Niching Strategies for Multimodal Rare-event Simulation

Hugh Jonathan Kinnear

University College London

A thesis submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

I, Hugh Jonathan Kinnear, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Abstract

Reliability analysis focuses on estimating the small probability of failure of physical systems, that is, the probability that demand exceeds their capacity. There are many modern techniques capable of solving this problem, such as subset simulation, sequential importance sampling and improved cross entropy. Unfortunately, significant challenges arise for these methods when dealing with problems that have a high-dimensional input, a computationally expensive performance function, and crucially, some form of multimodality. This thesis proposes strategies to address multimodality inspired by niching techniques from the field of evolutionary multimodal optimisation.

The thesis also introduces several novel methods that combine niching techniques with concepts from reliability analysis. The foundational component is the niching initial sampling, a robust algorithm that is able to consistently populate all the high density regions of a multimodal reliability problem's failure region. This procedure is then used to develop two novel frameworks. Firstly, niching decomposition subset simulation, suitable for very high-dimensional reliability problems, uses a hill valley test to explicitly decompose the problem into simpler components, demonstrating improved performance over standard existing methods on difficult benchmarks found in the literature including the meatball function counterexample and black swan reliability problems. Secondly the niching model framework, suitable for relatively low-dimensional reliability problems, integrates niching initial sampling with modelling techniques and methods for estimating normalisation constants such as importance sampling, bridge sampling and line sampling. in particular the use of bridge sampling allows for a novel approach to using classification algorithms for reliability analysis. These successful developments and applications demonstrate the power of niching techniques, particularly in high dimensions, and move toward the eventual goal of a meta-algorithm for black-box problems that can determine the most appropriate reliability method based on problem characteristics.

Impact Statement

Computational models are used to measure risk across a wide variety of disciplines. Global climate models predict the probability of extreme weather events, financial models calculate the default risks for creditors and engineering models simulate structures collapsing. Advances in computer science have enabled such models to become more complex, and as a result, there is an increasing demand for rare-event simulation methods that can deal with high-dimensional, computationally expensive, black-box models. In addition to all of these difficulties, some of the most challenging rare-event simulation problems also exhibit multimodal behaviour where multiple distinct regions of the models input space need to be discovered and analysed.

In evolutionary multimodal optimisation, niching techniques are incorporated into evolutionary optimisation algorithms in order to maintain samples in disparate regions of the input space. This work establishes a link between evolutionary multimodal optimisation and rare-event simulation so that niching techniques may be incorporated into rare-event simulation methods. The result is the contribution of new rare-event simulation methods that are explicitly designed to perform well with multimodal computational models. These new methods are tested on a set of multimodal benchmarks and are shown to outperform popular existing techniques. Furthermore, there is a strong link between rare-event simulation methods and sequential Monte Carlo techniques used for Bayesian inference, which could also potentially benefit from these developments.

The first new technique that is introduced is an algorithm called niching initial sampling, which is able to consistently populate all the important regions of a model's high-dimensional input space with no access to any gradient information or prior expert knowledge. This procedure serves as the basis for the rest of the novel techniques introduced in this work and could be used as the first step for many existing rare-event simulation methods. Additionally, niching initial sampling could be applied to solve general black-box optimisation problems outside the field of rare-event simulation. The hill valley graph is another novel tool introduced in this work that could have implication in the field of black-box optimisation. The hill valley graph transforms points in high-dimensional space into a graph which allows users to understand and visualise high-dimensional geometries.

Niching decomposition and the niching model framework are two further original contributions. Niching decomposition takes a challenging rare-event simulation problem and decomposes it into simpler problems which are tractable for existing rare-event simulation methods. This an attractive approach since it is able to benefit from and combine with tried and tested techniques. The niching model framework enables niching techniques to be combined with methods such as importance sampling, bridge sampling and line sampling and unifies them all under one scheme. Additionally, it enables a new way for classification algorithms to be applied to rare-event simulation problems.

This work has led to a publication in a peer-reviewed journal.

Contents

Acknowledgements

I would like to express my gratitude to my supervisor, Alejandro Diaz, for his constant guidance, support and encouragement throughout this research. I would also like to thank my parents for their unconditional love, without which this work would not have been possible.

Nomenclature

$\mathbb{1}_A(\boldsymbol{x})$	Indicator function, returns 1 if $\boldsymbol{x} \in A$, 0 otherwise
$\mathbb{R}$	Real numbers
$\mathbb{R}^d$	The d-dimensional Euclidean space
$\|\boldsymbol{x}\|$	Euclidean norm of vector $\boldsymbol{x}$
$\mathbb{E}_\pi[X]$	Expectation of X with respect to probability density function π
$\mathbb{V}_\pi[X]$	Variance of X with respect to probability density function π
$\mathbb{P}(A)$	Probability of event A
$\mathrm{Cov}(X, Y)$	Covariance of X and Y
$\phi(\cdot)$	Standard normal probability density function
$\phi_d(\cdot)$	Standard d-dimensional normal probability density function
$\Phi(\cdot)$	Standard normal cumulative distribution function
$\mathcal{N}(\mu, \Sigma)$	A normal distribution with mean vector μ and covariance Σ
$\mathcal{U}[a, b]$	A uniform distribution on the interval $[a, b]$
$\mathbf{0}_d$	d-dimensional zero vector
$\boldsymbol{I}_d$	d-dimensional identity matrix
$\sim F$	Distributed according to F
$\overset{\text{iid}}{\sim} F$	Independently and identically distributed according to F

ALP	asynchronous label propagation
BoA	basin of attraction
BS	bridge sampling
CCDF	complementary cumulative density function
CDF	cumulative density function
CE	cross entropy
CLT	central limit theorem
CoV	coefficient of variation
cMH	conditional Metropolis-Hastings
CRP	challenging reliability problem
cSuS	conditional subset simulation
EA	evolutionary algorithm
EM	expectation-maximisation
EMO	evolutionary multimodal optimisation
HLRF	Hasofer–Lind-Rackwitz–Fiessler
HVBoA	hill valley basin of attraction
HVG	hill valley graph
IAT	integrated autocorrelation time

i.i.d.	independently and identically distributed
iCE	improved cross entropy
IS	importance sampling
KDE	kernel density estimation
KL	Kullback–Leibler
LS	line sampling
LSVC	linear support vector classifier
MC	Monte Carlo
MCMC	Markov chain Monte Carlo
MCO	Markov chain optimiser
MH	Metropolis-Hastings
MM	modified Metropolis
MMO	multimodal optimisation
MSR	metastable region
MSE	mean squared error
NBS	niching bridge sampling
NBC	nearest better clustering
NDP	non-degeneracy percentage
NDSuS	niching decomposition subset simulation
NInS	niching initial sampling
NIS	niching importance sampling
NLS	niching line sampling
NMF	niching model framework
NRMSE	normalised root mean squared error
NS	niche surrogate
NSuS	niching subset simulation
PDF	probability density function
pCN	pre-conditioned Crank Nicolson
PRNG	pseudorandom number generator
SIS	sequential importance sampling
SLLN	strong law of large numbers
SNS	standard normal space
SuS	subset simulation
TDOF	two-degree-of-freedom
vMF	von Mises-Fisher
vMFM	von Mises-Fisher mixture
vMFN	von Mises-Fisher-Nakagami
vMFNM	von Mises-Fisher-Nakagami mixture
VRT	variance reduction technique

1 Introduction

1.1 Rare-event simulation

The study of physical, biological, social and economic phenomena frequently demands the study of events that, despite having a negligible probability of occurring, can lead to disastrous consequences. In this context, the objective of rare-event simulation is to estimate small probabilities by generating extreme scenarios. Examples of such rare-events include ruin probabilities for insurance companies [1], large losses for financial traders [2], default risks for creditors [3], extreme heat waves [4] and rainfall [5] across Europe, aircraft collision probabilities [6] and buffer overflows in queueing systems [7]. Whilst all these problems share the same abstract formulation, the associated terminology often depends on the application. Furthermore, there are cases where similar or even identical methods have been developed independently in different domains.

Reliability analysis is concerned with determining the dependability of engineered systems. In this context, the rare-event of interest is when a system fails, such as a structure collapsing after sustaining an intolerable amount of damage. Examples include truss structures [8], vehicle suspension systems [9], wind turbines [10], oscillators [11] and wing box models [12], amongst many others. This work is written from the perspective of reliability analysis, and so will adopt the appropriate naming conventions. However, the methods discussed could be applied to any rare-event simulation problem.

The task of estimating the probability of a rare-event in reliability analysis is called a *reliability problem* and the associated techniques are called *reliability methods*. The reliability problem is now formally defined. Let $d \in \mathbb{N}$ denote the dimension of the reliability problem. This corresponds to the number of inputs describing the physical system under study. Define a set of *random inputs* as a random vector, $\boldsymbol{X} = (X_1, X_2, \ldots, X_d) \in \mathbb{R}^d$, that models the uncertain inputs of a system, where $\mathbb{R}^d$ will be referred as the *input space*. The random inputs are distributed according to the *input density*, a probability density function (PDF) denoted as $f : \mathbb{R}^d \to [0, \infty)$. The rare-event of interest in the reliability analysis context is called the *failure region* and is denoted as $\mathcal{F}$ and random inputs that lie in the failure region will be called *failure samples*. Let the safe region be defined as $\mathcal{S} = \mathbb{R}^d \setminus \mathcal{F}$. The *probability of failure* is given as

$$P_{\mathcal{F}} := \mathbb{P}(\mathcal{F}) = \int_{\mathbb{R}^d} \mathbb{1}_{\mathcal{F}}(\boldsymbol{x}) f(\boldsymbol{x}) d\boldsymbol{x}. \tag{1.1}$$

By definition of the event being rare, it will be assumed that the probability of failure is small, say less than 10^{-3}.

Despite the variety of applications, failure regions, and rare-events in general, can typically be represented by a standard form that will be assumed in this work. The performance of a system is modelled by a *performance function* $g \colon \mathbb{R}^d \to \mathbb{R}$, which assigns

a scalar *performance* to every set of inputs. If the performance of a set of inputs exceeds a *critical threshold* b, then that set of inputs is said to be in the failure region, that is

$$\mathcal{F} := \{\boldsymbol{x} \in \mathbb{R}^d : g(\boldsymbol{x}) \geq b\}. \tag{1.2}$$

This work will follow the convention that $b = 0$. This is done without the loss of generality, since it is possible to translate the performance function without altering the probability of failure. Note that the failure region is sometimes defined with a flipped or strict inequality in other studies.

The conditional density of the random inputs given the failure region is referred to as the *failure density*,

$$f_{\mathcal{F}}(\boldsymbol{x}) := \frac{\mathbb{1}_{\mathcal{F}}(\boldsymbol{x}) f(\boldsymbol{x})}{P_{\mathcal{F}}}. \tag{1.3}$$

Note that the probability of failure appears as the normalisation constant in the expression for the failure density. The task of generating samples from the failure density is closely related to the reliability problem and for many reliability methods it is an essential intermediary step. In some cases, producing samples from the failure density may be an objective in its own right, or even the primary objective. The use of such samples to understand what happens when a system fails is referred to a *probabilistic failure analysis*.

1.2 Aims and objectives

A specific class of reliability problem is now defined, which will be referred to as a *challenging reliability problem (CRP)*. This is not a formal term found elsewhere in reliability analysis literature and is only intended as a convenient label to be used within the context of this work. A CRP has the following properties:

- The performance function is computationally expensive to evaluate. In particular, it is assumed that the cost of any other procedure is negligible, and so the total computational cost is well approximated by the number of calls to the performance function.
- The reliability problem is high-dimensional, say, approximately $d \geq 100$. Additionally, it is not possible to identify an intrinsic low-dimensional structure that approximates the reliability problem. When this is not the case, dimension reduction technique may be employed [13–15].
- The performance function is a black-box. That is, there is no analytical expression available for the performance function or its derivatives and only input-output information is known. A reliability problem with this property will be referred to as a *black-box reliability problem*.
- There is no prior expert knowledge that can be utilised. For instance, sometimes it is assumed that the general shape of the failure region is known a priori [16].
- Augmenting the standard deviation of the random inputs, as was done in [17], is not guaranteed to increase the probability of failure.
- The reliability problem exhibits some form of multimodality such as a multimodal failure density. Section 3.1 will discuss this property in detail.

If the multimodality property is removed from the above list the resultant reliability problem has been well studied and poses no issues for many existing reliability methods. For instance, subset simulation, one of the most popular reliability methods which will be discussed in Section 2.3.1, is perfectly capable of producing an accurate estimate for the probability of failure for such a reliability problem. In fact, there are many CRPs for which subset simulation is still a suitable reliability method. The specific forms of multimodality that can cause problems for existing reliability methods are discussed in Section 3.1 and concrete examples of such reliability problems are explored in Section 3.2.1. It is this deficiency of existing reliability methods that motivates the research aim of this thesis.

Research Aim: To develop reliability methods capable of efficiently and accurately estimating the probability of failure of a CRP.

The term "efficiently and accurately" used in the research aim statement deserves some clarification, which will be provided in Section 3.2.2. In order to achieve this aim, the following research objectives have been pursued:

- Research and experiment with existing reliability methods to understand why they struggle with CRPs.
- Characterise the types of multimodality that are challenging for existing reliability methods.
- Curate a set of benchmark CRPs and decide upon some evaluation criteria so that the performance of competing reliability methods may be compared.
- Understand how niching techniques have been used in the field of evolutionary computation to deal with multimodality and apply those principles to reliability analysis.
- Develop a reliability method that is suitable for very high-dimensional CRPs. That is, a reliability method with a computational complexity that is largely independent of the dimension of the reliability problem. Methods such as subset simulation have this property.
- Develop a reliability method that is capable of explicitly modelling the failure region or failure density of a CRP. Such models often require more parameters in higher dimensions and so such a reliability method would be suitable for relatively low-dimensional CRPs.

1.3 Contributions

The main original contributions of this work are as follows:

- The niching initial sampling procedure. This is a robust initial sampling algorithm that can populate all of the important areas of the failure region of a black-box reliability problem by employing niching techniques.

- Subset simulation (SuS) is enhanced via combination with the niching initial sampling procedure in order to improve performance on CRPs. The resulting algorithm is called niching decomposition subset simulation. In particular, this new algorithm outperforms subset simulation on benchmarks such as the black swan function [18] and the meatball function [19].

- The niching model framework, which integrates the niching initial sampling procedure with modular interchangeable components involving techniques such as importance sampling, bridge sampling and line sampling. A particularly novel aspect of this framework is the use of bridge sampling to facilitate the approximation of the failure density with any binary classification algorithm.

- The hill valley graph, which is an object that may be used to understand the topology of a high-dimensional objective function by transforming samples from input space into graph space.

- The basins of attraction of a reliability problem. This a useful definition for characterising and analysing the behaviour of reliability methods on CRPs.

During the course of this research, parts of the work have been submitted for publication. Specifically, one paper entitled "Niching subset simulation" has been published in Probabilistic Engineering Mechanics [20] and another entitled "Niching importance sampling" is currently under review at Reliability Engineering and System Safety.

1.4 Outline

This thesis has the following structure. Chapter 2 introduces reliability analysis which can be broadly split into two categories. The first category, analytic reliability analysis, is covered in Section 2.1 and focuses on approximating the performance function using its Taylor expansion around important points in the input space. Section 2.2 and Section 2.3 deal with the second category, Monte Carlo reliability analysis, where the former introduces some fundamental principles and the latter shows how those principles have been applied in existing reliability methods.

