$. Thus, the reliability constraint in (16) takes the following form:

$$z_0 + \frac{1}{N(1-\alpha)}\sum_{j=1}^{N} \max\left\{0, g(x, v^j) - z_0\right\} \leq 0. \tag{26}$$

Algorithm 4 implements an adaptive sample-adjustment scheme that ensures the sample size is gradually increased to infinity. Iterates generated by Algorithm 4 are guaranteed to converge to a stationary point of **BP** (Royset, 2010b). Algorithm 5 is similar to Algorithm 4 but includes an active-set strategy. We next discuss each algorithm in turn.

A. ALGORITHMS WITH FIXED SAMPLE SIZE

1. Algorithm 1: Reformulation of BP Using Sampling and Auxiliary Variables

The constraint (26) is non-smooth and is therefore not tractable by standard nonlinear optimization algorithms. Algorithm 1, an existing algorithm, uses a reformulation to overcome this difficulty. This reformulation was first proposed in Rockafellar and Royset (2010).

We introduce auxiliary variables z_j, $j = 1,...,N$, and find that for any j,

$$\max\left\{0, g(x,v^j) - z_0\right\} \le z_j \tag{27}$$

is equivalent to

$$0 \le z_j \tag{28}$$

and

$$g(x,v^j) - z_0 \le z_j. \tag{29}$$

Hence, we can use (28) and (29) to reformulate the "max" part of the constraint in (26). This results in the following transcription:

$$\begin{aligned}
\mathbf{BP}_N: \quad & \min_{x,\bar{z}} f(x) \\
\text{s.t. } & z_0 + \frac{1}{N(1-\alpha)} \sum_{j=1}^{N} z_j \le 0 && (30) \\
& g_k(x,v^j) - z_0 \le z_j, \quad \forall j \in J \quad \forall k \in K && (31) \\
& z_j \ge 0, \quad \forall j \in J && (32) \\
& x \in X,
\end{aligned}$$

where $\bar{z} = (z_0, z_1, ..., z_N)'$ and $J = \{1, 2, ..., N\}$. For large N, **BP**$_N$ is approximately equivalent to **BP** (Rockafellar & Royset, 2010). However, **BP**$_N$ necessitates introducing a new auxiliary design variable, i.e., z_j, for every realization of the random vector: This typically results in large-scale problems. Given **BP**$_N$, our first algorithm takes the following simple form:

Algorithm 1: Apply a standard nonlinear solver to **BP**$_N$**.**

We use the SNOPT (Gill et al. 1998) solver here and below in the following algorithms in computational experiments.

2. Algorithm 2: External Active-Set Strategy

BP$_N$ becomes a large-scale problem as N increases. Algorithm 2 aims to overcome this difficulty by using an active-set strategy proposed in Chung, Polak, and Sastry (2010).

In this approach, we do not let the standard nonlinear solver consider constraints and variables assumed to be "unimportant" at an optimal solution. The following is an adaptation of the algorithm proposed in Chung et al. (2010) to **BP**$_N$. Let $y = (x, \overline{z})$ and $\varepsilon > 0$. Then, we let

$$\omega_k^j(y) = g_k(x, v^j) - z_0 - z_j, \qquad \forall j \in J \qquad \forall k \in K \tag{33}$$

$$\chi(y) = \max_{j \in J, k \in K} \omega_k^j(y) \tag{34}$$

$$\chi(y)_+ = \max\{0, \chi(y)\} \tag{35}$$

$$w_\varepsilon(y) = \{(j,k) \in J \times K \mid \omega_k^j(y) \geq \chi(y)_+ - \varepsilon\}, \tag{36}$$

where $w_\varepsilon(y)$ represents "active" constraints at y. In each iteration of this algorithm we limit the number of iterations of the solver. Let $A_W^n(y)$ denote the solution from n iterations of the solver, starting from y, applied to the following problem:

$$\mathbf{BP}_N(W): \qquad \min_{x, \overline{z}} f(x)$$

$$\text{s.t. } z_0 + \frac{1}{N(1-\alpha)} \sum_{j \in W} z_j \leq 0 \tag{37}$$

$$g_k(x, v^j) - z_0 \leq z_j, \quad (j,k) \in W \tag{38}$$

$$z_j \geq 0, \qquad \forall j \tag{39}$$

$$x \in X.$$

We note that for $j \in J$ such that $(j,k) \notin W$ for any $k \in K$, z_j in $\mathbf{BP}_N(W)$ can be set to zero.

Algorithm 2:

Data: y_0 initial guess for variable values, $\varepsilon > 0$, $\upsilon > 0$, and $n > 0$ an integer.

```
Step 0: Set $i = 0$ and initial working set $W_i = w_\varepsilon(y_i)$.

Step 1: Compute $y_{i+1} = A^n_{W_i}(y_i)$.

Step 2: Compute $\chi(y_{i+1})$.

    If $\chi(y_{i+1}) \le 0$ and $|\chi(y_{i+1}) - \chi(y_i)| \le \upsilon$

        STOP,

    else compute $w_\varepsilon(y_{i+1})$,

        $W_{i+1} = W_i \cup \{w_\varepsilon(y_{i+1})\}$,

        Replace $i$ by $i+1$ and go to Step 1.
```

In Algorithm 2, we hope to reduce the solution time by considering only the active constraints in the optimization.

3. Algorithm 3: Exponential Smoothing of Max-Function

We next introduce exponential smoothing for **BP**$_N$. Let $L = \{0,1,2,...,m\}$ and $g_0(x,v) = 0$ for all x and v. Then, the following equations represent the exponential smoothing of the max-function in (26) (Polak, Womersley, & Yin, 2008):

$$\eta_k^j(x,z_0) = g_k(x,v^j) - z_0, \quad \forall j \in J \quad \forall k \in L \tag{40}$$

$$\zeta^j(x,z_0) = \max_{k \in L} \eta_k^j(x,z_0), \quad \forall j \in J \tag{41}$$

and

$$\delta_p^j(x,z_0) = \zeta^j(x,z_0) + \frac{1}{p}\log\left(\sum_{k=0}^{m} e^{p\left[\eta_k^j(x,z_0) - \zeta^j(x,z_0)\right]}\right), \ \forall j \in J \tag{42}$$

where $p > 0$ is a smoothing parameter. Thus, replacing the reliability constraint in (26) by (42) we obtain the following approximation to **BP**$_N$:

$$\mathbf{BP}_{Np}: \qquad \min_{x,\, z_0} f(x)$$
$$z_0 + \frac{1}{N(1-\alpha)} \sum_{j=1}^{N} \delta_p^j(x, z_0) \le 0 \tag{43}$$
$$x \in X.$$

If all the limit-state functions are continuously differentiable, which we assume they are in this thesis, then $\delta_p^j(x, z_0),\ j \in J$, are also continuously differentiable.

We improve the quality of the smoothing approximation as we increase the value of p. That is, the error in the constraint due to smoothing vanishes, as $p \to \infty$ (Polak, Womersley, & Yin, 2008). Our third algorithm then takes the following form:

Algorithm 3: Apply a standard nonlinear solver to **BP**$_{Np}$.

B. ALGORITHMS WITH INCREASING SAMPLE SIZE

We next introduce another approach, which adaptively increases the sample size during the optimization process. This approach intends to avoid using unnecessarily large sample sizes. Suppose that

$$X = \left\{ x \mid f^j(x) \le 0,\ j = 1, 2, ..., q \right\}, \tag{44}$$

where $f^j(x)$ represents nonlinear and non-negativity constraints and

$$\varsigma_{Np}(x, z_0) = z_0 + \frac{1}{N(1-\alpha)} \sum_{j=1}^{N} \delta_p^j(x, z_0) \tag{45}$$

represents the reliability constraint defined in (43). Moreover, let

$$\psi_{Np}(x, z_0) = \max\left\{ \varsigma_{Np}(x, z_0),\ \max_{j\ =\ 1,\ ...,\ q} f^j(x) \right\}, \tag{46}$$

$$\psi_{Np}(x, z_0)_+ = \max\left\{ 0, \psi_{Np}(x, z_0) \right\}, \tag{47}$$

and

$$\theta_{Np}(x, z_0) = \min_{h,\xi} \Big\{ \max\big\{ -\psi_{Np}(x, z_0)_+ + \nabla f(x)'h,$$
$$\max\big[\varsigma_{Np}(x, z_0) - \psi_{Np}(x, z_0)_+ + \nabla_x \varsigma_{Np}(x, z_0)'h + \nabla_{z_0} \varsigma_{Np}(x, z_0)'\xi,$$
$$\max_{j=1,\dots,q} \{ f^j(x) - \psi_{Np}(x, z_0)_+ + \nabla f^j(x)'h \} \big] \big\} + \frac{1}{2}\|h\|^2 + \frac{1}{2}\xi^2 \Big\}. \tag{48}$$

We note that the calculation of $\theta_{Np}(x, z_0)$ requires the solution of a convex quadratic program and can therefore be carried out in finite time (Royset, 2010a). We use these terms in the definition of Algorithms 4 and 5 below.

1. Algorithm 4: Adaptive Sample-Size Algorithm

Algorithm 4 is similar to one described in Royset (2010a), but includes the use of smoothing; see Royset (2010b). Let $A^n_{Np}(x, z_0)$ denote the solution from n iterations, starting from x and z_0, of a standard nonlinear solver applied to **BP**$_{Np}$.

Algorithm 4:

Data: N_0 initial sample size, $p_0 > 0$ initial smoothing parameter, $s \in (0.1]$ sample-size increase factor, x_0 initial design vector, $\varepsilon > 0$ error control threshold, $e \in (0,1)$ decrease factor for ε, sample set $\Omega = (v^1, v^2, ...)'$ generated by independent sampling of V, and n an integer.

```
Step 0: Set $i = 0$ and $z_{0i} = 0$.

Step 1: Compute $(x_{i+1}, z_{0i+1}) = A^n_{N_i p_i}(x_i, z_{0i})$

Step 2: Compute $\theta_{N_i p_i}(x_{i+1}, z_{0i+1})$ and $\psi_{N_i p_i}(x_{i+1}, z_{0i+1})$

    If $\theta_{N_i p_i}(x_{i+1}, z_{0i+1}) \geq -\varepsilon$ and $\psi_{N_i p_i}(x_{i+1}, z_{0i+1}) \leq \varepsilon$

        Set $\tilde{N} = \min\{\lfloor sN_i \rfloor, 1\times10^4\}$,

        $N_{i+1} = N_i + \tilde{N}$,

        $p_{i+1} = N_{i+1}/1000$,

        and replace $\varepsilon$ by $\varepsilon e$.

    else

        $N_{i+1} = N_i$, $p_{i+1} = p_i$.
```

Step 3: Replace i by $i+1$ and go to Step 1.

2. Algorithm 5: Adaptive Sample Size Algorithm with Active-Set Strategy

Algorithm 5 is a modification of Algorithm 4. In this algorithm we only consider a working set of sample points, denoted by $S,$ in each iteration. We use the same parameters as in Algorithm 4, but additionally introduce a parameter γ that we use to determine which sample points to include in S. Algorithm 5 is identical to Algorithm 4 when $\gamma = \infty$. Let $A^n_{NpS}(x, z_0)$ denote the solution from n iterations, starting from x and z_0, of the solver as applied to the following problem:

$$\mathbf{BP}_{NpS}: \quad \min_{x,\, z_0} f(x)$$

$$z_0 + \frac{1}{N(1-\alpha)} \sum_{j \in S} \delta_p^j(x, z_0) \le 0, \tag{49}$$

$$x \in X.$$

We also define $\theta_{NpS}(x, z_0)$ by (48) where (46) is replaced by

$$\psi_{NpS}(x, z_0) = \max\left\{ \varsigma_{NpS}(x, z_0),\ \max_{j \in 1,\dots,q} f^j(x) \right\}, \tag{50}$$

where

$$\varsigma_{NpS}(x, z_0) = z_0 + \frac{1}{N(1-\alpha)} \sum_{j \in S} \delta_p^j(x, z_0). \tag{51}$$

Algorithm 5:

Data: As in Algorithm 4 and $\gamma > 0$.

Step 0: Set $i = 0$, $z_{0i} = 0$, and $S_i = \{v^1, v^2, \dots, v^{N_i}\}$.

Step 1: Compute $(x_{i+1}, z_{0i+1}) = A^n_{N_i p_i S_i}(x_i, z_{0i})$ and

$$\tilde{S} = \left\{ j \in \{1, 2, \dots, N_i\} \mid g(x_{i+1}, v^j) + \gamma > 0 \right\}.$$

Step 2: Compute $\theta_{N_i p_i \tilde{S}}(x_{i+1}, z_{0i+1})$ and $\psi_{N_i p_i \tilde{S}}(x_{i+1}, z_{0i+1})$

If $\theta_{N_i p_i \tilde{S}}(x_{i+1}, z_{0i+1}) \ge -\varepsilon$ and $\psi_{N_i p_i \tilde{S}}(x_{i+1}, z_{0i+1}) \le \varepsilon$

```
        Set $\tilde{N} = \min\{\lfloor sN_i \rfloor, 1\times 10^4\}$,
        $N_{i+1} = N_i + \tilde{N}$,
        $S_{i+1} = \{ j \in \{1,2,...,N_{i+1}\} \mid g(x_{i+1}, v^j) + \gamma > 0 \}$
        $p_{i+1} = N_{i+1}/1000$, and replace $\varepsilon$ by $\varepsilon e$.
    else
        $N_{i+1} = N_i$, $p_{i+1} = p_i$, and $S_{i+1} = S_i \cup \tilde{S}$.
Step 3: Replace $i$ by $i+1$ and go to Step 1.
```

We next compare the five algorithms described in this chapter on six test examples from the literature.

IV. NUMERICAL EXAMPLES

In order to evaluate the quality of solutions obtained by Algorithms 1–5 we use a method proposed by Royset (2010a). That procedure gives a confidence interval for a measure of distance, called theta, between a design x and a Fritz-John point (e.g., Bertsekas, 1999, pp. 323–335) of **BP** and a confidence interval for constraint violation in **BP**. If the confidence interval for theta for a given x degenerates to the point zero and the constraint violation is nonpositive, then x is a feasible Fritz-John point of **BP**. If the left end point of the confidence interval for theta as well as the right end point of the confidence interval for the constraint violation are close to zero for a specific x, then x is near a Fritz-John point.

We test Algorithms 1–5 on six engineering design examples from the litrature. These examples range from simple models with two design variables to complicated structural designs; see Table 2 for an overview of the examples with number of design variables (# DV), limit state functions (# LS), and random variables (# RV). The examples include a mix of both linear and nonlinear objective and limit-state functions. We refer to the Appendix for detailed information about the examples. We implement Algorithms 1–5 in MATLAB and use the TOMLAB/SNOPT (Holmstrom, 1999) as nonlinear solver. The computations are run on a desktop computer with 3.25 GB RAM and 3.16 GHz processor speed. We use $1-\alpha = 0.001349898$ in examples, which corresponds to the -3 quantile of the standard normal distribution.