Multimodality is discussed in Chapter 3. Section 3.1 explores how different reliability methods interact with different types of multimodality and introduces some definitions which are useful for characterising and analysing reliability problems. Concrete examples of CRPs are introduced as a set of benchmarks in Section 3.2 as well as metrics that can be used to evaluate the performance of competing of reliability methods on the benchmarks. Section 3.3 discusses how niching techniques are used within the field of evolutionary computation in order to manage multimodal objective functions.

Chapter 4 introduces novel methods for tackling very high dimensional CRPs that combine niching techniques with subset simulation. Section 4.1 discusses how a reliability problem may be broken down into easier reliability problems with niching techniques and introduces two algorithms: niching initial sampling and niching decomposition subset simulation. The performance of niching decomposition subset simulation on the benchmark reliability problems is analysed in Section 4.2. Section 4.3 discusses the niching subset simulation algorithm. In some sense, niching decomposition subset simulation is the improved successor of niching subset simulation, and so the relative deficiencies of the latter will be explored. The hill valley graph, an important component of niching

subset simulation, is also introduced as a stand alone concept that could potentially be employed elsewhere.

Reliability methods that combine explicit models for the the failure density and the failure region with niching techniques are introduced in Chapter 5. Section 5.1 describes a general modular framework for combining niching techniques with existing modelling techniques. In Section 5.2 specific implementations of this general framework are introduced using importance sampling, bridge sampling, binary classifiers and line sampling. The performance of these specific implementations is then tested and analysed on the benchmark reliability problems in Section 5.3. Chapter 6 concludes the thesis.

2 Reliability analysis

There are a vast number of reliability methods and they may classified according to many different criteria. However, the most important and often drawn distinction is between analytic reliability methods, which will presented first, and Monte Carlo reliability methods, which will be presented second.

2.1 Analytic reliability methods

Analytic reliability methods tend to consist of three distinct stages. Firstly, the reliability problem is transformed into a new reliability problem with an identical probability of failure, but with a different input density which allows for the simplification of the integral that defines the probability of failure. Next, important points in the input space are located via an optimisation procedure that may be used to approximate the failure region. Finally, the performance function is approximated using its Taylor expansion around these important points. The resulting approximate probability of failure may often then be computed analytically. These three steps are now discussed in order.

2.1.1 Standard normal space

Many reliability methods, including both analytic and Monte Carlo reliability methods, rely on the assumption that the random inputs have a standard multivariate normal distribution, that is $\mathcal{N}(\mathbf{0}_d, \boldsymbol{I}_d)$. A reliability problem with this property will be referred to as being in *standard normal space (SNS)*. When a reliability problem is not in SNS, it is often possible to derive a reliability problem with an equivalent probability of failure that is. To do so, a transformation $T : \mathbb{R}^d \to \mathbb{R}^d$ is determined such that

$$P_{\mathcal{F}} = \int_{\mathbb{R}^d} \mathbb{1}_{\mathcal{F}_T}(\boldsymbol{x}) \phi_d(\boldsymbol{x}) d\boldsymbol{x}, \tag{2.1}$$

where $\mathcal{F}_T := \{\boldsymbol{x} \in \mathbb{R}^d : g(T^{-1}(\boldsymbol{x})) \geq 0\}$.

Equation 2.1 implies that for a set of random inputs $\boldsymbol{X} \sim f$, it is a necessary condition that $T^{-1}(\boldsymbol{X}) \sim \mathcal{N}(\mathbf{0}_d, \boldsymbol{I}_d)$. A transformation can accomplish this by applying two complementary concepts. Given a cumulative density function (CDF) denoted by F, the *inversion principle* states that

$$F^{-1}(U) \sim F, \quad \text{for} \quad U \sim \mathcal{U}[0, 1], \tag{2.2}$$

and the *probability integral transform* is given as

$$F(X) \sim \mathcal{U}[0, 1], \quad \text{for} \quad X \sim F. \tag{2.3}$$

The way in which these ideas are combined depends on the provided description of the random input's distribution.

Let $\boldsymbol{X}$ be a set of random inputs distributed according to a joint CDF, denoted as F. Suppose the following conditional CDFs are available,

$$F_{X_i|X_1,\ldots,X_{i-1}}(x_i|x_1,\ldots,x_{i-1}) := \mathbb{P}(X_i \leq x_i|X_1 = x_1,\ldots,X_{i-1} = x_{i-1}), \tag{2.4}$$

for $1 \leq i \leq d$, where for $i = 1$ this is equivalent to the marginal CDF denoted as F_{X_1}. In this case, the *Rosenblatt transformation* [21, 22] may be applied,

$$T_R(x_1,\ldots,x_d) := \begin{pmatrix} \Phi^{-1}(F_{X_1}(x_1)) \\ \Phi^{-1}(F_{X_2|X_1}(x_2|x_1) \\ \vdots \\ \Phi^{-1}(F_{X_d|X_1,\ldots,X_{d-1}}(x_d|x_1,\ldots,x_{d-1})) \end{pmatrix}. \tag{2.5}$$

The random input's distribution may also be presented as a collection of marginal CDFs, $F_{X_i}(x_i)$ for $1 \leq i \leq d$, with the dependency structure described by a normal copula. That is, the joint CDF may be written as

$$F(x_1,\ldots,x_d) = \Phi_{\boldsymbol{R}}\left[\Phi^{-1}(F_{X_1}(x_1)),\ldots\Phi^{-1}(F_{X_d}(x_d))\right] \tag{2.6}$$

where $\Phi_{\boldsymbol{R}}$ is CDF of a multivariate normal with zero mean vector and correlation matrix $\boldsymbol{R}$. In this case the *Nataf transformation* [23–25] may be applied and is given as

$$T_N(x_1,\ldots,x_d) := \boldsymbol{L}^{-1}\begin{pmatrix} \Phi^{-1}(F_{X_1}(x_1)) \\ \Phi^{-1}(F_{X_2}(x_1)) \\ \vdots \\ \Phi^{-1}(F_{X_d}(x_d)) \end{pmatrix}, \tag{2.7}$$

where $\boldsymbol{L}$ is the Cholesky factor of $\boldsymbol{R}$, with $\boldsymbol{L}\boldsymbol{L}^T = \boldsymbol{R}$. In some cases the dependency structure may not be available in this form. However, it may still be possible to model the dependency structure using a normal copula by using numerical techniques to find a correlation matrix that makes Equation 2.6 approximately true.

2.1.2 Design points

Let the *design points* of a reliability problem, denoted as $\boldsymbol{x}_1^*,\ldots,\boldsymbol{x}_k^*$, be the modes of the failure density, where modes are defined as local maxima. Let a *global design point* be a design point that attains the maximum possible probability density under the failure density. Note that there may be multiple global design points and that in reliability analysis literature the term design point often refers exclusively to global design points. Typically, a design point lies in the *limit state surface* defined as

$$\Lambda := \{\boldsymbol{x} \in \mathbb{R}^d : g(\boldsymbol{x}) = 0\}, \tag{2.8}$$

though this is not guaranteed.

Assuming the reliability problem is in SNS and that the design points lie in the limit state surface, an equivalent be definition of design points is the local solutions to the following constrained optimisation problem,

$$\begin{aligned} \min_{\boldsymbol{x}\in\mathbb{R}^d} \quad & \|\boldsymbol{x}\| \\ \text{s.t.} \quad & g(x) = 0. \end{aligned} \tag{2.9}$$

In this context, the reliability index of a design point $\boldsymbol{x}^*$ is given as $\beta = \|\boldsymbol{x}^*\|$. If it is possible to evaluate the gradient of the performance function, methods like the Hasofer–Lind-Rackwitz–Fiessler (HLRF) algorithm [26, 27] or its variants [28] may be employed to solve the constrained optimisation problem. When the gradients are not available, as is the case for the CRP, algorithms of the type that will be discussed in Section 3.3.1 will be required to locate the design points.

Analytic reliability methods attempt to model the failure region using the neighbourhoods of the design points. Denote the balls of radius $\varepsilon > 0$ around each of the design points as

$$B_i^\varepsilon := \{\boldsymbol{x} \in \mathbb{R}^d : \|\boldsymbol{x} - \boldsymbol{x}_i^*\| < \varepsilon\}, \tag{2.10}$$

for $1 \leq i \leq k$. The probability of the intersection of these balls with the failure region can be used to approximate the failure region [29],

$$P_\mathcal{F} \approx \sum_{i=1}^{k} \mathbb{P}(B_i^\varepsilon \cap \mathcal{F}). \tag{2.11}$$

For many reliability problems, and in particular low-dimensional reliability problems, this approximation is justified since the neighbourhoods contain the vast majority of the density of the failure region.

The approximation in Equation 2.11 is not applicable to all reliability problems, and it is especially uncommon for it to be accurate in high-dimensional cases such as a CRP. The following trivial d-dimensional reliability problem in SNS provides some intuition as to why this is the case. Set the failure region as $\mathcal{F} = \mathbb{R}^d$, which means the probability of failure is 1 and that the single design point, $\boldsymbol{x}_1^*$, is at the origin. Let $\boldsymbol{X}$ be a set of random inputs in SNS and fix a $\varepsilon > 0$. It can be shown that

$$\mathbb{P}(B_1^\varepsilon \cap \mathcal{F}) \to 0 \quad \text{as} \quad d \to \infty. \tag{2.12}$$

One way of informally justifying this is to note that $\mathbb{P}(B_1^\varepsilon \cap \mathcal{F}) = \mathbb{P}(\|\boldsymbol{X}\| \leq \varepsilon)$ and that due to the central limit theorem (CLT), $\|\boldsymbol{X}\|$ has an approximately normal distribution for large d [30],

$$\|\boldsymbol{X}\| \approx \mathcal{N}(\sqrt{d}, 1/2). \tag{2.13}$$

That is, in high-dimensional SNS, the vast majority of density is concentrated around a sphere of radius $\sqrt{d}$. This region is called the *important ring* and it is only the parts of the failure region that intersect with it that contribute meaningfully to the probability of failure.

2.1.3 First-order reliability methods

Consider a reliability problem in SNS and where the design points lie in the limit state surface. First-order reliability methods [26] use a first-order Taylor approximation of the performance function around a design point $\boldsymbol{x}^*$ to define a *linearised performance function*,

$$g(\boldsymbol{x}) \approx g_\mathrm{L}(\boldsymbol{x}) := g(\boldsymbol{x}^*) + \nabla g(\boldsymbol{x}^*)(\boldsymbol{x} - \boldsymbol{x}^*). \tag{2.14}$$

This expression may be simplified by noting that $g(\boldsymbol{x}^*) = 0$. Now it can be seen that the failure region associated with the linearised performance function depends only on the direction of $\nabla g(\boldsymbol{x}^*)$ and not its size.

The direction of gradient of the performance function at the design point can be found by applying the Lagrange multiplier method to the constrained optimisation problem defined in Equation 2.9. This results in the following objective function that should be minimised,

$$\mathcal{L}(\boldsymbol{x}, \lambda) = \boldsymbol{x}^T \boldsymbol{x} + \lambda g(\boldsymbol{x}). \tag{2.15}$$

Setting the derivative to 0 implies that

$$\nabla g(\boldsymbol{x}^*) = -\frac{2}{\lambda}(\boldsymbol{x}^*)^T. \tag{2.16}$$

This shows that the gradient of the performance function at the design point is collinear with the design point. Moving from the design point towards the origin should not increase the value of the performance otherwise this would contradict the definition of the design point. This implies that $\lambda < 0$ and that gradient of the performance function at the design point must point in the same direction as the design point.

The failure region associated with the linearised performance function may now be written as,

$$\mathcal{F}_{\mathrm{L}} := \{\boldsymbol{x} \in \mathbb{R}^d : (\boldsymbol{x}^*)^T \boldsymbol{x} \geq \beta^2\}, \tag{2.17}$$

where β is the reliability index of $\boldsymbol{x}^*$. Due to the rotational symmetry of SNS, the coordinate axis may be rotated without affecting the probability of failure, from say $\boldsymbol{x} = (x_1, \ldots, x_d)$ to $\boldsymbol{z} = (z_1, \ldots, z_d)$. If this is done such that z_1 points in the direction of the design point, then it can be seen that membership of the linearised failure region depends entirely on the first coordinate. In particular, it may be written as

$$\mathcal{F}_{\mathrm{L}} = \{(z_1, \ldots, z_d) : z_1 \geq \beta\}. \tag{2.18}$$

It is clear that the probability of the failure can be now simply computed using the complement of the standard normal CDF. In summary, first-order reliability methods use the following approximation:

$$P_{\mathcal{F}} \approx \mathbb{P}(\mathcal{F}_{\mathrm{L}}) \tag{2.19}$$

$$= \int_{\mathcal{F}_{\mathrm{L}}} \phi_d(\boldsymbol{x}) d\boldsymbol{x} \tag{2.20}$$

$$= \int_{\mathcal{F}_{\mathrm{L}}} \phi_d(\boldsymbol{z}) d\boldsymbol{z} \tag{2.21}$$

$$= \int_{\beta}^{\infty} \phi(z_1) dz_1 \tag{2.22}$$

$$= \Phi(-\beta). \tag{2.23}$$

If there are multiple design points, $\boldsymbol{x}_1^*, \ldots, \boldsymbol{x}_k^*$, then this approximation may be made for each one, where the results are summed for the final probability of failure estimates,

$$\mathcal{F}_{\mathrm{L}} \approx \sum_{i=1}^{k} \Phi(-\beta_i) \tag{2.24}$$

where $\beta_1, \ldots, \beta_k$ are the respective reliability indexes. In this case, it is possible that the linearised failure regions overlap. For many reliability problems, this is not an issue, since the density in the overlapping sections tends to be negligible. However, if any pair of design points are very close to one another, the overlap may contain a significant amount of density and so some adjustment should be made to avoid over-estimating the probability of failure.

Second-order reliability methods [31] attempt to improve the accuracy of the approximation made by first-order reliability methods by incorporating the local curvature of the limit state surface,

$$g(\boldsymbol{x}) \approx g(\boldsymbol{x}^*) + \nabla g(\boldsymbol{x}^*)(\boldsymbol{x} - \boldsymbol{x}^*) + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^*)^T \nabla^2 g(\boldsymbol{x}^*)(\boldsymbol{x} - \boldsymbol{x}^*), \tag{2.25}$$

where $\nabla^2 g(\boldsymbol{x}^*)$ is the Hessian at the design point. Unfortunately, the approximate failure region obtained by substituting the performance function for this approximation does not in general yield a probability of failure with a closed-form solution like in the first-order case. A common strategy is to employ an asymptotic formula as $\beta \to \infty$ [29].

There has been some debate as to how useful design points and the analytic methods that rely on them are for high-dimensional reliability problems. This is largely because, as mentioned before, the neighbourhood of design points are generally not within the high density region of the input space. It has been shown in some studies, for some specific high-dimensional reliability problems, that first-order reliability methods can give inaccurate estimates for the probability of failure [30, 32]. However, it has also been argued [33, 34] that this has more to do with the specific reliability methods that were employed, not the concept of design points in general.