Example	Description	# DV	# LS	# RV
1	Analytical example (Hock & Schittkowski, 1981).	2	2	2
2	Design of a cantilever beam subject to horizontal and vertical loads (Eldred & Binchon, 2006).	2	2	4
3	Design of a rectangular short column (Bichon, Mahadevan, & Eldred, 2009).	2	1	3
4	Design of a uniform column of tubular section with hinge joints at both ends subject to a random compressive load (Rao, 2009, pp. 10-14)..	2	2	1
5	Design of a speed reducer (Rao, 2009, pp. 472-473).	7	9	7
6	Vehicle design considering side-impact crashworthiness (Samson et al. 2009).	11	10	7

Table 2. Overview of Examples (# DV Denotes Number of Design Variables, # LS Denotes Number of Limit-state Functions, # RV Denotes Number of Random Variables)

Tables 3–8 show numerical results for Algorithms 1–3 when applied to Examples 1–6. Here, we set algorithm parameters ε, n, p, and υ equal to 0.001, 5, 1000, and 1×10^{-6} respectively. The sample size N is varied. We stop Algorithms 1 and 3 when SNOPT's default optimality tolerance is satisfied. Algorithm 2 stops as specified by its Step 2.

Example 1	Algorithm	Objective Function Value	Solution Time (seconds)	Theta Interval
N = 1000	1	15.867436	5.9	$(-0.0901,0]$
	2	15.867436	0.1	$(-0.0905,0]$
	3	15.867808	1.6	$(-0.0779,0]$
N = 5000	1	15.873157	213.2	$(-0.0163,0]$
	2	15.873157	0.6	$(-0.0165,0]$
	3	15.873291	7.4	$(-0.0217,0]$
N = 10000	1	15.872825	1406.1	$(-0.0042,0]$
	2	15.872824	0.7	$(-0.0032,0]$
	3	15.873006	15.9	$(-0.0027,0]$

Table 3. Results for Algorithms 1–3 on Example 1

Table 3 shows that Algorithms 1–3 with a larger sample size yield a smaller theta interval, which means that the corresponding solutions become closer to a Fritz-John point as N increases. That is, the design quality improves as N increases. However, the solution time for Algorithm 1 increases dramatically with larger sample sizes.

Example 2	Algorithm	Objective Function Value	Solution Time (seconds)	Theta Interval
N = 1000	1	9.592329	10.6	$(-724.3833,0]$
	2	9.592329	0.4	$(-723.3969,0]$
	3	9.592329	24.4	$(-707.6998,0]$
N = 5000	1	9.697515	298.7	$(-379.4498,0]$
	2	9.697515	48.9	$(-385.1690,0]$
	3	9.697515	30.3	$(-384.0893,0]$
N = 10000	1	9.818805	1966.6	$(-230.7896,0]$
	2	9.818805	199.4	$(-225.4224,0]$
	3	9.818805	42.3	$(-228.1197,0]$

Table 4. Results for Algorithms 1–3 on Example 2

Table 4 shows that the designs computed with the given sample sizes are not particularly close to a Fritz-John point as the theta intervals are large. For this example we compute a design with theta interval $(-1.3523,0]$ with a sample of size 2×10^5 in 1207.8 seconds using Algorithm 3.

Example 3	Algorithm	Objective Function Value	Solution Time (seconds)	Theta Interval
N = 1000	1	236.350524	5.3	$(-0.0468,0]$
	2	236.350525	0.1	$(-0.0475,0]$
	3	236.422249	4.9	$(-0.0434,0]$
N = 5000	1	246.780726	152.2	$(-0.0452,0]$
	2	246.780726	0.2	$(-0.0474,0]$
	3	246.794029	13.7	$(-0.0463,0]$
N = 10000	1	247.759968	1020.8	$(-0.0644,0]$
	2	247.760098	1.1	$(-0.0657,0]$
	3	247.769011	30.3	$(-0.0636,0]$

Table 5. Results for Algorithms 1–3 on Example 3

Example 4	Algorithm	Objective Function Value	Solution Time (seconds)	Theta Interval
N = 1000	1	26.726018	2.0	$(-0.6408,0]$
	2	26.726018	0.2	$(-0.6388,0]$
	3	26.726054	5.1	$(-0.6323,0]$
N = 5000	1	26.740881	17.3	$(-0.0916,0]$
	2	26.740881	2.8	$(-0.0916,0]$
	3	26.740904	31.6	$(-0.0982,0]$
N = 10000	1	26.742723	435.8	$(-0.0982,0]$
	2	26.742723	9.8	$(-0.0982,0]$
	3	26.742745	56.9	$(-0.0982,0]$

Table 6. Results for Algorithms 1–3 on Example 4

Example 5	Algorithm	Objective Function Value	Solution Time (seconds)	Theta Interval
N = 1000	1	3469.238339	3.2	$(-8.2179,0]$
	2	3469.238338	0.2	$(-8.2447,0]$
	3	3472.067164	14.1	$(-8.2455,0]$
N = 5000	1	3723.733266	17.8	$(-0.7182,0]$
	2	3723.733267	1.1	$(-0.6977,0]$
	3	3725.084942	86.9	$(-0.7243,0]$
N = 10000	1	3760.540797	1500.0	$(-0.1234,0]$
	2	3760.540798	4.3	$(-0.1256,0]$
	3	3761.796161	156.1	$(-0.1216,0]$

Table 7. Results for Algorithms 1–3 on Example 5

We see in Tables 3–7 that for Algorithm 2, the solution times with different sample sizes are close, meaning that Algorithm 2 is not highly affected by the increase in sample size.

Example 6	Algorithm	Objective Function Value	Solution Time (seconds)	Theta Interval
N = 1000	1	24.597346	2.6	$(-0.4643,0]$
	2	24.597347	0.2	$(-0.4619,0]$
	3	24.605073	7.3	$(-0.4532,0]$
N = 5000	1	24.774812	20.8	$(-0.0288,0]$
	2	24.774808	0.8	$(-0.0289,0]$
	3	24.779162	29.6	$(-0.0265,0]$
N = 10000	1	24.850419	118.0	$(-0.0124,0]$
	2	24.850436	1.9	$(-0.0135,0]$
	3	24.853971	68.8	$(-0.0086,0]$

Table 8. Results for Algorithms 1–3 on Example 6

Tables 3–8 show that the objective function values computed with Algorithms 1 and 2 are close to each other. However, the solution times, which are quite similar for small sample sizes, differ as we increase the sample size. This is because we add one more variable and constraint for every limit-state function for each increment in the sample size. For example, the Example 5 solution-time ratio for Algorithm 1 to Algorithm 2 is 16 for $N = 1000$, while it is 349 for $N = 10000$. Therefore, we conclude that for more complex problems, Algorithm 1 has large memory and longer solution time requirements. For Algorithm 3, the objective function value is worse than that of Algorithms 1 and 2. However, we do not see a dramatic increase in solution times with larger sample sizes for Algorithm 3. We have the largest increase in solution time in Example 4, where the time for $N = 10000$ is 11 times larger than that for $N = 1000$. However, the solution time for Algorithm 1 increases by a factor of 218 for the same example. We next discuss the differences in solutions for different algorithm parameter changes.

Although we see that Algorithm 2 is able to solve Examples 1–6 quickly and with high accuracy, the choice of user-defined algorithm parameters yields differences in solution times. Tables 9–12 present solution times for Algorithm 2 in seconds for different sample sizes as the algorithm parameters change. The row numbers 0.01, 0.001 and 0.0001 and the column numbers from 1 to 10 are the values for ε and n, respectively.

ε	Iteration Limit for the Solver (n)									
	1	2	3	4	5	6	7	8	9	10
0.01	28.7	16.8	8.4	5.7	4.4	4.8	4.4	5.6	6.2	8.4
0.001	29.2	17.3	9.6	7.9	6.3	6.7	7.7	7.1	7.6	9.6
0.0001	29.9	18.2	10.9	8.7	7.0	7.8	7.5	7.8	8.3	10.0

Table 9. Run Times (sec) for Algorithm 2 on Example 5 with $N = 12000$ Given Different Algorithm Parameter Settings

Table 9 shows that as ε increases the solver needs more time. Moreover, the computation times improve from $n=1$ through $n=5$, and gradually deteriorate for larger values of n. But in Table 9 we see that the choice of algorithm parameters does not yield much different solution times for Example 5 with $N = 12000$.

ε	Iteration Limit for the Solver (n)									
	1	2	3	4	5	6	7	8	9	10
0.01	335	212	146	153	187	215	148	662	1157	149
0.001	372	238	357	335	429	632	691	775	1139	1586
0.0001	417	302	632	538	612	769	862	944	1207	1691

Table 10. Run Times (sec) for Algorithm 2 on Example 5 with $N = 120000$ Given Different Algorithm Parameter Settings

We see in Table 10 that with a larger sample the variability in solution times is more evident. Moreover, we see that the results in Table 10 do not follow the characteristics of those on Table 9, where the run times are unevenly increasing or decreasing with respect to parameter choices.

ε	Iteration Limit for the Solver (n)									
	1	2	3	4	5	6	7	8	9	10
0.01	6.24	2.22	2.61	3.44	3.06	3.37	3.08	3.03	3.05	2.99
0.001	4.17	4.65	5.09	5.08	5.12	5.21	5.21	5.28	5.23	5.36
0.0001	6.43	5.18	6.14	6.11	6.09	6.20	6.19	6.24	6.22	6.19

Table 11. Run Times (sec) for Algorithm 2 on Example 6 with $N = 12000$ Given Different Algorithm Parameter Settings

We see in Table 11 that larger ε values yield longer run times. However, we do not see a clear increasing or decreasing pattern in solution times depending on n.

ε	Iteration Limit for the Solver (n)									
	1	2	3	4	5	6	7	8	9	10
0.01	174	85	45	45	46	47	47	47	49	49
0.001	117	130	191	150	152	151	149	151	150	150
0.0001	254	199	305	284	288	287	287	287	287	288

Table 12. Run Times (sec) for Algorithm 2 on Example 6 with $N = 120000$ Given Different Algorithm Parameter Settings

Tables 9–12 show that there is no single parameter value that is best in all examples for Algorithm 2. Thus, we conclude that the parameter choice gives different solution-time results not only for the different examples but also for the same example with different sample sizes. Furthermore, ε values larger than 0.01 effectively lead to large working sets and turn Algorithm 2 into Algorithm 1. We next discuss the effects of parameter choice for Algorithm 2 on solution quality.

We select $\varepsilon = 0.01$ and $n = 1$ as the base parameter choices for Tables 13 and 14. We then optimize with different values of ε and n, and then calculate the absolute differences in the objective function values from that obtained using the base values of ε and n.

ε	Iteration Limit for the Solver (n)									
	1	2	3	4	5	6	7	8	9	10
0.01	0	0.8	0.8	0.09	0.09	0.09	0.09	0.09	0.09	0.09
0.001	136	0.04	25	3360	0.09	0.09	0.09	0.09	0.09	0.09
0.0001	113	4.2	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.09

Table 13. Objective Function Value Differences (in 10^{-7}) for the Parameter Selections with Respect to Objective Function of the Base Parameter Choice (with Algorithm 2 on Example 5 with $N = 12000$)

ε	Iteration Limit for the Solver (n)									
	1	2	3	4	5	6	7	8	9	10
0.01	0	3.7	4.1	3.7	3.7	3.7	3.7	3.7	3.7	3.7
0.001	3.0	6.7	5.3	7.7	3.7	3.7	3.7	3.7	3.7	3.7
0.0001	16	204	11	11	3.7	3.7	3.7	3.7	3.7	3.7

Table 14. Objective Function Value Differences (in 10^{-7}) for the Parameter Selections with Respect to Objective Function of the Base Parameter Choice (with Algorithm 2 on Example 6 with $N = 120000$)

We see in Tables 13 and 14 that there is no dramatic variability in objective function values computed with various parameters. Thus, we conclude that the parameter choice does not affect the quality of the design. We next discuss Algorithm 3 with various smoothing parameter p values.

We apply Algorithm 3 on Example 1 with a sample size of 15000. We optimize the design with different values of p, compute the theta interval in order to evaluate the design quality, and display the results in Table 15.

p	Objective Function Value	Solution Time (seconds)	Theta Interval
1	20.62291723	24.5	$(-1.4575161,0]$
10	16.29547551	17.2	$(-0.5836538,0]$
100	15.88688647	27.2	$(-0.3088083,0]$
1000	15.87386166	30.8	$(-0.0062922,0]$
10000	15.87370318	33.8	$(-0.0064495,0]$
100000	15.87370048	35.4	$(-0.0062150,0]$
1000000	15.87370032	38.5	$(-0.0063207,0]$
10000000	15.87370031	41.1	$(-0.0061170,0]$

Table 15. Algorithm 3 on Example 1 with Different p Values

Table 15 shows that design quality improves as we increase the value of p. However, we do not have much improvement in the theta interval after $p = 1000$. Instead, we see only run-time increases with larger values of p. Since large p values may result in ill-conditioning and slow convergence of the solver, we use $p = 1000$ as default.

We conclude that Algorithm 1 cannot handle large sample sizes due to memory difficulty. In fact, Algorithm 1 solves $\mathbf{BP}_N$ that include more than Nm nonlinear constraints and N variables. Algorithm 2 may need to consider the same number of constraints and variables, but the numerical tests indicate that its active-set strategy is reasonably efficient in practice. On solutions of Examples 1–6, we find that $\mathbf{BP}_N(W)$ contains on average 0.2% of nonlinear constraints that we have in $\mathbf{BP}_N$. Finally, in Algorithm 3 in contrast to Algorithms 1 and 2 we have only one nonlinear constraint associated with limit-state functions and number of design variables does not increase with sample size. Therefore, Algorithms 2 and 3 efficiently compute near-optimal designs for large N. Algorithm 2 can handle large samples, but lacks the ability to deal with extremely large problems. But we apply Algorithm 3 on Example 6 with a sample size of 1×10^7 and compute the design, with a $(-0.0006, 0]$ theta interval in 18 hours. Thus, we conclude that Algorithm 3 is able to solve $\mathbf{BP}_N$ with exceptionally large sample sizes. We next discuss consequences of various algorithm parameters for Algorithms 4 and 5.