2.2 Monte Carlo methods

From now on, this work will be mainly focussed on Monte Carlo (MC) reliability methods rather than analytic reliability methods. This is principally due to the properties of the CRP. Firstly, the high dimensionality of the CRP makes the concept of a design point in general less attractive as previously discussed. Secondly, since there is no gradient information available, the optimisation algorithms that are typically used to locate the design point can not be employed. Thirdly, the multimodality of the CRP may imply the existence of multiple design points, which can be difficult to account for, especially when their high density neighbourhoods overlap. It may be possible to overcome these issues in many cases. However, MC reliability methods have properties that make them more natural candidates for tackling a CRP.

Let $\boldsymbol{X} \in \mathbb{R}^d$ be a random vector distributed according to p, a PDF, and let $h : \mathbb{R}^d \to \mathbb{R}$ be a scalar function. Denote the first two moments of the distribution of $h(\boldsymbol{X})$ as

$$\mu := \mathbb{E}[h(\boldsymbol{X})] \quad \text{and} \quad \sigma^2 := \mathbb{V}[h(\boldsymbol{X})], \tag{2.26}$$

where it will be assumed that both μ and σ^2 exist and are finite. The estimation of μ is a fundamental task in reliability analysis, principally because the probability of failure may be written in the above form,

$$P_{\mathcal{F}} = \mathbb{E}_f[\mathbb{1}_{\mathcal{F}}(\boldsymbol{X})]. \tag{2.27}$$

However, it is often the case that this expectation is not estimated directly. In fact, simulation reliability methods can be characterised according to how they decompose the above expectation into new expectations that are easier to estimate. These estimations are carried out by MC methods, the first step of which is to to simulate random vectors distributed according to p.

2.2.1 Simulation

In general, the goal of a *simulation method* is to generate random variables according to a target distribution. The specific task considered here will be to efficiently generate independently and identically distributed (i.i.d.) random vectors,

$$\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N \overset{\text{iid}}{\sim} p. \tag{2.28}$$

All simulation methods require the generation of i.i.d. uniform samples,

$$U_1, \ldots, U_n \overset{\text{iid}}{\sim} \mathcal{U}[0,1]. \tag{2.29}$$

This can be accomplished by using a *pseudorandom number generator (PRNG)*. Examples of popular PRNGs used in modern statistical software include the Mersenne Twister [35] and the permuted congruential generator [36]. These algorithms do not yield truly random samples. Instead, PRNGs are deterministic algorithms that produce a sequence of numbers starting from a seed. The resultant samples are only random in the sense that they are able to pass some chosen statistical tests that ensure particular features of the uniform distribution are present. As a result, there can be problems when the downstream task that uses the samples requires a property of the uniform distribution that is not guaranteed.

There are many ways to use uniform random samples in order to generate samples with a target distribution. One simple approach is the apply the inversion principle, similar to the Rosenblatt and Nataf transformations. For instance, suppose that the components of a random vector, $\boldsymbol{X} = (X_1, \ldots, X_d)$, are independent with marginal CDFs denoted by F_{X_i} for $1 \leq i \leq d$. Then the inversion principle implies that

$$(F_{X_1}^{-1}(U_1), \ldots, F_{X_d}^{-1}(U_d)) \sim p, \quad \text{for} \quad U_1, \ldots, U_d \overset{\text{iid}}{\sim} \mathcal{U}[0,1]. \tag{2.30}$$

This simulation method requires independent components and that the inverse CDFs may be efficiently evaluated, which is not the case for many standard distributions.

An alternative approach is to apply the rejection principle. Suppose there exists a PDF, denoted by q, that can be efficiently evaluated and sampled from and a constant c such that

$$p(\boldsymbol{x}) \leq c \cdot q(\boldsymbol{x}) \quad \text{for all} \quad \boldsymbol{x} \in \mathbb{R}^d. \tag{2.31}$$

Then the following procedure may be used to produce a sample from the required distribution. First, sample $\boldsymbol{X}' \sim q$ and $U \sim \mathcal{U}[0,1]$. Next, check the condition

$$U < \frac{p(\boldsymbol{X}')}{c \cdot q(\boldsymbol{X}')}. \tag{2.32}$$

If true, then $\boldsymbol{X}' \sim p$, otherwise, reject $\boldsymbol{X}'$ and repeat the process. This approach requires not only the identification of q and c, but also the ability to evaluate p, up to a normalisation constant. The constant c should be chosen as small as possible in order to reduce the probability a candidate is rejected.

The inversion and rejection principles are used as fundamental components in many simulation techniques, though they are rarely used on their own to sample from a standard distribution. In practice, the specific properties of the required distribution are often taken advantage of. A simple example of this type of approach is the Box-Muller transform, which is used to generate samples from a standard normal distribution [37]. Given $U_1, U_2 \overset{\text{iid}}{\sim} \mathcal{U}[0,1]$ the transform is given as

$$Z_1 := \sqrt{-2\ln U_1} \cdot \cos(2\pi U_2), \tag{2.33}$$

$$Z_2 := \sqrt{-2\ln U_1} \cdot \sin(2\pi U_2), \tag{2.34}$$

where $Z_1, Z_2 \overset{\text{iid}}{\sim} \mathcal{N}(0,1)$.

Throughout the rest of this work, it will be assumed that it is is possible to efficiently sample from standard distributions. However, in reliability analysis, it is often the case that the target density p does not have a closed form expression to take advantage of, is computationally expensive to evaluate and is only known up to a normalisation constant. The next section deals with such cases.

2.2.2 Markov chains

In this work, a *Markov chain* will be defined as a collection of indexed random vectors, referred to as *states* and denoted by $\{\boldsymbol{X}_i \in \mathbb{R}^d : i \in \mathbb{N}\}$, where the distributions of the states are defined as follows: the seed $\boldsymbol{X}_1$ is distributed according to a *seed distribution* and for $i \geq 1$,

$$\boldsymbol{X}_{i+1} \sim \tau(\cdot|\boldsymbol{X}_i), \tag{2.35}$$

where $\tau(\cdot|\cdot)$ is a conditional PDF called the transition density. Note that this process is time-homogenous since the transition density does not change and that it possesses the Markov property: the distribution of each state only depends on the immediately preceding random state.

A Markov chain is said to have p as its *stationary distribution* if p satisfies the *stationary equation*,

$$p(\boldsymbol{y}) = \int_{\mathbb{R}^d} \tau(\boldsymbol{y}|\boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x}. \tag{2.36}$$

The stationary equation implies that if

$$\boldsymbol{X}_1 \sim p, \quad \text{then} \quad \boldsymbol{X}_2, \boldsymbol{X}_3, \ldots \sim p. \tag{2.37}$$

A Markov chain is referred to as *stationary* if all of its states are distributed according to the stationary distribution.

A Markov chain will be referred to as ergodic if for any seed, given enough time, it is able to traverse all the important regions of its stationary distribution. The ergodic property has a formal technical definition that will not be required for this work. Suppose a Markov chain is ergodic with stationary distribution p. In this case, the distributions of the Markov chain's states approach the stationary distribution,

$$\lim_{i \to \infty} \|p_i - p\|_{\text{TV}} = 0, \tag{2.38}$$

where p_i is the density function of $\boldsymbol{X}_i$ and the norm is the total variation.

A *Markov chain Monte Carlo (MCMC) algorithm* attempts to sample from a target distribution p by generating a Markov chain with p as its stationary distribution. Clearly, such algorithms are only able to yield a finite number of a samples, and so the truncated process $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ is referred to as a Markov chain of length N. The dependent samples produced by a MCMC algorithm are generally less effective for constructing estimators than the independent samples yielded by the simulation methods described in Section 2.2.1. This idea will be discussed in greater detail in Section 2.2.3. One factor that determines the extent of this inefficiency is how quickly the Markov chain traverses the important regions of the stationary distribution. That is, the Markov chain being ergodic is not sufficient for a good estimate in practical scenarios where the number of samples is finite. Instead, the rate of traversal, often referred to as the *mixing speed*, is crucial. When the mixing speed of a MCMC algorithm is so slow that it becomes unviable to use its samples to construct an estimator, the MCMC algorithm is referred to as having *ergodic issues*.

In most practical scenarios, MCMC algorithms are not able to generate stationary Markov chains since it is not possible to sample a seed from the stationary distribution since sampling from the stationary distribution is the ultimate goal. However, Equation 2.38 implies that after some point, an ergodic Markov chain will become approximately stationary regardless of the seed distribution. For this reason, MCMC algorithms commonly have *burn-in* phase where the Markov chain is generated but the samples are discarded. After the burn-in phase is complete, the samples are kept and used to construct estimators.

In general, MCMC algorithms do not explicitly define their transition density. Instead it is defined implicitly through the description of a generative process. Such a generative process, that takes in the current state and outputs a new state, will be referred to as a *step*. A common strategy is to design a process such that the resultant transition density satisfies the *detailed balance equation*,

$$\tau(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x}) = \tau(\boldsymbol{x}|\boldsymbol{y})p(\boldsymbol{y}). \tag{2.39}$$

The reason for this is that the detailed balance equation is sufficient to imply that the stationary equation holds.

One of the most fundamental MCMC algorithms is the Metropolis-Hastings (MH) algorithm [38,39]. The power of the MH algorithm is that it can sample from any density, given that it is possible to evaluate point-wise up to some normalisation constant. Given the current state $\boldsymbol{x}$, a conditional PDF called the *proposal density* $\xi(\cdot|\boldsymbol{x})$ and a target density p, the MH algorithm is defined by the following generative process that can be shown to produce a transition density that satisfies the detailed balance equation:

Metropolis-Hastings:

1. Sample a candidate according to the proposal density, $\boldsymbol{x}' \sim \xi(\cdot|\boldsymbol{x})$.
2. Compute the *acceptance probability*,
$$A(\boldsymbol{x}', \boldsymbol{x}) := \min\left(1, \frac{p(\boldsymbol{x}')\xi(\boldsymbol{x}|\boldsymbol{x}')}{p(\boldsymbol{x})\xi(\boldsymbol{x}'|\boldsymbol{x})}\right). \tag{2.40}$$
3. Generate a uniform sample, $u \sim \mathcal{U}[0, 1]$.

4. Accept or reject the candidate,

$$\boldsymbol{x}' \leftarrow \begin{cases} \boldsymbol{x}' & u \leq A(\boldsymbol{x}', \boldsymbol{x}), \\ \boldsymbol{x} & \text{otherwise.} \end{cases} \tag{2.41}$$

5. Return $\boldsymbol{x}'$.

The proposal density must be chosen such that it is possible to efficiently generate form it. One of the standard choices for the proposal density is the *normal proposal density*, $\xi(\cdot|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}, \sigma_{\mathrm{p}}^2 \boldsymbol{I}_d)$, where $\sigma_{\mathrm{p}} > 0$ is a parameter called the *proposal scale*. The value of the proposal scale can have a large effect on the speed at which the Markov chain mixes. One strategy for tuning the proposal scale is to monitor the *acceptance rate* during the burn-in phase, where the acceptance rate is the number of candidates that get accepted compared the length of the chain.

The acceptance rate is considered for the following reason. Suppose the chain is currently inside a high density region of the stationary distribution. If the proposal scale is chosen to be small, the candidate samples will typically have a relatively high probability density since they are likely to also lie in the high density region of the stationary distribution. As a result, the acceptance rate will be relatively high. However, this also means that Markov chain will move slowly through the input space. On the other hand, when the proposal scale is large, accepted candidates will make large jumps around the input space. However, the acceptance rate will usually be lower, since many of the candidates will not lie in a high density region and so will be rejected. Therefore, by monitoring the acceptance rate, it may be determined if the proposal scale is too large or too small. The optimal target acceptance rate depends on the particular problem being considered, but some heuristics based on theoretical results have been derived [40].

2.2.3 Estimators

Given $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N \overset{\text{iid}}{\sim} p$, the *MC estimator* is defined as

$$\mu \approx \hat{\mu} := \hat{\mu}_p[h(\boldsymbol{x})] := \frac{1}{N} \sum_{i=1}^{N} h(\boldsymbol{X}_i). \tag{2.42}$$

The MC estimator has many desirable properties. Firstly, it is *unbiased*,

$$\mathbb{E}[\hat{\mu}] = \mu, \tag{2.43}$$

and has a variation that tends to 0 as then number of samples increases,

$$\mathbb{V}[\hat{\mu}] = \frac{\sigma^2}{N} \to 0 \text{ as } N \to \infty. \tag{2.44}$$

Taken together, these properties imply that the MC estimator is *consistent*,

$$\mathbb{P}(|\hat{\mu} - \mu| > \varepsilon) \to 0 \text{ as } N \to \infty \quad \text{for any} \quad \varepsilon > 0, \tag{2.45}$$

and *converges in mean squared error (MSE)*:

$$\mathrm{MSE}[\hat{\mu}] = \mathbb{E}\left[(\hat{\mu} - \mu)^2\right] \to 0 \text{ as } N \to \infty. \tag{2.46}$$

Informally, these properties imply that the estimator will, on average, give better estimates as the sample size increases. Clearly this is a desirable trait for an estimator to posses. However, convergence in MSE does not provide any guarantees of accuracy for any single estimator, only the average of estimators.

Fortunately, the *strong law of large numbers (SLLN)* states that an MC estimator converges to its target almost surely, written as

$$\mathbb{P}\left(\hat{\mu} \to \mu \text{ as } N \to \infty\right) = 1. \tag{2.47}$$

This is a stronger property that implies, informally, as more samples are added the estimate will become more accurate for almost every individual MC estimator. Another useful property of the MC estimator, due to the CLT, is that its distribution can be reasonably approximated using a normal distribution for large N,

$$\hat{\mu} \approx \mathcal{N}\left(\mu, \mathbb{V}[\hat{\mu}]\right). \tag{2.48}$$

Now the MC estimator is generalised to the *MCMC estimator*. Let $((\boldsymbol{X}_i^{(j)})_{i=1}^{n_s})_{j=1}^{n_c}$ be n_c independent stationary Markov chains of length n_s with stationary distribution p and let $N = n_s n_c$. The special case of the MC estimator is recovered when $n_c = N$. Define the MCMC estimator as

$$\mu \approx \hat{\mu} := \hat{\mu}_p[h(\boldsymbol{x})] := \frac{1}{N} \sum_{j=1}^{n_c} \sum_{i=1}^{n_s} h(\boldsymbol{X}_i^{(j)}). \tag{2.49}$$

Note that the notation from the MC estimator is being reused here in the general case. When the distinction is important, it will be made clear if $\hat{\mu}$ is specifically a MC estimator.

The MCMC estimator is unbiased,

$$\mathbb{E}[\hat{\mu}] = \mu, \tag{2.50}$$

and its variance is given by

$$\mathbb{V}[\hat{\mu}] = \frac{\sigma^2}{N} \cdot \gamma(n_s) \tag{2.51}$$

where the *integrated autocorrelation time (IAT)* is

$$\gamma(n_s) = 1 + 2 \sum_{k=1}^{n_s - 1} \left(1 - \frac{k}{n_s}\right) \frac{\gamma_k}{\sigma^2} \tag{2.52}$$

and the covariance between samples k steps apart is

$$\gamma_k := \mathrm{Cov}(h(\boldsymbol{X}_1^{(1)}), h(\boldsymbol{X}_{1+k}^{(1)})). \tag{2.53}$$

Under certain assumptions, the MCMC estimator has many of the same properties as the MC estimator [41]. When the Markov chains are ergodic, the MCMC estimator converges to its target almost surely due to the SLLN for MCMC. When the Markov chains are *geometrically ergodic*, the distribution of the MCMC estimator may be approximated by a normal distribution due to the CLT for MCMC. Loosely speaking, geometrically ergodic means that a Markov chain mixes very quickly.