We apply Algorithm 4 to Example 6 to see the theta interval differences due to the choice of algorithm parameters. Table 16 presents the results for Algorithm 4 with a run-time limit used as the stopping criterion. That is, the algorithm stops after a specific time, here 15 minutes, and evaluates the quality of the last computed design.

e	s	n	Theta Interval	Final Sample Size
0.1	0.3	15	$(-0.0273379,0]$	10598
0.1	0.3	20	$(-0.0175738,0]$	13777
0.1	0.5	15	$(-0.0089000,0]$	45624
0.1	0.5	20	$(-0.0089010,0]$	45624
0.3	0.3	15	$(-0.0202027,0]$	30267
0.3	0.3	20	$(-0.0079490,0]$	69347
0.3	0.5	15	$(-0.0599124,0]$	45624
0.3	0.5	20	$(-0.0041565,0]$	85624
0.5	0.3	15	$(-0.0116882,0]$	89342
0.5	0.3	20	$(-0.0035743,0]$	69347
0.5	0.5	15	$(-0.0054132,0]$	95624
0.5	0.5	20	$(-0.0077265,0]$	95624

Table 16. Parameter Comparison for Example 6 with Algorithm 4 Starting From Initial $N = 1000$ With 15-Minute Run Time Limit

From Table 16, we see that for larger values of e the sample size tends to increase more compared to smaller e values, which in turn may cause memory difficulty for more complex structural design examples. We obtain better results with $n = 20$ than with $n = 15$ for fixed e and s. We also see that we get generally better results with $s = 0.5$ when the other parameters are hold constant. Therefore, we conclude that $e = 0.1$, $s = 0.5$, and $n = 20$ is an appropriate choice for parameter values and use them in the following calculations. We next present Algorithm 5 results for different γ values.

As discussed in Chapter III, we use γ to determine the active sample points. We apply Algorithm 5 to Example 1 with various γ-values and construct Table 17 after 15-minute runs.

γ	Constraint Violation	Theta Interval	Final Sample Size
0.0000001	$(-\infty$,1.52274)	$(-1.3017056,0]$	75624
0.000001	$(-\infty$,-0.57666)	$(-3.0074925,0]$	55624
0.00001	$(-\infty$,0.00037)	$(-0.0041879,0]$	65624
0.0001	$(-\infty$,0.00009)	$(-0.0017403,0]$	55624
0.001	$(-\infty$,-0.00201)	$(-0.0026550,0]$	45624
0.01	$(-\infty$,0.00037)	$(-0.0004014,0]$	45624
1	$(-\infty$,-0.00028)	$(-0.0001733,0]$	145624
10	$(-\infty$,-0.00024)	$(-0.0009416,0]$	385624
100	$(-\infty$,0.00047)	$(-0.0011301,0]$	425624
1000	$(-\infty$,-0.00123)	$(-0.0009869,0]$	95624
10000	$(-\infty$,0.00056)	$(-0.0007012,0]$	95624
100000	$(-\infty$,0.00050)	$(-0.0009243,0]$	105624
1000000	$(-\infty$,-0.00014)	$(-0.0019971,0]$	125624
10000000	$(-\infty$,0.00026)	$(-0.0018027,0]$	135624

Table 17. γ Comparison With Algorithm 5 on Example 1 with Initial $N = 1000$, $e = 0.1$, $s = 0.5$, $n = 20$ with 15-Minute Run Time Limit

Table 17 shows that solution quality improves as we increase γ from 0.0000001 to 1, but degrades for $\gamma > 1$. Since the smallest theta interval appears to occur at $\gamma = 1$, we want to investigate the values of γ near 1, and use 0.1, 1, and 10, in the following calculations.

We present below the results for Algorithms 4 and 5 on Examples 1–6. In order to show the performance of the algorithms we set a run-time limit as the stopping criterion. We use different run times in accordance with the complexity of the example. For relatively simple Examples 1 and 3, we set the time limit to 15 and 30 minutes, respectively; the time limit is set to 60 minutes for the rest. Rather than setting a time limit, however, the user may prefer to modify Algorithms 4 and 5 to iteratively check the design quality and stop when a user-defined quality level is reached. We start with initial $N = 1000$.

	e	s	n	γ	Final Sample Size	Constraint Violation	Theta Interval
Alg. 4	0.1	0.5	5	-	45624	$(-\infty, 0.00014)$	$(-0.00036, 0]$
	0.3	0.5	5	-	105624	$(-\infty, -0.00104)$	$(-0.00047, 0]$
Alg. 5	0.1	0.5	20	0.1	145624	$(-\infty, -0.00028)$	$(-0.00017, 0]$
	0.1	0.5	20	1	385624	$(-\infty, -0.00024)$	$(-0.00094, 0]$
	0.1	0.5	20	10	425624	$(-\infty, 0.00046)$	$(-0.00113, 0]$

Table 18. Results for Algorithms 4 and 5 on Example 1 with 15-Minute Run Time

We see in Table 18 that we have the best theta interval with Algorithm 5, when $\gamma = 0.1$. The design computed by Algorithm 4 with $e = 0.1$ has also a small theta interval and its constraint violation is less than for Algorithm 5.

	e	s	n	γ	Final Sample Size	Constraint Violation	Theta Interval
Alg. 4	0.1	0.5	20	-	335624	$(-\infty, 0.94085)$	$(-1.24864, 0]$
	0.3	0.5	20	-	205624	$(-\infty, 1.79601)$	$(-2.15182, 0]$
Alg. 5	0.1	0.5	20	0.1	194581	$(-\infty, 0.09473)$	$(-0.54203, 0]$
	0.1	0.5	20	1	2891871	$(-\infty, 0.19126)$	$(-0.66174, 0]$
	0.1	0.5	20	10	3491871	$(-\infty, 0.40577)$	$(-0.60458, 0]$

Table 19. Results for Algorithms 4 and 5 on Example 2 with 60-Minute Run Time

Table 19 shows that the sample size rapidly increases for Algorithm 5. However, Algorithm 5 only considers active constraints. Therefore, Algorithm 4 is more likely to cause memory difficulty for highly complex structural design examples.

	e	s	n	γ	Final Sample Size	Constraint Violation	Theta Interval
Alg. 4	0.1	0.5	20	-	215624	$(-\infty, 0.00000)$	$(-0.00863, 0]$
	0.3	0.5	20	-	205624	$(-\infty, 0.00123)$	$(-0.00676, 0]$
Alg. 5	0.1	0.5	20	0.1	691871	$(-\infty, 7.43356)$	$(-1.67174, 0]$
	0.1	0.5	20	1	1591871	$(-\infty, 0.00034)$	$(-0.00941, 0]$
	0.1	0.5	20	10	2491871	$(-\infty, 0.00095)$	$(-0.00437, 0]$

Table 20. Results for Algorithms 4 and 5 on Example 3 with 30-Minute Run Time

From Table 20, we see that Algorithm 5 with $\gamma = 0.1$ results in an unsatisfactory solution. Since the theta interval is large and the right end point of the constraint violation interval is also large, we conclude that the solution is infeasible.

	e	s	n	γ	Final Sample Size	Constraint Violation	Theta Interval
Alg. 4	0.1	0.5	20	-	85624	$(-\infty, -0.04217)$	$(-0.09810, 0]$
	0.3	0.5	20	-	75624	$(-\infty, 0.07769)$	$(-0.18332, 0]$
Alg. 5	0.1	0.5	20	0.1	119721	$(-\infty, 0.00000)$	$(-0.17360, 0]$
	0.1	0.5	20	1	129721	$(-\infty, 0.52574)$	$(-0.65265, 0]$
	0.1	0.5	20	10	991871	$(-\infty, -0.02971)$	$(-0.09809, 0]$