Comparing the variances of the MC and MCMC estimators reveals why independent samples are preferable where possible for constructing an estimator. Let the *effective sample size* of a MCMC estimator be defined as

$$N_{\text{eff}} = \frac{N}{\gamma(n_s)}. \tag{2.54}$$

That is, the effective sample size is the number of samples a MC estimator would require to achieve the same variance as the MCMC estimator. Typically $\gamma(n_s) > 0$, which implies that $N_{\text{eff}} < N$ and that a MC estimator typically requires fewer computational resources to produce an estimator with a given variance than an MCMC estimator. In general, the quicker a Markov chain mixes, the faster ρ_k will decay as k grows, which means the IAT will grow less slowly with the number of samples, which ultimately results in a larger effective sample size.

Let the *coefficient of variation (CoV)* of the estimator $\hat{\mu}$ be given as,

$$\delta := \delta_p[h(\boldsymbol{x})] := \frac{\sqrt{\mathbb{V}[\hat{\mu}]}}{\mathbb{E}[\hat{\mu}]}. \tag{2.55}$$

Define an estimator of the CoV as follows:

$$\delta \approx \hat{\delta}_p[h(\boldsymbol{x})] := \frac{\hat{\sigma}}{\hat{\mu}} \cdot \sqrt{\frac{\hat{\gamma}(n_s)}{N}}, \tag{2.56}$$

where

$$\hat{\gamma}(n_s) := 1 + 2\sum_{k=1}^{n_s-1}\left(1 - \frac{k}{n_s}\right)\frac{\hat{\gamma}_k}{\hat{\sigma}^2} \tag{2.57}$$

and

$$\hat{\gamma}_k := \left[\frac{1}{n_c(n_s-k)}\sum_{j=1}^{n_c}\sum_{i=1}^{n_s-k} h(\boldsymbol{X}_i^{(j)})h(\boldsymbol{X}_{i+k}^{(j)})\right] - \hat{\mu}^2, \tag{2.58}$$

with $\hat{\sigma}^2 = \hat{\gamma}_0$. Note that in the case of independent samples, $\hat{\gamma}(n_s) = 1$. The CoV is often estimated in reliability analysis and reported as a measure of uncertainty in an estimate.

Since it is typically possible to use a simulation method to generate independent samples from the input distribution, a MC estimator may be used as a reliability method

$$P_{\mathcal{F}} \approx \hat{P}_{\mathcal{F}}^{\text{MC}} := \hat{\mu}_f[\mathbb{1}_{\mathcal{F}}(\boldsymbol{x})]. \tag{2.59}$$

When an indicator function acts on a random vector, the result is a Bernoulli random variable. It follows that the CoV of $\hat{P}_{\mathcal{F}}^{\text{MC}}$ and its estimator is given as

$$\delta_{\mathcal{F}}^{\text{MC}} := \sqrt{\frac{1-P_{\mathcal{F}}}{N \cdot P_{\mathcal{F}}}} \approx \hat{\delta}_{\mathcal{F}}^{\text{MC}} := \sqrt{\frac{1-\hat{P}_{\mathcal{F}}^{\text{MC}}}{N \cdot \hat{P}_{\mathcal{F}}^{\text{MC}}}}. \tag{2.60}$$

When the probability of failure is very small, which is true by definition for any rare-event simulation problem, a very large number of samples are required in order to obtain a desired CoV. Since every sample requires an evaluation of the performance function, which is computationally expensive in the CRP, this results is a very inefficient estimator. There are however attractive features of the MC estimator as a reliability method. The CoV expression does not depend on the dimension of the reliability problem, the input

density or the performance function. This means that the features of the CRP that cause issues for other reliability methods, such as high dimensions and multimodality pose no problems for a MC estimator. Sometimes MC reliability methods are referred to as *variance reduction techniques (VRTs)* sine they aim to reduce the variance of the MC estimator whilst attempting to retain some of the properties that make it so robust.

2.2.4 Importance sampling

In importance sampling (IS), an expectation is rewritten with respect to a PDF called the *importance density* that is denoted as q,

$$\mathbb{E}_p[h(\boldsymbol{x})] = \int_{\mathbb{R}^d} p(\boldsymbol{x})h(\boldsymbol{x})d\boldsymbol{x} = \int_{\mathbb{R}^d} q(\boldsymbol{x})\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}h(\boldsymbol{x})d\boldsymbol{x} = \mathbb{E}_q\left[\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}h(\boldsymbol{x})\right]. \tag{2.61}$$

Define an *IS estimator* as

$$\mu \approx \hat{\mu}^{\mathrm{IS}}(q) := \hat{\mu}_q\left[\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}h(\boldsymbol{x})\right]. \tag{2.62}$$

The idea of the IS estimator is to choose an importance density that makes its variance as small as possible. Assuming that $h(\boldsymbol{x}) \geq 0$ for all $\boldsymbol{x} \in \mathbb{R}^d$, let the *optimal importance density* be given as

$$q^*(\boldsymbol{x}) := \frac{p(\boldsymbol{x})h(\boldsymbol{x})}{\mu}. \tag{2.63}$$

The variance of a IS estimator is minimised by the optimal importance density,

$$\mathbb{V}[\hat{\mu}^{\mathrm{IS}}(q^*)] = 0. \tag{2.64}$$

For an importance density to be viable, it must be possible to both evaluate it pointwise and to generate samples from it efficiently. Clearly it is not possible to evaluate the optimal importance density pointwise, since its normalisation constant is the ultimate quantity of interest. It should be noted however, that it is sometimes possible to sample from the optimal importance chain using a MCMC algorithm. In practice, the importance density is often chosen from a parametric family of densities such that it as close to the optimal importance density as possible.

Given an importance density q, the importance sampling estimator may be used to directly as estimate the probability of failure,

$$P_{\mathcal{F}} \approx \hat{P}_{\mathcal{F}}^{\mathrm{IS}} := \hat{\mu}_q\left[\frac{f(\boldsymbol{x})}{q(\boldsymbol{x})}\mathbb{1}_{\mathcal{F}}(\boldsymbol{x})\right]. \tag{2.65}$$

There are many reliability methods based on this estimator and they are characterised according to how they select the importance density. Note that in this context, the optimal importance density is the failure density. The CoV of $\hat{P}_{\mathcal{F}}^{\mathrm{IS}}$ and its estimator are given by

$$\delta_{\mathcal{F}}^{\mathrm{IS}} := \delta_q\left[\frac{f(\boldsymbol{x})}{q(\boldsymbol{x})}\mathbb{1}_{\mathcal{F}}(\boldsymbol{x})\right] \approx \hat{\delta}_{\mathcal{F}}^{\mathrm{IS}} := \hat{\delta}_q\left[\frac{f(\boldsymbol{x})}{q(\boldsymbol{x})}\mathbb{1}_{\mathcal{F}}(\boldsymbol{x})\right]. \tag{2.66}$$

Another key strategy employed by some reliability methods is the to apply the importance sampling transformation repeatedly. Consider a sequence of PDFs which can be written in terms of their unnormalised form and normalisation constants

$$q_i(\boldsymbol{x}) := \frac{\tilde{q}_i(\boldsymbol{x})}{C_i}, \tag{2.67}$$

for $0 \leq i \leq m$ and where $q_m = f_{\mathcal{F}}(\boldsymbol{x})$ and C_0 is the only normalisation constant that is known. These densities will be referred to as *intermediate densities*. Any reliability method that uses a sequence of intermediate densities will be referred to as an *intermediate method*. The sequence may be used in the following transformation of the probability of failure expectation,

$$\mathbb{E}_f[\mathbb{1}_{\mathcal{F}}(\boldsymbol{x})] = \mathbb{E}_{q_{m-1}}\left[\frac{f(\boldsymbol{x})}{q_{m-1}(\boldsymbol{x})}\mathbb{1}_F(\boldsymbol{x})\right] \tag{2.68}$$

$$= \mathbb{E}_{q_{m-1}}\left[\frac{\tilde{q}_m(\boldsymbol{x})}{q_{m-1}(\boldsymbol{x})}\right] \tag{2.69}$$

$$= C_{m-1} \cdot \mathbb{E}_{q_{m-1}}\left[\frac{\tilde{q}_m(\boldsymbol{x})}{\tilde{q}_{m-1}(\boldsymbol{x})}\right] \tag{2.70}$$

$$= \mathbb{E}_{q_{m-2}}\left[\frac{\tilde{q}_{m-1}(\boldsymbol{x})}{q_{m-2}(\boldsymbol{x})}\right] \cdot \mathbb{E}_{q_{m-1}}\left[\frac{\tilde{q}_m(\boldsymbol{x})}{\tilde{q}_{m-1}(\boldsymbol{x})}\right] \tag{2.71}$$

$$= C_0 \cdot \prod_{i=0}^{m-1} \mathbb{E}_{q_i}\left[\frac{\tilde{q}_{i+1}(\boldsymbol{x})}{\tilde{q}_i(\boldsymbol{x})}\right]. \tag{2.72}$$

Each of the expectations in the product may estimated by a *ratio estimator*, defined as

$$\mathbb{E}_{q_i}\left[\frac{\tilde{q}_{i+1}(\boldsymbol{x})}{\tilde{q}_i(\boldsymbol{x})}\right] \approx \hat{R}_i := \hat{\mu}_{q_i}\left[\frac{\tilde{q}_{i+1}(\boldsymbol{x})}{\tilde{q}_i(\boldsymbol{x})}\right], \tag{2.73}$$

for $0 \leq i \leq m-1$. Now, the probability of failure may be estimated using a *product of ratios estimator*,

$$P_{\mathcal{F}} \approx \hat{P}_{\mathcal{F}}^R := C_0 \cdot \prod_{i=0}^{m-1} \hat{R}_i. \tag{2.74}$$

Any reliability method that uses a product of ratios estimator will be referred to as a *ratio method*. Note that a ratio method clearly has to also be an intermediate method, though the inverse statement is not necessarily true as will be explored in Section 2.3.3. It can be shown that a product of ratios estimator is asymptotically unbiased and consistent [42]. If the ratio estimators are assumed to be independent, then the product of ratios estimator CoV and its estimator are given as

$$\delta_{\mathcal{F}}^R := \sqrt{\sum_{i=0}^{m-1}\left(\delta_{q_i}\left[\frac{\tilde{q}_{i+1}(\boldsymbol{x})}{\tilde{q}_i(\boldsymbol{x})}\right]\right)^2} \approx \hat{\delta}_{\mathcal{F}}^R := \sqrt{\sum_{i=0}^{m-1}\left(\hat{\delta}_{q_i}\left[\frac{\tilde{q}_{i+1}(\boldsymbol{x})}{\tilde{q}_i(\boldsymbol{x})}\right]\right)^2}. \tag{2.75}$$

In order to minimise the the CoV of the product of ratios estimator, the intermediate densities should be chosen so the that consecutive densities in the sequence are close to one another. That is, they give similar densities to the same points in the input space. The intermediate densities can be thought of as a bridge from a density that is easy to sample from and where the normalisation constant is known to the failure density which is hard to sample from and has the probability of failure as its normalisation constant.

2.2.5 Line sampling

Another strategy for transforming the expectation that defines the probability of failure is to use a technique called *line sampling (LS)* [43]. To understand LS, some preliminaries are required. The first step of a LS technique is to identify an *important direction* which is a unit vector denoted by $\alpha \in \mathbb{R}^d$. The identification of such an important direction will be discussed in more detail later, but in general it should point towards a high density region of the failure density. An example of an important direction could be the direction of a design point which, as previously discussed, may be located using an optimisation algorithm.

Define the *orthogonal complement* of the important direction as

$$\boldsymbol{\alpha}^{\perp} := \{\boldsymbol{x} \in \mathbb{R}^d : \boldsymbol{x} \cdot \boldsymbol{\alpha} = 0\}. \tag{2.76}$$

It will be assumed here that that the reliability problem is in SNS. Given a random vector $\boldsymbol{X}$ in SNS, it is possible to generate a random vector which has a standard normal distribution restricted to the orthogonal complement of $\boldsymbol{\alpha}$,

$$\boldsymbol{X}_{\boldsymbol{\alpha}} := \boldsymbol{X} - (\boldsymbol{X} \cdot \boldsymbol{\alpha})\,\boldsymbol{\alpha}. \tag{2.77}$$

Finally, note that due to the rotational symmetry of SNS, the following property holds,

$$\phi_d(\boldsymbol{x}_{\boldsymbol{\alpha}} + t\boldsymbol{\alpha}) = \phi_{d-1}(\boldsymbol{x}_{\boldsymbol{\alpha}})\phi(t) \tag{2.78}$$

for $t \in \mathbb{R}$ and $\boldsymbol{x}_{\boldsymbol{\alpha}} \in \boldsymbol{\alpha}^{\perp}$.

The probability of failure may be rewritten in the following way:

$$P_{\mathcal{F}} = \int_{\mathbb{R}^d} \mathbb{1}_{\mathcal{F}}(\boldsymbol{x})\phi_d(\boldsymbol{x})d\boldsymbol{x} \tag{2.79}$$

$$= \int_{\boldsymbol{\alpha}^{\perp}} \int_{\mathbb{R}} \mathbb{1}_{\mathcal{F}}(\boldsymbol{x}_{\boldsymbol{\alpha}} + t\boldsymbol{\alpha})\phi_d(\boldsymbol{x}_{\boldsymbol{\alpha}} + t\boldsymbol{\alpha})dtd\boldsymbol{x}_{\boldsymbol{\alpha}} \tag{2.80}$$

$$= \int_{\boldsymbol{\alpha}^{\perp}} \left(\int_{\mathbb{R}} \mathbb{1}_{\mathcal{F}}(\boldsymbol{x}_{\boldsymbol{\alpha}} + t\boldsymbol{\alpha})\phi(t)dt \right) \phi_{d-1}(\boldsymbol{x}_{\boldsymbol{\alpha}})d\boldsymbol{x}_{\boldsymbol{\alpha}} \tag{2.81}$$

$$= \mathbb{E}_{\boldsymbol{X}_{\boldsymbol{\alpha}}}\left[P_{\mathcal{F}}^{\boldsymbol{\alpha}}\left(\boldsymbol{X}_{\boldsymbol{\alpha}}\right)\right], \tag{2.82}$$

where

$$P_{\mathcal{F}}^{\alpha}\left(\boldsymbol{x}_{\boldsymbol{\alpha}}\right) = \int_{\mathbb{R}} \mathbb{1}_{\mathcal{F}}(\boldsymbol{x}_{\boldsymbol{\alpha}} + t\boldsymbol{\alpha})\phi(t)dt. \tag{2.83}$$

Using this expression for the probability of failure, the *LS estimator* is given as

$$P_{\mathcal{F}} \approx \hat{P}_{\mathcal{F}}^{\mathrm{LS}} := \hat{\mu}_{\boldsymbol{X}_{\boldsymbol{\alpha}}}\left[P_{\mathcal{F}}^{\boldsymbol{\alpha}}\left(\boldsymbol{X}_{\boldsymbol{\alpha}}\right)\right]. \tag{2.84}$$

In summary, a reliability method that uses a LS must first identify an important direction $\boldsymbol{\alpha}$. Then, some number of samples must be generated from SNS restricted to the orthogonal complement of α using samples from SNS and Equation 2.77. Each of these samples defines a one dimensional reliability problem with the probability of failure given by Equation 5.49. All of these one dimensional reliability problems are then solved and

the resulting estimates for those probabilities of failure are used to estimate the original probability of failure with an MC estimator.

The idea behind this strategy is that if the important direction is chosen well, then the variance of the LS estimator will be lower than the variance of a MC given the same number of samples. As an extreme example, consider the following simple two-dimensional failure region,

$$\mathcal{F} = \{\boldsymbol{x} = (x_1, x_2) \in \mathbb{R}^2 : x_1 > 4\}. \tag{2.85}$$

In this case, if the important direction is chosen as $\boldsymbol{\alpha} = (1, 0)$, the resulting LS estimator will have zero variance, whereas if the important direction is chosen as $\boldsymbol{\alpha} = (0, 1)$ then the LS estimator has the same variance as the MC estimator for the probability of failure.

All that remains is to discuss how the one dimensional reliability problems are solved. Of course, in theory any reliability method may be used. However in practice, whatever methods is used should require very few evaluations of the performance function. If this is not the case, then any gain in computational efficiency that is made by the reduction in variance of the LS estimator compared to the MC estimator will be offset by the computational cost of solving the one dimensional reliability problems. Typically a first-order reliability method is used. More details will be discussed in Section 5.2.3.

2.3 Monte Carlo reliability methods

Now three MC reliability methods are discussed in detail. There are a few reasons why these specific methods have been chosen out of the hundreds of MC reliability methods that have been developed [44]. Primarily they have been chosen since they are able to solve some CRPs. That is, they can handle high-dimensional, black-box reliability problems that exhibit some forms of multimodality. The issue, as will be shown in Section 2.3 where these three methods are tested on a set of benchmarks, is that there are many types of CRPs that are challenging for these methods. Nevertheless, these three methods will serve as very good starting points for developing novel techniques that can deal with CRPs more reliably. Other secondary reasons for their selection include that they are relatively easy to implement with simple control flows and that they are popular methods that are widely cited.

There are many different characteristics of MC reliability methods that may be used to categorise them. In this work, there will be a special focus on the distinction between ratio methods and methods that require the fitting of a model, which will be referred to as *model methods*. Note that in theory these two properties are not mutually exclusive. The reason for this is that the computational cost of a ratio method tends to be largely independent of the dimensionality of a given reliability problem. This makes them suitable for extremely high dimensional reliability problems. On other hand, as the dimension of a reliability problem increases, typically the number of parameters required to accurately model objects of that reliability problem, such as the failure region or failure density, also increases. The more parameters a model has, the more samples that are required to fit them, which typically entails more performance function evaluations. As a result, model methods tend to be more computationally expensive for higher dimensional reliability problems. However, for relatively low-dimensional reliability problems, they have the capacity to be more efficient than ratio methods.

The first two reliability methods presented in this section are ratio methods, which will serve as the starting point for the novel methods in Chapter 4, and the third is a model method, which will serve as the starting point for the novel methods in Chapter 5.

2.3.1 Subset simulation

Subset simulation (SuS) [42] has become one of the most widespread MC reliability methods. To date it has been successfully applied to solve practical reliability problems such as truss structures [45], corroded pipelines [46], slope analysis [47], deteriorating systems [48] and geotechnical structures [49] to name but a few.

SuS is a ratio method that uses the following sequence of intermediate densities,

$$q_i^{\mathrm{SuS}}(\boldsymbol{x}) \propto f(\boldsymbol{x})\mathbb{1}_{\mathcal{F}_i}(\boldsymbol{x}), \tag{2.86}$$

where the $\mathcal{F}_i := \{\boldsymbol{x} \in \mathbb{R}^d : g(\boldsymbol{x}) \geq b_i\}$ are called the *intermediate failure regions* defined by the *intermediate thresholds* $b_i \in \mathbb{R}$ for $0 \leq i \leq m$. The intermediate thresholds are chosen adaptively aside from the first and last which are predetermined as follows: $b_0 = -\infty$ which implies that $\mathcal{F}_0 = \mathbb{R}^d$, q_0^{SuS} is the input density and that $C_0 = 1$ in Equation 2.74; $b_m = 0$ which implies that $\mathcal{F}_m = \mathcal{F}$ and that q_m^{SuS} is the failure density. Because of the particular form of the sequence of densities, in the context of SuS Equation 2.72 may be written as,

$$P_{\mathcal{F}} = \mathbb{P}(\mathcal{F}_1)\mathbb{P}(\mathcal{F}_2|\mathcal{F}_1)\ldots\mathbb{P}(\mathcal{F}_{m-1}|\mathcal{F}_{m-2})\mathbb{P}(\mathcal{F}|\mathcal{F}_{m-1}). \tag{2.87}$$

The SuS procedure generates a random sequence of *levels*

$$(L_0, L_1, \ldots, L_{m-1}), \tag{2.88}$$

where each level is a random matrix, $L_i \in \mathbb{R}^{N_{\mathrm{L}} \times d}$, where N_{L} is a user-defined parameter called the *level size* and $m \in \mathbb{N}$ is a random variable which denotes the number of levels. The rows of each level represent the samples of that level. The object L_i will be referred to as level i for $0 \leq i \leq m-1$. The SuS algorithm is given as follows:

Subset simulation:

1. Generate N_{L} samples from the input distribution. These samples are collectively referred to as the level 0. Set $i = 0$.

2. Evaluate the performance function on all members of level i and label them $(\boldsymbol{x}_j^{(i)})_{j=1}^{N_{\mathrm{L}}}$ such that

$$(\boldsymbol{x}_j^{(i)})_{j=1}^{N_{\mathrm{L}}} \quad \text{such that} \quad g(\boldsymbol{x}_j^{(i)}) \geq g(\boldsymbol{x}_{j+1}^{(i)}) \quad \text{for} \quad 1 \leq j \leq N_{\mathrm{L}} - 1. \tag{2.89}$$

3. If

$$\frac{1}{N_{\mathrm{L}}} \sum_{i=1}^{N_{\mathrm{L}}} \mathbb{1}_F(x_i^{(j)}) \geq p_{\mathrm{L}}, \tag{2.90}$$

where p_{L} is a user-defined parameter called the *level probability*, or $i = n_{\mathrm{limit}}$, where n_{limit} is a user-defined parameter called the *level limit*, then terminate the algorithm.

4. Determine the next intermediate threshold as

$$b_{i+1} = \frac{g(\boldsymbol{x}_{n_c}^{(i)}) + g(\boldsymbol{x}_{n_c+1}^{(i)})}{2}, \tag{2.91}$$

where the *number of chains* is given as $n_c = N_{\mathrm{L}} \cdot p_{\mathrm{L}}$. This also determines the intermediate failure region $\mathcal{F}_{i+1}$.

5. Run a MCMC algorithm with stationary distribution q_{i+1}^{SuS} to generate n_c chains of length $n_s = p_{\mathrm{L}}^{-1}$ where the seeds are $(\boldsymbol{x}_j^{(i)})_{j=1}^{n_c}$. Update $i \leftarrow i+1$. Refer to the collective Markov chains samples as level i. Go to Step 2.

It is typically possible to generate independent samples from the input distribution using a simulation method. This means that the first ratio estimator will be a MC estimator whilst the rest will be MCMC estimators. The level probability must be chosen such that both n_c and n_s are integers.

The ratio estimators that make up the product of ratios estimator have a simplified form in SuS. This is because the adaptive choice of the intermediate threshold implies that most of them will be equal to the level probability, with the exception of the final ratio estimator. Explicitly, the product of ratio estimators for SuS is given as

$$P_{\mathcal{F}} \approx \hat{P}_{\mathcal{F}}^{\mathrm{SuS}} := p_{\mathrm{L}}^{m-1} \cdot \frac{1}{N_{\mathrm{L}}} \sum_{i=1}^{N_{\mathrm{L}}} \mathbb{1}_F(x_i^{(m-1)}). \tag{2.92}$$

In some cases, there may be some samples that have a performance exactly equal to an intermediate threshold. In this case the ratio estimators may not be equal exactly to the level probability and should be appropriately adjusted and seeds for the MCMC algorithm should be chosen uniformly at random with replacement to ensure that all the levels have the correct level size.

Because of the way the seeds and intermediate densities are selected in SuS, the seeds are always distributed according to the target distribution of the MCMC algorithm. This means that no burn-in phase is required for the MCMC algorithm. This desirable property of SuS is sometimes referred to as *perfect sampling*. It should be noted that the seeds are not in general independent of one another, and so the MCMC estimator used by SuS is not strictly the one that is defined in Section 2.2.3.

Given a random variable X, let the *complementary cumulative density function (CCDF)* be defined as

$$\bar{F}_X(b) := \mathbb{P}(X \geq b). \tag{2.93}$$

Clearly, the probability of failure may be written in terms of a CCDF

$$P_{\mathcal{F}} = \bar{F}_{g(\boldsymbol{X})}(0). \tag{2.94}$$

A run of SuS may be used to estimate $\bar{F}_{g(\boldsymbol{X})}(b)$ for any $b \leq 0$, not just $b = 0$. This is done by appropriately adjusting the estimator in Equation 2.92. Technically it is also possible to use a run of SuS to estimate $\bar{F}_{g(\boldsymbol{X})}(b)$ for $b > 0$ but the estimate would likely be inaccurate. In this work, the primary aim of a reliability method is considered to be the estimation of the probability of failure. However, depending on the context, estimates of these additional values of the CCDF could be useful to a practitioner.

The MH algorithm could be used as the MCMC algorithm for SuS. However, in high dimensions it can become difficult to choose a proposal density that results in a fast mixing

Markov chain. For example, consider a MH algorithm with a standard multivariate normal distribution as its stationary distribution and a normal proposal density. In high dimensions, the density of the stationary distribution is concentrated in the important ring which is not aligned with the shape of the proposal density. Unless the proposal scale is chosen to be very small, the candidates suggested will almost always lie outside the important ring and so will be rejected. In the case that the proposal scale is very small, so that the proposal density lies within the important ring, the Markov chain will take very small steps. In either case, the resulting Markov chain will mix very slowly.

Alternative MCMC algorithms have been suggested [50] that can sample efficiently in high dimensions from densities of the form required by SuS, $f(\boldsymbol{x})\mathbb{1}_{\mathcal{F}_i}(\boldsymbol{x})$. They may be characterised by what assumptions they makes about the input distribution. The modified Metropolis (MM) [42] algorithm assumes that the components of the input distribution are independent of each other. That is, the input distribution may be written as

$$f(\boldsymbol{x}) = \prod_{i=1}^{d} f_i(x_i). \tag{2.95}$$

The MM algorithm requires the practitioner to select a univariate proposal density for each dimension. For simplicity, the same proposal density, denoted by ξ, will be used for all the dimensions. Given the current state $\boldsymbol{x} = (x_1, \ldots, x_d)$, the MM algorithm is given as follows:

Modified Metropolis:

1. Sample a candidate for each dimension,

$$x_i' \sim \xi(\cdot|x_i) \quad \text{for} \quad 1 \leq i \leq d. \tag{2.96}$$

2. Compute the acceptance probability for each dimension using Equation 2.40,

$$A_i := A(x_i', x_i) \quad \text{for} \quad 1 \leq i \leq d. \tag{2.97}$$

3. Generate $u_1, \ldots, u_d \overset{\text{iid}}{\sim} \mathcal{U}[0,1]$.

4. Define a joint candidate as $\boldsymbol{x}' := (x_1', \ldots, x_d')$ where

$$x_i' \leftarrow \begin{cases} x_i' & \text{if} \quad u_i \leq A_i, \\ x_i & \text{otherwise,} \end{cases} \quad \text{for} \quad 1 \leq i \leq d. \tag{2.98}$$

5. Accept or reject the joint candidate,

$$\boldsymbol{x}' \leftarrow \begin{cases} \boldsymbol{x}' & \boldsymbol{x}' \in \mathcal{F}_i \\ \boldsymbol{x} & \text{otherwise.} \end{cases} \tag{2.99}$$

6. Return $\boldsymbol{x}'$

Another alternative MCMC algorithm for SuS is the pre-conditioned Crank Nicolson (pCN) algorithm [51], which in this context, assumes that the reliability problem is in SNS. Given a current state $\boldsymbol{x}$, the pCN algorithm is given as follows:

Pre-conditioned Crank Nicolson:

1. Sample $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}_d, \boldsymbol{I}_d)$.
2. Define a candidate as
$$\boldsymbol{x}' := \left(\sqrt{1-\sigma_{\mathrm{p}}^2}\right)\boldsymbol{x} + \sigma_{\mathrm{p}}\boldsymbol{z}. \tag{2.100}$$
3. Accept or reject the joint candidate,
$$\boldsymbol{x}' \leftarrow \begin{cases} \boldsymbol{x}' & \boldsymbol{x}' \in \mathcal{F}_i \\ \boldsymbol{x} & \text{otherwise.} \end{cases} \tag{2.101}$$
4. Return $\boldsymbol{x}'$.

2.3.2 Sequential importance sampling

Sequential importance sampling (SIS) [52] is another ratio method that uses a different sequence of intermediate densities than SuS. The intermediate densities SIS uses have the following unnormalised form

$$q_i^{\mathrm{SIS}}(\boldsymbol{x}) := f(\boldsymbol{x})\Phi\left(g(\boldsymbol{x})/\sigma_i\right), \tag{2.102}$$

where $\sigma_i > 0$ are *intermediate scale parameters* for $1 \leq i \leq m-1$. The initial density, q_0^{SIS}, is set to the input density and at the end of the algorithm the final density, q_m^{SIS}, is set to the failure density.

The SIS algorithm generates a random sequence of levels in the same way the SuS algorithm does, and in general is very similar to the SuS algorithm. The control flows of the two algorithms are essentially identical. The motivation behind SIS is that the intermediate densities have in some sense a more natural shape which could potentially be easier for MCMC algorithms to traverse. This is because the intermediate densities SuS uses are defined using an indicator function, which results in very sudden changes in density, whereas the intermediate densities SIS uses are more smooth.

One of the drawbacks of SIS compared to SuS is that SIS requires a resampling step. In SuS, the chosen seeds have the required target distribution of the next level, resulting in the perfect sampling property. This is not possible in SIS. Instead, on level i, the seeds are selected by resampling with replacement from the level according to probabilities proportional to the weights $q_{i+1}^{\mathrm{SIS}}(\boldsymbol{x})/q_i^{\mathrm{SIS}}(\boldsymbol{x})$. This is done so that the resampled seeds have approximately the target distribution. However, typically a burn-in phase is still required for the Markov chains, the length of which is controlled by a user-defined parameter denoted as n_{burn}.