Table 21. Results for Algorithms 4 and 5 on Example 4 with 60-Minute Run Time

Table 21 shows that for Example 4, the quality of the designs by Algorithm 4 with $e = 0.1$ and Algorithm 5 with $\gamma = 10$ are almost equal.

	e	s	n	γ	Final Sample Size	Constraint Violation	Theta Interval
Alg. 4	0.3	0.5	20	-	11389	$(-\infty, 1.85124)$	$(-1.96464, 0]$
	0.8	0.5	20	-	65624	$(-\infty, 0.21324)$	$(-0.24076, 0]$
Alg. 5	0.8	0.5	20	0.1	129721	$(-\infty, 0.21682)$	$(-0.23135, 0]$
	0.8	0.5	20	1	129721	$(-\infty, 0.04015)$	$(-0.05877, 0]$
	0.8	0.5	20	10	129721	$(-\infty, 0.08343)$	$(-0.10257, 0]$

Table 22. Results for Algorithms 4 and 5 on Example 5 with 60-Minute Run Time

We see in Table 22 that for Example 5, Algorithm 5 yields better theta intervals and small constraint violations than Algorithm 4.

	e	s	n	γ	Final Sample Size	Constraint Violation	Theta Interval
Alg. 4	0.1	0.5	20	-	65624	$(-\infty, 0.00659)$	$(-0.01540, 0]$
	0.3	0.5	20	-	125624	$(-\infty, 0.00000)$	$(-0.00464, 0]$
Alg. 5	0.1	0.5	20	0.1	2491871	$(-\infty, 0.00003)$	$(-0.00168, 0]$
	0.1	0.5	20	1	2691871	$(-\infty, 0.00041)$	$(-0.00183, 0]$
	0.1	0.5	20	10	2591871	$(-\infty, 0.00029)$	$(-0.00068, 0]$

Table 23. Results for Algorithms 4 and 5 on Example 6 with 60-Minute Run Time

From Table 23, we see that we obtain a good design with Algorithm 5 when $\gamma = 10$. The lower end point of the theta interval is almost the same as that obtained after 18 hours using Algorithm 3 with $N = 1 \times 10^7$.

We see in Tables 18–23 that Algorithm 5 is capable of handling larger sample sizes, and thus Algorithm 5 generally computes better designs compared to Algorithm 4. Moreover, because Algorithm 5 only considers the active sample points the iterations take less time compared to Algorithm 4. Therefore, with a stopping criterion based on a user-defined quality level, Algorithm 5 is likely to be faster.

We next present failure probability and buffered failure probability comparisons for a given design. For each example, we select the design with the closest theta interval among the solutions presented in Tables 3–8 and Tables 18–23. We calculate a 95% confidence interval (CI) using MCS for the failure probability. Using the same sample, we also estimate the buffered failure probability. Since we use the same sample, we assume that the error in the buffered failure probability calculation is similar to that in the failure probability estimate. Table 24 presents the resulting estimates. We see that the buffered failure probability is greater than the failure probability for the same design as expected; see Chapter I. On average, the buffered failure probability is 30% of the failure probability.

Example	Design	Theta Interval	CI for Failure Probability	Buffered Failure Probability Estimate
1	8.90895 2.81726	(−0.00017,0]	(0.00052, +/− 0.00005)	0.001329
2	3.88551 2.73362	(−0.54202,0]	(0.000001, +/− 0.0000006)	0.001015
3	9.82582 25	(−0.00437,0]	(0.00052, +/− 0.00005)	0.001402
4	5.45094 0.29593	(−0.09820,0]	(0.00037, +/−0.000036)	0.000971
5	3.6 0.72 19.52866 7.56277 8.28022 3.47997 5.40634	(−0.12156,0]	(0.00047, +/−0.000046)	0.004488
6	0.5 1.43498 0.5 1.25441 1.05796 0.5 0.34 0.345 15 15	(−0.00068,0]	(0.00031, +/−0.00003)	0.001382

Table 24. Computational Comparisons of Failure Probabilities and Buffered Failure Probabilities

V. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

Reliability-based design optimization (RBDO) aims to determine a minimum-cost design for an engineering structure subject to one or more reliability constraints; this thesis considers models having a single reliability constraint. Traditionally, the reliability constraint is given in terms of an upper bound on the failure probability, i.e., the probability that the system performs unsatisfactory. This approach is theoretically and computationally troublesome since a constraint on the failure probability is difficult to deal with in the algorithms used to solve the nonlinear programs arising in RBDO. This thesis considers an alternative approach to RBDO based on "buffered failure probability," and examines five solution algorithms, four of which are developed in this thesis, that use sample-average approximations. Buffered failure probability is more conservative than the traditional failure probability. Thus, a design that satisfies a reliability constraint based on the buffered failure probability also satisfies one based on the failure probability. Finally, the buffered failure probability is much easier to handle in optimization algorithms as it results in smooth and possibly convex optimization problems.

The buffered failure probability approach uses sample averages to estimate expectations. In numerical tests, we show that the sample size needs to be set relatively large to ensure high-quality solutions, and that one standard algorithm may break down because of the resulting memory and run-time requirements. We develop new algorithms that overcome this difficulty and obtain an average speed-up in solution time by a factor of 560 in comparison with the existing methodology based on a standard nonlinear solver. We are able to handle sample sizes two orders of magnitude larger in comparison with the existing method. We also avoid the need for preselecting sample sizes, which can be difficulty in practice, by using adaptive sample-adjustment schemes.

We examine the difference between the failure probability and buffered failure probability approaches in a stochastic knapsack problem as well as other examples and find that for these test examples the buffered failure probability averages typically three times larger than the failure probability for a design computed with buffered failure probability approach. Hence, a design based on the buffered failure probability approach may result in a more costly but more reliable design than that obtained using a failure probability approach. However, in view of the computational difficulties associated with this approach, we believe that the buffered failure probability approach is a viable alternative.

B. SUGGESTED WORK AHEAD

This research area is open to more developments in efficiency and accuracy of the algorithms. We believe the active-set strategy deserves the interest of future researchers with the goal to eliminate its sensitivity to user-specified parameters.

LIST OF REFERENCES

Bertsekas, D. P. (1999). *Nonlinear programing* (2nd ed.). Belmont, MA: Athena Scientific.

Bichon, B. J., Mahadevan, S., & Eldred, M. S. (May 4–7, 2009). Reliability-based design optimization using efficient global reliability analysis. Paper AIAA 2009-2261 in *Proceedings of the 50th AIAA/ASME/ACHE/AHS/ASC Structures, Structural Dynamics, and Material Conference*. Palm Springs,CA.

Choi, S. K., Grandhi, R. V., & Canfield, R. A. (2007). *Reliability-based structural design.* London, UK: Springer-Verlag.

Chung, H., Polak, E., & Sastry, S. (2010). On the use of outer approximations as an external active set strategy. *Journal of Optimization Theory and Application.* doi:10.1007/s10957-010-9655-8

Conn, A. R., Gould, N. I. M., & Toint, P. L. (1992). LANCELOT: A Fortran package for large-scale nonlinear optimization (Release A). *Springer Series in Computational Mathematics, 17.*

Du, X., & Chen, W. (2004). Sequential optimization and reliability assesment method for efficient probabilistic design. *ASME Journal of Mechanical Design, 126,* 225–233.

Eldred, M. S. & Bichon, B. J. (1–4 May, 2006). Second order reliability formulations in DAKOTA/UQ. Paper AIAA 2006-1828 in *Proceedings of the 47th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Material Conference.* Newport, RI.