The SIS algorithm has some of the same user-defined parameters as SuS. In particular, it uses p_{L}, N_{L}, n_{limit} and also the same definitions for n_c and n_s. The SIS algorithm is given by the following steps:

Sequential importance sampling:

1. Generate N_{L} samples from the input distribution. These samples are collectively referred to as the level 0. Set $i = 0$

2. Estimate the failure CoV using samples form level i,

$$\delta_{\mathcal{F}}^{\mathrm{SIS}} \leftarrow \hat{\delta}_{q_i^{\mathrm{SIS}}} \left[\frac{f(\boldsymbol{x}) \mathbb{1}_{\mathcal{F}}(\boldsymbol{x})}{q_i^{\mathrm{SIS}}(\boldsymbol{x})} \right]. \tag{2.103}$$

If $\delta_{\mathcal{F}}^{\mathrm{SIS}} < \delta_{\mathrm{target}}$, or $i = n_{\mathrm{limit}}$, then terminate the algorithm, where δ_{target} is the user-defined *target CoV*.

3. Solve the following optimisation problem using samples from level i to estimate the CoV,

$$\sigma_{i+1} := \underset{\sigma_{i+1} \in (0, \sigma_i)}{\operatorname{argmin}} \left| \hat{\delta}_{q_i^{\mathrm{SIS}}} \left[\frac{q_{i+1}^{\mathrm{SIS}}(\boldsymbol{x})}{q_i^{\mathrm{SIS}}(\boldsymbol{x})} \right] - \delta_{\mathrm{target}} \right|. \tag{2.104}$$

4. Resample with replacement n_c seeds from level i according to probabilities proportional to the weights $q_{i+1}^{\mathrm{SIS}}(\boldsymbol{x})/q_i^{\mathrm{SIS}}(\boldsymbol{x})$.

5. Run a MCMC algorithm with stationary distribution q_i^{SIS} to generate n_c chains of length $n_s = p_{\mathrm{L}}^{-1}$ using seeds from previous step. A burn-in phase is used for each chain with n_{burn} samples. Update $i \leftarrow i + 1$. Refer to the collective Markov chains samples as level $i + 1$. Go to Step 2.

The SIS estimator for the probability of failure is just the product of ratios estimator with respect to the SIS intermediate densities. It does not have a simplified from like the SuS estimator. A new MCMC algorithm is required in order to sample from the SIS intermediate densities. Given a current state $\boldsymbol{x}$ and a proposal scale, the conditional Metropolis-Hastings (cMH) [52] is given as follows:

Conditional Metropolis-Hastings:

1. Sample $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}_d, \boldsymbol{I}_d)$ and define candidate $\boldsymbol{x}'$ using Equation 2.100.

2. Compute the following acceptance ratio

$$A^{\mathrm{cMH}}(\boldsymbol{x}', \boldsymbol{x}) := \min \left(1, \frac{\Phi\left(g(\boldsymbol{x}')/\sigma_i\right)}{\Phi\left(g(\boldsymbol{x})/\sigma_i\right)} \right). \tag{2.105}$$

3. Generate a uniform sample, $u \sim \mathcal{U}[0, 1]$.

4. Accept or reject the candidate,

$$\boldsymbol{x}' \leftarrow \begin{cases} \boldsymbol{x}' & u \leq A^{\mathrm{cMH}}(\boldsymbol{x}', \boldsymbol{x}), \\ \boldsymbol{x} & \text{otherwise.} \end{cases} \tag{2.106}$$

5. Return $\boldsymbol{x}'$.

2.3.3 Improved cross entropy

One common way of measuring the distance between two densities is to use the *Kullback–Leibler divergence*, also known as the *relative cross entropy (CE)*, defined as

$$D_{\mathrm{KL}}(p(\boldsymbol{x}), q(\boldsymbol{x})) := \mathbb{E}_p \left[\ln \left(\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} \right) \right]. \tag{2.107}$$

One strategy for choosing an importance density for importance sampling is to minimise the CE between a parametric family of densities, denoted by $h(\boldsymbol{x}; \boldsymbol{\nu})$ where $\boldsymbol{\nu}$ is a set of parameters, and the optimal importance density. Recall that in the context of estimating the probability of failure, the optimal importance density is the failure density. Formally, the following optimisation problem is considered,

$$\boldsymbol{\nu}^* = \underset{\boldsymbol{\nu}}{\operatorname{argmin}}\, D_{\mathrm{KL}}(f_F(\boldsymbol{x}), h(\boldsymbol{x}; \boldsymbol{\nu})), \tag{2.108}$$

$$= \underset{\boldsymbol{\nu}}{\operatorname{argmin}}\, \mathbb{E}_{f_F} \left[\ln \left(\frac{f_F(\boldsymbol{x})}{h(\boldsymbol{x}; \boldsymbol{\nu})} \right) \right], \tag{2.109}$$

$$= \underset{\boldsymbol{\nu}}{\operatorname{argmax}}\, \mathbb{E}_{f_F} \left[\ln(h(\boldsymbol{x}; \boldsymbol{\nu})) \right]. \tag{2.110}$$

The final expression does not involve $f_F(x)$ inside the expectation since it does not depend on $\boldsymbol{\nu}$.

In general it is not possible to evaluate the expectation in Equation 2.110 and so it is estimated with either an MC or MCMC estimator depending on the context. This results in the following approximate optimisation problem,

$$\hat{\boldsymbol{\nu}}^* = \underset{\boldsymbol{\nu}}{\operatorname{argmax}}\, \hat{\mu}_{f_F} \left[\ln(h(\boldsymbol{x}; \boldsymbol{\nu})) \right]. \tag{2.111}$$

Updating rules for solving the optimisation problem, dependent on the parametric family chosen, can be typically derived by setting the gradient of the objective function equal to zero. This approach requires samples from the failure density which are typically generated by some initial sampling procedure. A recent example of a reliability method that follows this approach is shown in [53].

Another strategy is to employ a sequence of importance densities that gradual become better approximations of the failure density. This is a similar idea to intermediate densities, however in this case, the normalisation constant of the densities are known. An advantage of this approach is that samples from the failure density are not required, instead, samples from the previous importance density may be use to fit the parameters of the next importance density. That is, given an initial set of parameters denoted by $\boldsymbol{\nu}_0$, Equation 2.110 may be manipulated to produce the following sequence of optimisation problems,

$$\boldsymbol{\nu}_i = \underset{\boldsymbol{\nu}}{\operatorname{argmax}}\, \mathbb{E}_{h_{\boldsymbol{\nu}_{i-1}}} \left[\ln(h(\boldsymbol{x}; \boldsymbol{\nu})) \frac{f_{\mathcal{F}}(\boldsymbol{x})}{h(\boldsymbol{x}; \boldsymbol{\nu}_{i-1})} \right], \tag{2.112}$$

for $1 \leq i \leq m-1$ where m is the number of importance densities. The corresponding approximate optimisation problems are given by

$$\hat{\boldsymbol{\nu}}_i = \underset{\boldsymbol{\nu}}{\operatorname{argmax}}\, \hat{\mu}_{h_{\hat{\boldsymbol{\nu}}_{i-1}}} \left[\ln(h(\boldsymbol{x}; \boldsymbol{\nu})) \frac{f_{\mathcal{F}}(\boldsymbol{x})}{h(\boldsymbol{x}; \hat{\boldsymbol{\nu}}_{i-1})} \right], \tag{2.113}$$

for $1 \leq i \leq m-1$. The final density in the sequence, $h(\boldsymbol{x}; \hat{\boldsymbol{\nu}}_{m-1})$, is used for an importance sampling estimator. Typically some initial sampling procedure is required to ensure that the first importance density is able to produce at least a small amount of failure samples. An example of this type of approach is shown in [54].

A variant of this sequence approach is to introduce an additional sequence of intermediate densities, denoted by q_i, such as those used by SuS or SIS. That is, for the ith optimisation problem defined by Equation 2.113, replace the $f_{\mathcal{F}}$ with q_i for $1 \leq i \leq m-1$. The resulting sequence of optimisation functions may used to fit the required parameters. The advantage of such an approach is that it obviates the need for an initial sampling procedure since the q_i may be chosen adaptively. There have been reliability methods of this type that use q_i^{SuS} as the intermediate sequence [55, 56] and more recently and algorithm called improved cross entropy (iCE) has been suggested which uses q_i^{SIS} instead [57]. The iCE procedure requires some of the same user-defined parameters as SIS. In particular, it requires N_{L}, n_{limit} and δ_{target}. Given an initial set of parameters, $\hat{\boldsymbol{\nu}}_0$, the iCE procedure is given by the following steps.

Improved cross entropy:

1. Set $i = 0$.
2. Generate N_{L} samples from $h(\boldsymbol{x}; \hat{\nu}_i)$ and call the resulting samples level i.
3. Estimate the failure CoV $\delta_{\mathcal{F}}^{\text{SIS}}$ using samples form level i,
$$\delta_{\mathcal{F}}^{\text{SIS}} \leftarrow \hat{\delta}_{h_{\boldsymbol{\nu}_i}} \left[\frac{f(\boldsymbol{x}) \mathbb{1}_{\mathcal{F}}(\boldsymbol{x})}{q_i^{\text{SIS}}(\boldsymbol{x})} \right]. \tag{2.114}$$
If $\delta_{\mathcal{F}}^{\text{SIS}} < \delta_{\text{target}}$, or $i = n_{\text{limit}}$, then terminate the algorithm.
4. Solve the following optimisation problem using samples from level i to estimate the CoV,
$$\sigma_{i+1} = \underset{\sigma_{i+1} \in (0, \sigma_i)}{\operatorname{argmin}} \left| \hat{\delta}_{h_{\hat{\nu}_i}} \left[\frac{q_{i+1}^{\text{SIS}}(\boldsymbol{x})}{h(\boldsymbol{x}; \hat{\boldsymbol{\nu}}_i)} \right] - \delta_{\text{target}} \right|. \tag{2.115}$$
5. Fit $\hat{\boldsymbol{\nu}}_{i+1}$ using the samples of level i by maximising the following objective function,
$$\hat{\boldsymbol{\nu}}_{i+1} = \underset{\boldsymbol{\nu}}{\operatorname{argmax}}\, \hat{\mu}_{h_{\hat{\nu}_i}} \left[\ln(h(\boldsymbol{x}; \boldsymbol{\nu})) \frac{q_{i+1}^{\text{SIS}}(\boldsymbol{x})}{h(\boldsymbol{x}; \hat{\boldsymbol{\nu}}_i)} \right]. \tag{2.116}$$
Update $i \leftarrow i + 1$. Go to step 2.

The final *iCE estimator* for the probability of failure is then just a IS estimator using the samples from the final level, that is level $m - 1$,
$$P_{\mathcal{F}} \approx \hat{P}_{\mathcal{F}}^{\text{iCE}} := \hat{\mu}_{h_{\boldsymbol{\nu}_{m-1}}} \left[\frac{f(\boldsymbol{x}) \mathbb{1}_{\mathcal{F}}(\boldsymbol{x})}{h(\boldsymbol{x}; \hat{\boldsymbol{\nu}}_{m-1})} \right]. \tag{2.117}$$
Note that this is typically a MC estimator since it is typically possible to generate independent samples from the importance density. Any parametric family may be used in the iCE procedure, so long as it is possible to efficiently solve the optimisation problems defined in Equation 2.116. The specific parametric family that is suggested in [57], and the one that is used in the numerical experiments in this work, will be described in Section 5.1.1. For now it will suffice to mention that a mixture distribution is used, and the number of components in that mixture distribution is controlled by a user parameter called the *initial components* and is denoted by k_{init}. Note that iCE is both a model method and an intermediate method, but it is not a ratio method.

3 Multimodality

The most important property of the CRP is that it exhibits some form of multimodality. This is because methods like SuS, SIS and iCE tend to have very few issues with the type of reliability problem that results from removing this property from the CRP. Up until this point, the term multimodality has been used informally. Section 3.1 formally defines some variants of multimodality that are relevant in the reliability analysis context and that cause issues for existing reliability methods. Section 3.2 introduces a set benchmark CRP and explores the behaviour of existing reliability methods on those benchmarks. Finally, Section 3.3 discusses how similar problems of multimodality are tackled in the in the field of black-box optimisation and how similar ideas could be applied to reliability analysis.

3.1 Multimodal reliability analysis

Multimodality may be exhibited by a reliability problem in many different forms. The simplest example is a reliability problem with a multimodal failure density. This section will formally define other types of multimodality that will be useful for characterising reliability problems and understanding the behaviour of reliability methods.

3.1.1 Metastable multimodal

Let $\tau(\cdot|\cdot)$ be a transition density of a Markov chain with stationary distribution p. The *retention probability* of a set $\mathcal{M} \subset \mathbb{R}^d$ is given as

$$r(\mathcal{M}) := \frac{1}{\int_{\mathcal{M}} p(\boldsymbol{x}) d\boldsymbol{x}} \int_{\mathcal{M}} \int_{\mathcal{M}} \tau(\boldsymbol{y}|\boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x} d\boldsymbol{y}. \tag{3.1}$$

When the retention probability of a set is high, say

$$r(\mathcal{M}) > 1 - \varepsilon, \tag{3.2}$$

where $\varepsilon > 0$ is a small constant, then the set will be referred to as a *metastable region (MSR)*. For convenience, the dependency of a MSR on ε will be suppressed. It may be assumed that ε has been chosen such that once a Markov chain enters a MSR it is extremely unlikely to leave within a reasonable number of steps, where what is reasonable depends on the context.

If multiple disjoint MSRs exist for a Markov chain, then the Markov chain will be referred to as *metastable multimodal*. A MSR will be referred to as *high density* if it has a high probability with respect to the stationary distribution. Again, the meaning of high probability will depend on the context. A *low density* MSR is similarly defined. If

a Markov chain is metastable multimodal with two high density MSRs, then it is highly likely to experience ergodicty issues. This is because it will become trapped in one of the MSRs and not be able to sample from the other, resulting in an empirical distribution that is a poor approximation of the stationary distribution. If instead the Markov chain has one high and one low density MSR, it may not experience ergodicity issues. This is because if the chain becomes trapped in the high density MSR, its empirical distribution may be able to approximate the stationary distribution quite well since samples from the low density MSR are relatively unimportant.

An example of a metastable multimodal Markov chain algorithm is now studied. Consider the one dimensional *symmetric performance function* parametrised by some constant $a > 0$,

$$g_{\text{sym}}(x) := |x| - a. \tag{3.3}$$

Let $f_{\mathcal{F}}$ be the failure density of a reliability problem in SNS using the symmetric performance function with $a = 2$. This failure density is shown in Figure 3.1a alongside some other densities. As can be seen it has two modes at $x = -2$ and $x = 2$. Each mode has a corresponding high density set of that it is contained within, that is,

$$\mathcal{M}^- := \{x \in \mathbb{R} : x \leq -2\}, \tag{3.4}$$

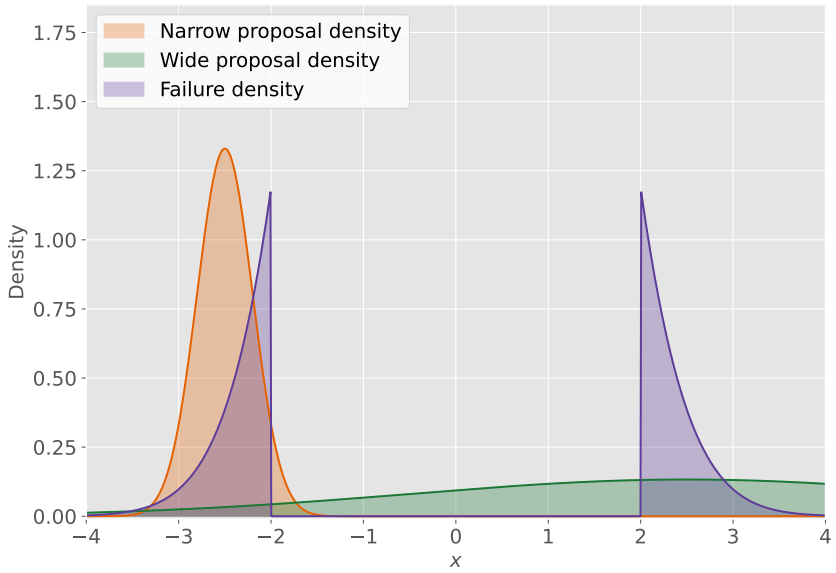
$$\mathcal{M}^+ := \{x \in \mathbb{R} : x \geq 2\}, \tag{3.5}$$

respectively. These two high density sets are separated by a region of zero density.

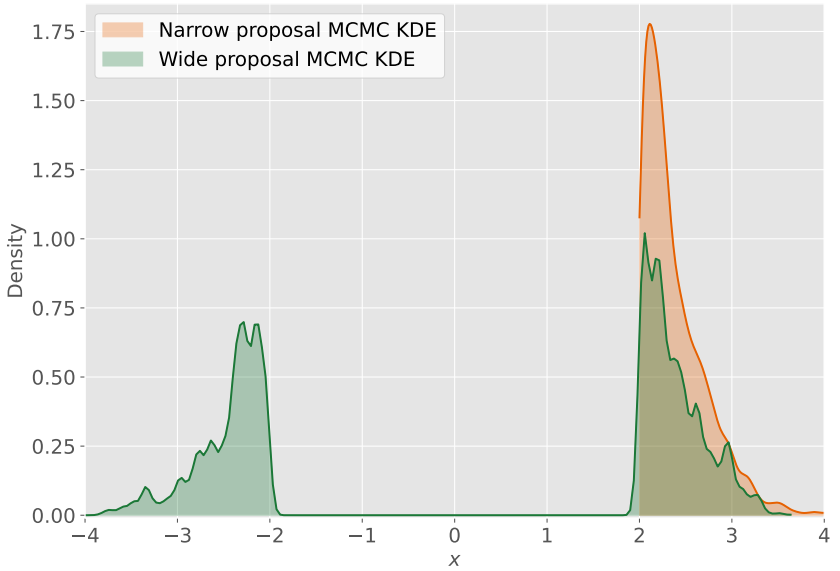
The MH algorithm can be used to sample from this failure density. Figure 3.1a also shows two potential proposal densities that could be used. Both are normal proposal densities, but one is narrower and one is wider with proposal scales of 0.3 and 3 respectively. The mean of these proposal densities depend on the current state of the Markov chain. Figure 3.1a shows the narrow and wide proposal densities with centres in $\mathcal{M}^-$ and $\mathcal{M}^+$ respectively. Note that in this case the narrow proposal assigns essentially zero density to $\mathcal{M}^+$ whereas the wide proposal density assigns a small amount of the density to $\mathcal{M}^-$. As a result, $\mathcal{M}^-$ and $\mathcal{M}^+$ are MSRs with respect to Markov chain with the narrow proposal density, but are not MSRs with respect to Markov chain with the wide proposal density.

Two MH Markov chains of length 5000 were generated with the failure density as their stationary distributions. One used the narrow proposal density and one used the wide proposal density. Both used the seed $x = 2.5$. Next, kernel density estimation (KDE) was used to estimate the empirical PDFs of the resulting samples. Figure 3.1b shows the KDE of the empirical PDFs of the resulting samples. As expected, the narrow proposal density is not able to jump between high density sets and so the resulting empirical distribution is unimodal, whereas the wide proposal density is able to jump between high density sets and so is able to capture the bi-modality of the failure density. It should be noted that the Markov chain with the narrow proposal density is still technically ergodic, since given enough time it would eventually make a jump between the MSRs. It is just extremely slowly mixing. Also note that ergodicity issues are not a property of a density, but of a MCMC algorithm with respect to a density. A density can be multimodal, but not metastable multimodal with respect to a MCMC algorithm, as has been demonstrated by the wide proposal density.

Whilst the wide proposal density is preferable in this context, it is still far from ideal. The wide proposal density will suggest many candidates in the region of zero density,



(a) Failure density of the symmetric performance function alongside candidate proposal densities.



(b) KDEs of the empirical distributions of two MCMC algorithms

which are guaranteed to be rejected, resulting in a low acceptance rate. The shape of the normal distribution is fundamentally different to the failure density and so for any choice of proposal scale, the resulting MCMC algorithm will be inefficient at best or have serious ergodic issues at worst. The optimal choice for a proposal density would be the failure density itself, independent of the current state of the Markov chain. Of course, this is not practical, as it is required that the proposal density can be sampled from efficiently. If this were the case for the failure density, the MCMC algorithm would not be required in the first place. An alternative strategy could be to use some bimodal proposal density that can be sampled from, such as a mixture of two normal distributions. However, it is not in general easy to decide on the parameters of such a proposal density.

Another approach to sampling from the failure density efficiently is to use an ensemble of Markov chains, rather than just one. In particular, start one chain in each MSR with the narrow proposal density and use their combined samples to construct estimators. Notice that the narrow proposal density matches the shape of both individual high density sets quite well, and so this process will be able to efficiently generate samples from the failure density. Each individual chain still experiences ergodic issues, but the ensemble does not. This idea works for this specific problem, since the MSRs have equal probability under the stationary distribution. However, if one MSR had a much higher density that the other, this method would result in a poor approximation of the failure density. It follows that in order for this ensemble strategy to work, the number of chains in each MSR must be proportional to the relative probabilities of the MSRs.

When the target density is a black box, as is the case for the failure density in a CRP, there is no way of knowing a priori whether a MCMC algorithm will be metastable multimodal. In particular there is no way of knowing how many MSRs there are, their location, or their relative probabilities. Without this information it is difficult to design an efficient MCMC algorithm. As a result, a common strategy is to employ an initial sampling procedure which is used to deduce such information, which can then be used inform the MCMC algorithm. This is similar to the idea of using the burn-in phase of a Markov chain algorithm to tune the parameters of a proposal density.

Intermediate methods that use Markov chains, such as SuS and SIS, take a different approach to handling ergodicity issues that arise from metastable multimodality when sampling from the intermediate densities they employ. The first intermediate density is sampled from using simulation methods that produce independent samples, and so there are no ergodicity issues. The next intermediate density is sampled from using a large ensemble of Markov chains. As discussed previously, this avoids ergodicity issues as long as each MSR is assigned the correct number of chains. This should be the case, since the seeds for the chains has been sampled from (approximately in the case of SIS) the target density itself. This logic may then be applied iteratively to all subsequent levels. However, as will be explored in the next section, this strategy does have issues. In particular, when an intermediate method uses a long sequence of metastable multimodal intermediate densities, small sampling errors in the early levels can propagate through to later levels, ultimately resulting in a poor estimate for the probability of failure.

3.1.2 Basin multimodal

Since the inception of SuS it has been known that multimodal intermediate densities can cause problems for intermediate methods [42] and such deficiencies have also been discussed in later work [19, 32]. This section introduces some novel definitions that are

useful for characterising the types of reliability problems for which it is difficult to estimate the probability of failure with an intermediate method. These definitions could be thought of as the analogue of metastable multimodal, with respect to a sequence of intermediate densities rather than a MCMC algorithm.

Suppose that an intermediate method is ran on a reliability problem, resulting in a sequence of adaptively chosen intermediate densities $q_0, \ldots, q_m$. Such a sequence is referred to as *intermediate multimodal* if there exists a partition of the input space $\mathcal{B}_1, \ldots, \mathcal{B}_K$ with $K > 1$, an index i^* with $0 \leq i^* \leq m$ and a set of regions denoted as $\mathcal{M}_j^{(i)}$ for $i^* \leq i \leq m$ and $1 \leq j \leq K$ with the following properties:

- $\mathcal{M}_j^{(i)} \subset \mathcal{B}_j$ for $i^* \leq i \leq m$ and $1 \leq j \leq K$.
- $\mathbb{P}_{q_i}(\mathcal{M}_j^{(i)})/\mathbb{P}_{q_i}(\mathcal{B}_j) \approx 1$ for $i^* \leq i \leq m$ and $1 \leq j \leq K$.

When an intermediate sequence is intermediate multimodal, it can lead to the empirical distributions of each level being poor approximations of the target distribution, which ultimately could result in an inaccurate estimate for the probability of failure. To understand why, intermediate methods that use a MCMC algorithm like SuS and SIS will be considered, though similar arguments hold for intermediate methods such as iCE. For $i \geq i^*$, almost all of the density of q_i is contained in the sets $\mathcal{M}_1^{(i)}, \ldots, \mathcal{M}_K^{(i)}$ which are high density sets surrounded by regions of low density. Thus it is reasonable to assume that these sets are MSRs, though of course the validity of this assumption depends on the specific MCMC algorithm being employed. It follows that, starting at level i^*, single chains of the MCMC algorithm will suffer from ergodicity issues because they will never make a step from $\mathcal{B}_j$ to $\mathcal{B}_{j'}$ where $j \neq j'$.

Methods like SuS and SIS generate a level by first resampling seeds from the previous level according to the relevant importance weights, and then by using those seeds to generate chains with a MCMC algorithm. In SuS the resampling step is done implicitly due to all the importance weights being either 0 or 1. The number of seeds selected for constructing the ith level that lie in $\mathcal{B}_j$, denoted as S_j^i, depends on the number of samples from the previous level that lie in $\mathcal{B}_j$. In particular, these random variables have positive covariance,

$$\operatorname{Cov}\left(|L_{i-1} \cap \mathcal{B}_j|, S_j^{(i)}\right) > 0, \tag{3.6}$$

for $1 \leq i \leq m$ and $1 \leq j \leq K$. Assume that for the levels prior to level i^*, the MCMC algorithm mixes very fast, such that the samples of these levels are effectively generated independently. In this case, the dependence of the number of seeds in each region of the input space on the previous level has a minimal effect on the number of samples in each region of the input space for the current level,

$$\operatorname{Cov}\left(|L_{i-1} \cap \mathcal{B}_j|, |L_i \cap \mathcal{B}_j|\right) \approx 0, \tag{3.7}$$

for $1 \leq i < i^*$ and $1 \leq j \leq K$. However, for level i^* and later levels, the previously discussed ergodicity issues mean that the number of seeds in each region of the input space deterministically sets the number of samples in each region of the input space. In particular,

$$|L_i \cap \mathcal{B}_j| = S_j^{(i)} \cdot n_s, \tag{3.8}$$

for $i^* \leq i \leq m$ and $1 \leq j \leq K$. It follows that

$$\operatorname{Cov}\left(|L_{i-1} \cap \mathcal{B}_j|, |L_i \cap \mathcal{B}_j|\right) > 0, \tag{3.9}$$

for $i^* \leq i \leq m$ and $1 \leq j \leq K$.

Because of the positive covariance between levels, if $\mathcal{B}_j$ is undersampled at one level, it will tend to be undersampled at the next level and vice versa for oversampling. Applying this reasoning iteratively to the whole sequence of subsequent levels implies that an initial random fluctuation can accumulate over successive resampling steps resulting in an empirical distribution that is a poor approximation of the target density. The most dramatic variant of this phenomena is the case where 0 seeds are selected in $\mathcal{B}_j$. In this case, it is guaranteed that $\mathcal{B}_j$ will not become populated by any future levels, regardless of how large its probability is with respect to the later intermediate densities.

Intermediate multimodal is a property of a particular intermediate sequence. It would be useful if it was possible to identify reliability problems that are likely to result in an intermediate multimodal sequence, when an intermediate methods is applied to them. Additionally, given an intermediate sequence, it may not be straight-forward to prove whether it is or is not intermediate multimodal. The following definition is a property of a reliability problem and it offers a concrete way of constructing a partition of the input space that can often be used to used to locate the regions required in the definition of an intermediate multimodal sequence. This is done using a dynamical system that attempts to model how a parametric family of intermediate densities, such as q_i^{SuS} or q_i^{SIS}, evolves from level to level of an intermediate method in SNS.

One approach to describing such a dynamical system is to use a differential equation that depends on the gradient of the performance function. However, even if it assumed that the gradient of the performance function is well-defined, regularity assumptions would be required to ensure that the associated initial value problems admit unique solutions. To circumvent these difficulties, a discrete time dynamical system defined on a finite grid is considered instead. Given a large $a > 0$ and a small $\varepsilon > 0$, define the *univariate grid* as

$$G := \{-a, -a + \varepsilon, -a + 2\varepsilon, \ldots, a - \varepsilon, a\}, \tag{3.10}$$

and the *grid* as

$$\mathcal{G} := \{(x_1, \ldots, x_d) \in \mathbb{R}^d : x_i \in G \text{ for } 1 \leq i \leq d\}. \tag{3.11}$$

At each step, the dynamical system will require a notion of points that are close to the current point that it is able to step to. Define the *grid neighbourhood* of a point $\boldsymbol{x} \in \mathcal{G}$ as

$$\mathrm{N}_{\mathcal{G}}(\boldsymbol{x}) := \{\boldsymbol{y} \in \mathcal{G} : \|\boldsymbol{x} - \boldsymbol{y}\| \leq r\}. \tag{3.12}$$

for $r > 0$.

Both the sequences q_i^{SuS} and q_i^{SIS} assign more density to points with higher performances as an intermediate method progresses. Additionally, in SNS, both methods assign more density to points with a smaller Euclidean norm. The following *update map* is a simple way of modelling both of these properties,

$$\Psi(\boldsymbol{x}) := \begin{cases} \operatorname{argmax}_{\boldsymbol{y} \in \mathrm{N}_{\mathcal{G}}(\boldsymbol{x})} g(\boldsymbol{y}) & g(\boldsymbol{x}) < 0, \\ \operatorname{argmin}_{\{\boldsymbol{y} \in \mathrm{N}_{\mathcal{G}}(\boldsymbol{x}) : g(\boldsymbol{y}) \geq 0\}} \|\boldsymbol{y}\| & g(\boldsymbol{x}) \geq 0. \end{cases} \tag{3.13}$$

Note that there should be some ordering defined on the grid points in order to decide cases of ties. Let the set of *attractors* of this update map be given as

$$\mathcal{A} := \{\boldsymbol{x} \in \mathcal{G} : \Psi(\boldsymbol{x}) = \boldsymbol{x}\}. \tag{3.14}$$

It is clear from the definition of the update map that there will typically be an attractor associated with each local maxima of the performance function in the safe region and with each mode of the failure density in the failure region.