Gill, P., Murray, W., & Saunders, M. (1998). *User's guide for SNOPT 5.3: A Fortran package for large-scale nonlinear programming. Technical Report SOL-98-1.* Standford, CA: Stanford University.

Hock, W., & Schittkowski, K. (1981). *Test examples for nonlinear programing codes.* Secaucus, USA: Springer-Verlag, New York.

Holmstrom, K. (1999). *The TOMLAB Optimization Environment in MATLAB.* Retrieved 5 November 2009, from http://www.ici.ro/camo/journal/v1n1.htm

Lee, J. O., Yang, Y. O. & Ruy, W. S. (2002). A comparative study on reliability-index and target-performence-based probabilistic structural design optimization. *Computers and Structures, 80,* 257–269.

Liang, J., Mourelatos, Z. P., & Tu, J. (2008). A single-loop method for reliability-based design optimization. *International Journal of Product Development, 5,* 76–92.

Melchers, R. E. (1999). *Structural reliability analysis and prediction* (2nd ed.). Chichester, UK: Wiley.

Polak, E., Womersley, R. S., & Yin, H. X. (2008). An algorithm based on active sets and smoothing for discretized semi-infinite minimax problems. *Journal of Optimization Theory and Applications*, *138*(2), 311–328.

Rao, S. S. (2009). *Engineering optimization theory and practice* (4th ed.). Hoboken, NJ: John Wiley & Sons.

Rockafellar, R. T., & Royset, J. O. (2010). On buffered failure probability in design and optimization of structures. *Reliability Engineering and System Safety, 95* (5), 499–510.

Rockafellar, R. T., & Uryasev, S. (2000). Optimization of conditional value-at-risk. *Journal of Risk, 2,* 21–42.

Rockafellar, R. T., & Uryasev, S. (2002). Conditional value-at-risk for general loss distributions. *Journal of Banking and Finance, 26,* 1443–1471.

Royset, J. O. (2010a). Optimality functions in stochastic programming. Working paper, Operations Research Department, Naval Postgraduate School, Monterey, CA, 2010. Retrieved 5 February 2010, from http://faculty.nps.edu/joroyset/docs/Royset_optimfcn.pdf

Royset, J. O. (2010b). *Algorithms for stochastic programs with random max-functions and applications to engineering design using buffered failure probability.* Working paper, Operations Research Department, Naval Postgraduate School, Monterey CA, 2010.

Royset, J. O., Kiureghian, D., & Polak, E. (2006). Optimal design with probabilistic objective and constraints. *Journal of Engineering Mechanics, 132*(1), 107–118.

Rubinstein, R. Y., & Kroese, D.P. (2008). *Simulation and Monte Carlo method* (2nd ed). New York, NY: Wiley.

Samson, S., Thoomu, S., Fadel, G., & Reneke, J. (30 August-2 September 2009). Reliable design optimization under aleatory and epistemic uncertainties. Paper DETC2009-86473 in *Proceedings of ASME 2009 International Design Engineering Technical Conference & 35th Design Automation Conference.* San Diego.

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

Truncated normal distribution. (n.d.) . Retrieved 9 December 2009 from Wikipedia, http://en.wikipedia.org/wiki/Truncated_normal_distribution

Tu, J., Choi, K. K., & Park, Y. H. (1999). A new study on reliability-based design optimization. *ASME Journal of Mechanical Design*, *121*, 557–564.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX

This appendix describes the six computational examples tested in this thesis.

Example 1: Analytical problem (Hock & Schittkowski, 1981).

DESIGN VARIABLES			
Variable	Description	Lower bound	Upper bound
x_1	Design variable	2	50
x_2	Design variable	0	50

Table 25. Design Variable Descriptions and Bounds for Example 1

RANDOM VARIABLES			
V	Description	Distribution	Parameters (μ, σ)
V_1	Random vector	Normal	(25, 0.03)
V_2	Random vector	Normal	(25, 0.03)

Table 26. Random Variable Descriptions and Distribution Parameters for Example 1

Objective function:

$$\min\ 0.1x_1^2x_2^2$$

Limit-state functions:

$$g_1(x,v) = v_1 - x_1x_2$$

$$g_2(x,v) = v_2 - x_1^2 - x_2^2$$

Example 2: Cantilever beam (Eldred & Binchon, 2006).

DESIGN VARIABLES			
Variable	Description	Lower bound	Upper bound
x_1	Thickness of the beam (cm)	1	4
x_2	Width of the beam (cm)	1	4

Table 27. Design Variable Descriptions and Bounds for Example 2

RANDOM VARIABLES			
V	Description	Distribution	Parameters (μ, σ)
V_1	Yield stress	Normal	$(4\times10^4, 2\times10^3)$
V_2	Young's Modulus	Normal	$(2.9\times10^7, 1.45\times10^6)$
V_3	Horizontal loads (kg)	Lognormal	$(5, 0.5)$
V_4	Vertical loads (kg)	Lognormal	$(5, 0.5)$

Table 28. Random Variable Descriptions and Distribution Parameters for Example 2

Objective function:

$$\min\ x_1 x_2$$

Limit-state functions:

$$g_1(x,v) = \frac{600v_4}{x_1^2 x_2} + \frac{600v_3}{x_1 x_2^2} - v_1$$

$$g_2(x,v) = \frac{4000000}{v_2 x_1 x_2} - \sqrt{\left(\frac{v_4^2}{x_1^4} + \frac{v_3^2}{x_2^4}\right)} - 2.25$$

Example 3: Rectangular short column (Bichon, Mahadevan, & Eldred, 2009).

DESIGN VARIABLES			
Variable	Description	Lower bound	Upper bound
x_1	Column cross section width (cm)	5	15
x_2	Column cross section depth (cm)	15	25

Table 29. Design Variable Descriptions and Bounds for Example 3

RANDOM VARIABLES			
V	Description	Distribution	Parameters (μ, σ)
V_1	Axial force (kg)	Normal	(500,100)
V_2	Bending moment	Normal	(2000,400)
V_3	Yield stress	Lognormal	(5,0.5)

Table 30. Random Variable Descriptions and Distribution Parameters for Example 3

Objective function:

$$\min \; x_1 x_2$$

Limit-state function:

$$g(x,v) = \frac{4v_2}{x_1 x_2 v_3} + \frac{v_1^2}{x_1^2 x_2^2 v_3^2} - 1$$

Example 4: Tubular column (Rao, 2009, pp. 10-14)..

DESIGN VARIABLES			
Variable	Description	Lower bound	Upper bound
x_1	Diameter of the column (cm)	2	14
x_2	Thickness of the tube (cm)	0.2	0.8

Table 31. Design Variable Descriptions and Bounds for Example 4

RANDOM VARIABLES			
V	Description	Distribution	Parameters (μ, σ)
V	Load on the system (kg)	Normal	(2500, 10)

Table 32. Random Variable Descriptions and Distribution Parameters for Example 4

Objective function:

$$\min \ 9.82 x_1 x_2 + 2 x_1$$

Limit-state functions:

$$g_1(x,v) = \frac{v}{\pi x_1 x_2} - 500$$

$$g_2(x,v) = \frac{v}{\pi x_1 x_2} - 1.7\pi^2 (x_1^2 - x_2^2)$$

Example 5: Speed reducer (Rao, 2009, pp. 472-473).

DESIGN VARIABLES			
Variable	Description	Lower bound	Upper bound
x_1	Face width (cm)	2.6	3.6
x_2	Module of teeth (cm)	0.7	0.8
x_3	Number of teeth on pinion	17	28
x_4	Length of shaft 1 (cm)	7.3	8.3
x_5	Length of shaft 2 (cm)	7.3	8.3
x_6	Diameter of shaft 1 (cm)	2.9	3.9
x_7	Diameter of shaft 2 (cm)	5.0	5.5

Table 33. Design Variable Descriptions and Bounds for Example 5

RANDOM VARIABLES			
V	Description	Distribution	Parameters (μ, σ)
V_1	Material property	Normal	$(x_1,\ 0.03)$
V_2	Material property	Normal	$(x_2,\ 0.03)$
V_3	Material property	Normal	$(x_3,\ 0.03)$
V_4	Material property	Normal	$(x_4,\ 0.03)$
V_5	Material property	Normal	$(x_5,\ 0.03)$
V_6	Material property	Normal	$(x_6,\ 0.03)$
V_7	Material property	Normal	$(x_7,\ 0.03)$

Table 34. Random Variable Descriptions and Distribution Parameters for Example 5

Objective function:

$$\min\ 0.7854x_1x_2^2(3.33333x_3^2 + 14.9334x_3 - 43.0934) - 1.508x_1(x_6^2 + x_7^2) +$$
$$7.477(x_6^3 + x_7^3) + 0.7854(x_4x_6^2 + x_5x_7^2)$$

Limit-state functions:

$$g_1(x,v) = \frac{27}{v_1v_2^2v_3} - 1$$

$$g_2(x,v) = \frac{397.5}{v_1v_2^2v_3^2} - 1$$

$$g_3(x,v) = \frac{1.93v_4^3}{v_2v_3v_6^4} - 1$$

$$g_4(x,v) = \frac{1.93v_5^3}{v_2v_3v_7^4} - 1$$

$$g_5(x,v) = \frac{\left[\left(\frac{745v_4}{v_2v_3}\right)^2 + 1.69\times10^7\right]^{0.5}}{0.1v_6^3} - 1100$$

$$g_6(x,v) = \frac{\left[\left(\frac{745v_5}{v_2v_3}\right)^2 + 1..575\times10^8\right]^{0.5}}{0.1v_7^3} - 850$$

$$g_7(x,v) = v_2v_3 - 40$$

$$g_8(x,v) = \frac{(1.5v_6 + 1.9)}{v_4} - 1$$

$$g_9(x,v) = \frac{(1.1v_7 + 1.9)}{v_5} - 1$$

Example 6: Vehicle side-impact crashworthiness problem (Samson et al. 2009).

DESIGN VARIABLES			
Variable	Description	Lower bound	Upper bound
x_1	Thickness of B-pillar inner (cm)	0.5	1.5
x_2	Thickness of B-pillar reinforce (cm)	0.5	1.5
x_3	Thickness of floor side inner (cm)	0.5	1.5
x_4	Thickness of cross member (cm)	0.5	1.5
x_5	Thickness of door beam (cm)	0.5	1.5
x_6	Thickness of door belt line (cm)	0.5	1.5
x_7	Thickness of roof rail (cm)	0.5	1.5
x_8	Material property of B-pillar inner	0.345	0.345
x_9	Material property of floor side inner	0.345	0.345
x_{10}	Barrier hitting height (cm)	15	15
x_{11}	Barrier hitting position (cm)	15	15

Table 35. Design Variable Descriptions and Bounds for Example 6

RANDOM VARIABLES			
V	Description	Distribution	Parameters (μ, σ)
V_1	Material property	Normal	$(x_1,\ 0.03)$
V_2	Material property	Normal	$(x_2,\ 0.03)$
V_3	Material property	Normal	$(x_3,\ 0.03)$
V_4	Material property	Normal	$(x_4,\ 0.03)$
V_5	Material property	Normal	$(x_5,\ 0.03)$
V_6	Material property	Normal	$(x_6,\ 0.03)$
V_7	Material property	Normal	$(x_7,\ 0.03)$

Table 36. Random Variable Descriptions and Distribution Parameters for Example 6

Objective functions:

$$\min\ 1.98 + 4.9x_1 + 6.67x_2 + 6.98x_3 + 4.01x_4 + 1.78x_5 + 2.73x_7$$

Limit-state function:

$$g_1(x,v) = 1.16 - 0.3717v_2v_4 - 0.00931v_2x_{10} - 0.484v_3x_9 + 0.01343v_6x_{10} - 1$$

$$g_2(x,v) = 0.261 - 0.0159v_1v_2 - 0.188v_1x_8 - 0.019v_2v_7 + 0.0144v_3v_5 + 0.0008757v_5x_{10} + 0.080445v_6x_9 + 0.00139x_8x_{11} - 0.00001575x_{10}x_{11} - 0.32$$

$$g_3(x,v) = 0.214 + 0.00817v_5 - 0.131v_1x_8 - 0.0704v_1x_9 + 0.03099v_2v_6 - 0.018v_2v_7 + 0.0208v_3x_8 + 0.121v_3x_9 - 0.00364v_5v_6 + 0.0007715v_5x_{10} - 0.0005354v_6x_{10} + 0.00121x_8x_{11} - 0.32$$

$$g_4(x,v) = 0.74 - 0.61v_2 - 0.163v_3x_8 + 0.001232v_3x_{10} - 0.166v_7x_9 + 0.0227v_2^2 - 0.32$$

$$g_5(x,v) = 28.98 - 3.81v_3 - 4.2v_1v_2 + 0.0207v_5x_{10} + 6.63v_6x_9 - 7.7v_7x_8 + 0.32x_9x_{10} - 32$$

$$g_6(x,v) = 33.86 + 2.95v_3 + 0.1792x_{10} - 5.057v_1v_2 - 11v_2x_8 - 0.0215v_5x_{10} - 9.98v_7x_8 + 22x_8x_9 - 32$$

$$g_7(x,v) = 46.36 - 9.9v_2 - 12.9v_1x_8 + 0.1107v_3x_{10} - 32$$

$$g_8(x,v) = 4.72 - 0.5v_4 - 0.19v_2v_3 - 0.0122v_4x_{10} + 0.009325v_6x_{10} + 0.000191x_{11}^2 - 4$$

$$g_9(x,v) = 10.58 - 0.674v_1v_2 - 1.95v_2x_8 + 0.02054v_3x_{10} - 0.0198v_4x_{10} + 0.028v_6x_{10} - 9.9$$

$$g_{10}(x,v) = 16.45 - 0.489v_3v_7 - 0.843v_5v_6 + 0.0432x_9x_{10} - 0.0556x_9x_{11} - 0.000786x_{11}^2 - 15.57$$

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Fort Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Johannes O. Royset, PhD
   Department of Operations Research
   Monterey, California

4. R. Kevin Wood, PhD
   Department of Operations Research
   Monterey, California

5. Habib Gürkan Başova
   Kara Kuvvetleri Komutanlığı
   Ankara, TURKEY

6. Kara Harp Okulu Kutuphanesi
   Bakanlıklar - 06100
   Ankara, TURKEY

7. Savunma Bilimleri Enstitüsü
   Kara Harp Okulu
   Bakanlıklar, Ankara, TURKEY

8. METU Library
   Ortadogu Teknik Universitesi
   Inonu Blv.
   Ankara, TURKEY

9. Bilkent University Library
   Bilkent Universitesi
   Ankara, TURKEY

10. Ekrem Erdem, PhD
    Iktisadi ve Idari Bilimler Fakultesi
    Melikgazi, Kayseri, TURKEY