Starting at some initial point in the grid, if the update map is repeatedly applied, then the eventual result will be an attractor. This is because the update map permits no cycles. Define the *repeated update map* as

$$\Psi_*(\boldsymbol{x}) := \begin{cases} \boldsymbol{x} & \boldsymbol{x} \in \mathcal{A} \\ \Psi_*(\Psi(\boldsymbol{x})) & \text{otherwise.} \end{cases} \tag{3.15}$$

So that the update map may be applied to any point $\boldsymbol{x} \in \mathbb{R}^d$, define the *nearest grid neighbour* as

$$\mathrm{NN}_{\mathcal{G}}(\boldsymbol{x}) := \operatorname*{argmin}_{\boldsymbol{y} \in \mathcal{G}} \|\boldsymbol{x} - \boldsymbol{y}\|. \tag{3.16}$$

Now label the members of $\mathcal{A}$ as $\boldsymbol{x}_1^{\mathcal{A}}, \ldots, \boldsymbol{x}_K^{\mathcal{A}}$. Finally a *basin of attraction (BoA)* may be defined,

$$\mathcal{B}_i = \{\boldsymbol{x} \in \mathbb{R}^d : \Psi_*(\mathrm{NN}_{\mathcal{G}}(\boldsymbol{x})) = \boldsymbol{x}_i^{\mathcal{A}}\}, \tag{3.17}$$

for $1 \leq i \leq K$. Note that $\mathcal{B}_1, \ldots, \mathcal{B}_K$ is a partition of the input space. When $K > 1$, the reliability problem will be referred to as *basin multimodal*. This definition depends on the constants a, ε and r. It may be assumed that $a = 10$, $\varepsilon = 0.01$ and $r = 0.01$ and so this dependence will be suppressed from now on. If the attractor of a BoA is a failure sample then the BoA will be referred to as a *failure BoA*. If the attractor of a BoA is a safe sample then the BoA will be referred to as a *safe BoA*.

Given the level of an intermediate method, the update map may be used to very roughly predict what the next level will look like for a low dimensional reliability problem. In particular, this can be done by applying the update map some number of times to each sample. Note that if update map is applied enough times the resulting samples will become concentrated in the neighbourhood of each attractor. If this is indeed an accurate model of the intermediate densities, then this would imply that eventually the intermediate densities will be multimodal with a mode corresponding to each attractor. This in turn would imply that the BoAs can be used to prove that the intermediate densities are intermediate multimodal. For this reason, a useful heuristic is that if a reliability problem is basin multimodal, then any intermediate method is likely to adaptively choose an intermediate multimodal sequence of intermediate densities on any particular run.

The focus here has been on intermediate methods, but basin multimodal reliability problems will be challenging for many other reliability methods as well. This is because reliability methods that do not use a sequence of intermediate densities tend to rely on an initial sampling scheme or optimisation algorithm to identify the location of the failure region. These types of procedures will tend to also struggle when there are multiple BoAs.

Consider a reliability problem in SNS that uses the one-dimensional symmetric performance with $a = 5$. The BoAs of this reliability problem are given as

$$\mathcal{B}_1 := \{x \in \mathbb{R} : x < 0\}, \tag{3.18}$$

$$\mathcal{B}_2 := \{x \in \mathbb{R} : x \geq 0\}. \tag{3.19}$$

In the case of SuS, the intermediate densities will have the same shape as the failure density studied in Section 3.1.1. That is, for later levels, two high density sets separated

by a region of zero density, where each of $\mathcal{B}_1$ and $\mathcal{B}_2$ contains one of the high density sets. This implies that any sequence of intermediate densities adaptively chosen by SuS will be intermediate multimodal.

The following numerical experiment was carried out on this reliability problem. The SuS algorithm, with level size 500, level probability 0.1 and using a pCN MCMC algorithm with the proposal density set to 0.5 was ran 500 times, resulting in 500 estimates for the probability of failure. In order to evaluate the quality of the estimates, the true probability of failure, also referred to as the reference probability, is required. In this simple case, the probability of failure may be written in terms of the CDF of the normal distribution and thus is easy to approximate accurately,

$$\mathbb{P}(\mathcal{F}) = \mathbb{P}(\mathcal{F} \cap \mathcal{B}_1) + \mathbb{P}(\mathcal{F} \cap B_2) \tag{3.20}$$

$$= 2 \cdot (1 - \Phi(5)) \tag{3.21}$$

$$\approx 5.73 \times 10^{-7}. \tag{3.22}$$

The symmetry of this reliability problem means that each BoA contains exactly half of the density of the failure region. The estimates of the numerical experiment can sorted into two cases depending on how their corresponding SuS run behaved:

1. Both $|L_{m-1} \cap \mathcal{F} \cap \mathcal{B}_1| > 0$ and $|L_{m-1} \cap \mathcal{F} \cap \mathcal{B}_2| > 0$.
2. Either $|L_{m-1} \cap \mathcal{F} \cap \mathcal{B}_1| = 0$ or $|L_{m-1} \cap \mathcal{F} \cap \mathcal{B}_2| = 0$.

It may be expected that those estimates in Case 2 would have a mean that is roughly half of the reference probability, since half of the failure region will contain no samples.

Figure 3.2 shows the empirical distributions of estimates associated with Case 1 and Case 2, approximated using KDE with 54 estimates of each type. As can be seen, the empirical distributions of the two estimators are very similar. In particular, the means of each are good estimates of the probability of failure. Whilst this would be expected of the Case 1 estimates, it may be surprising that the Case 2 estimates are capable of producing an accurate estimate for the probability of failure. This is possible due to the symmetry of this reliability problem. For any intermediate distribution q_i, define an associated mixture distribution,

$$q_i^w(\boldsymbol{x}) \propto w \cdot q_i^{\mathrm{SuS}}(\boldsymbol{x}) \mathbb{1}_{\mathcal{B}_1}(\boldsymbol{x}) + (1 - w) \cdot q_i^{\mathrm{SuS}}(\boldsymbol{x}) \mathbb{1}_{\mathcal{B}_2}(\boldsymbol{x}) \tag{3.23}$$

with $w \in [0, 1]$. Note that for any w and any intermediate failure region $\mathcal{F}'$ it is true that

$$\mathbb{P}_{q_i^{\mathrm{SuS}}}(\mathcal{F}') = \mathbb{P}_{q_i^w}(\mathcal{F}'). \tag{3.24}$$

In particular this statement holds for $w = 0$ and $w = 1$. This means that for all of the ratio estimators that comprise the product of ratios estimator that SuS uses, sampling from only one of the BoAs will still result in an accurate estimate.

It should be noted that whilst SuS is able to give an accurate estimate for the probability of failure in Case 2, the empirical distribution of the resulting failure samples is of course a poor approximation of the failure density since half of the failure region has not been sampled from. As a result, any probabilistic failure analysis that is conducted using these failure samples will likely yield inaccurate results. An intermediate method that does not use a ratio estimator, like iCE, depends on the empirical distribution of the

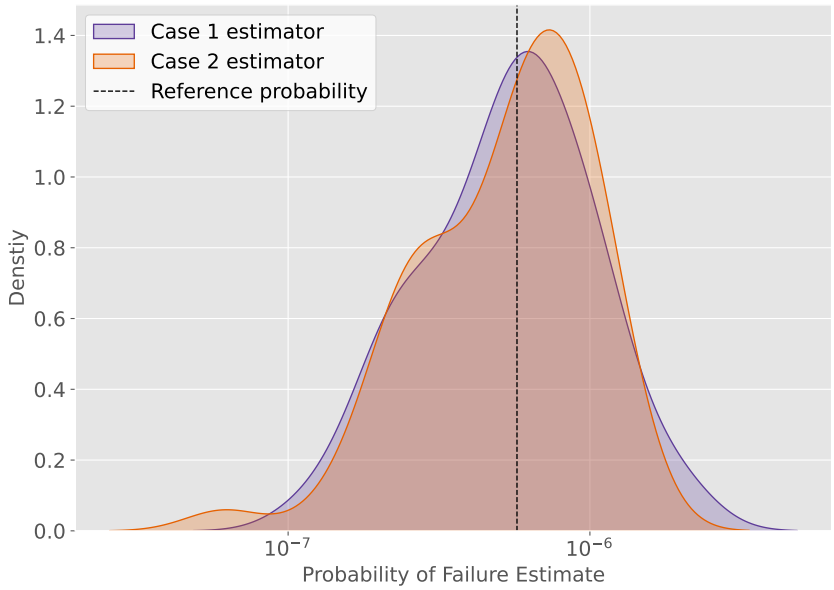


Figure 3.2: KDEs of the empirical distributions of the two cases of SuS estimators for the symmetrical performance function.

failure samples being close to the failure density. This is because the failure samples are used to fit the importance density that is then used for importance sampling. Because of this, in Case 2, iCE would yield an estimator for the probability of failure with a very high variance. In summary SuS is capable of producing an accurate estimate for the probability failure even when it fails to sample from the failure density, whereas iCE is not.

Because of the symmetry of the reliability problem,

$$\mathbb{P}_{q_i^{\mathrm{SuS}}}(\mathcal{B}_1) = \mathbb{P}_{q_i^{\mathrm{SuS}}}(\mathcal{B}_2) = 1/2, \tag{3.25}$$

for any intermediate density q_i^{SuS}. Because of this, it would be reasonable to conjecture that the probability of Case 2 might be essentially negligible. The results of the numerical experiment give the empirical estimate

$$\mathbb{P}(\text{Case 2}) \approx 11\%, \tag{3.26}$$

which contradicts this intuition. To understand why this is the case, the probability of Case 2 will now be computed using several different models. To begin, it will be assumed that each intermediate density is metastable multimodal with respect to the chosen MCMC algorithm and so no step will ever be taken by a chain from $\mathcal{B}_1$ to $\mathcal{B}_2$, or vice versa. It will also be assumed that the distribution of the number of levels is given as

$$\mathbb{P}(m = 7) = 1, \tag{3.27}$$

which is not unreasonable given the empirical data and by inspecting the reference probability of failure. Let the number of seeds chosen at each level that are in $\mathcal{B}_1$ be denoted as

$$S_i = |L_{i-1} \cap \mathcal{F}_i \cap \mathcal{B}_1|. \tag{3.28}$$

Finally, to simplify the following models, it will be assumed that

$$|L_6 \cap B_i| > 0 \implies |L_6 \cap \mathcal{F} \cap B_i| > 0, \tag{3.29}$$

for $i = 1, 2$. With these assumptions, the probability of Case 2 is given as

$$\mathbb{P}(\text{Case 2}) = \sum_{i=1}^{2} \mathbb{P}(|L_6 \cap B_i| = 0), \tag{3.30}$$

$$= 2 \cdot \mathbb{P}(|L_6 \cap \mathcal{B}_1| = 0), \tag{3.31}$$

$$= 2 \cdot \mathbb{P}(S_6 \cdot n_s = 0), \tag{3.32}$$

$$= 2 \cdot \mathbb{P}(S_6 = 0). \tag{3.33}$$

Describing the distribution of S_6 will be difficult, since it depends on the complex dynamics of SuS. However, the exact distribution of S_1 may be described easily,

$$S_1 \sim \text{Binomial}(n_c, 1/2), \tag{3.34}$$

where for this particular experiment $n_c = 50$. This is because on level 0, the samples are i.i.d. according input distribution and because of the symmetry of the reliability problem, each of the 50 seeds has an even chance of being in $\mathcal{B}_1$ or $\mathcal{B}_2$. If it is assumed that on all the levels the samples are i.i.d. according corresponding intermediate density, then the distributions of the rest of the S_i are the same,

$$S_i \sim \text{Binomial}(n_c, 1/2) \tag{3.35}$$

for $2 \leq i \leq 6$. This model would imply that

$$\mathbb{P}(\text{Case 2}) = 2 \cdot \mathbb{P}(S_6 = 0) = 2 \cdot \left(\frac{1}{2}\right)^{50} \approx 1.78 \times 10^{-15}. \tag{3.36}$$

Clearly this value deviates from the observed value significantly, and so the model needs to be adjusted.

In this simple model, the S_i are independent of one another, instead of there being positive covariance as discussed previously. In particular, due to the symmetry of the reliability problem, the model should be adjusted such that the following equality holds

$$\mathbb{E}[S_i | S_{i+1} = s] = s \tag{3.37}$$

for $2 \leq i \leq 6$. Note that this would imply that

$$S_j = 0 \implies S_i = 0 \quad \text{for} \quad i \geq j. \tag{3.38}$$

A simple way of ensuring this property is to use the following model,

$$S_i \sim \text{Binomial}(n_c, S_{i-1}/n_c) \tag{3.39}$$

for $2 \leq i \leq 6$. The following empirical estimate was made using 10^6 runs of this model,

$$\mathbb{P}(\text{Case 2}) = 2 \cdot \mathbb{P}(S_6 = 0) \approx 4.10 \times 10^{-4}. \tag{3.40}$$

Whilst this is much closer to the observed value than the previous estimate, there is still a large discrepancy and so the model needs to be adjusted again.

The previous model assumes that the samples at each level are generated independently, when in reality they are generated using a MCMC algorithm. To account for the resulting correlation, a smaller effective sample size N_{eff} may be used instead of the level size N_{L}. Consider the following model,

$$S_i \sim (N_{\text{L}}/N_{\text{eff}}) \cdot \text{Binomial}(N_{\text{eff}} \cdot p_{\text{L}}, S_{i-1}/n_c). \tag{3.41}$$

Here the number of seeds are chosen from the smaller effective sample size. However, to ensure that Equation 3.37 still holds, the result must be scaled up by a factor of $(N_{\text{L}}/N_{\text{eff}})$. Note that the S_i in this model are no longer necessarily integers, but they still take the value 0 and so the model is suitable for estimating the quantity of interest in this context. With $N_{\text{eff}} = 150$, the following empirical estimate was made using 10^6 runs of this model,

$$\mathbb{P}(\text{Case 2}) = 2 \cdot \mathbb{P}(S_6 = 0) \approx 10\%. \tag{3.42}$$

This estimate is more closely aligned with the observed probability then the previous attempts. In summary, the combination of consecutive metastable multimodal intermediate densities and correlated samples can, over many levels, lead to large deviations from the target distribution.

Now the reliability problem is altered so that it is asymmetrical. Remaining in the SNS, consider the reliability problem defined by the one-dimensional *asymmetrical performance function*,

$$g_{\text{asym}}(x) = \begin{cases} |x| - a & x > -a, \\ -a & \text{otherwise}, \end{cases} \tag{3.43}$$

with parameter $a = 5$. A numerical experiment, with all the same parameters as the previous numerical experiment was ran on this reliability problem. This time, the reference probability is exactly half of what it was on the previous example,

$$\mathbb{P}(\mathcal{F}) = 1 - \Phi(5) \tag{3.44}$$

$$\approx 2.87 \times 10^{-7}. \tag{3.45}$$

This reliability problem has the same BoAs, $\mathcal{B}_1$ and $\mathcal{B}_2$, as the previous example. This time, the estimates are sorted using 3 cases:

1. Both $|L_{m-1} \cap \mathcal{F} \cap \mathcal{B}_1| > 0$ and $|L_{m-1} \cap \mathcal{F} \cap \mathcal{B}_2| > 0$.
2. $|L_{m-1} \cap \mathcal{F} \cap \mathcal{B}_1| = 0$.
3. $|L_{m-1} \cap \mathcal{F} \cap \mathcal{B}_2| = 0$.

Note that the failure region is a subset of $\mathcal{B}_2$, thus any Case 3 estimate will be 0.

The following are the relative proportions of the three cases that were observed in the numerical experiment:

$$\mathbb{P}(\text{Case 1}) = 81\%, \tag{3.46}$$

$$\mathbb{P}(\text{Case 2}) = 13\%, \tag{3.47}$$

$$\mathbb{P}(\text{Case 3}) = 6\%. \tag{3.48}$$

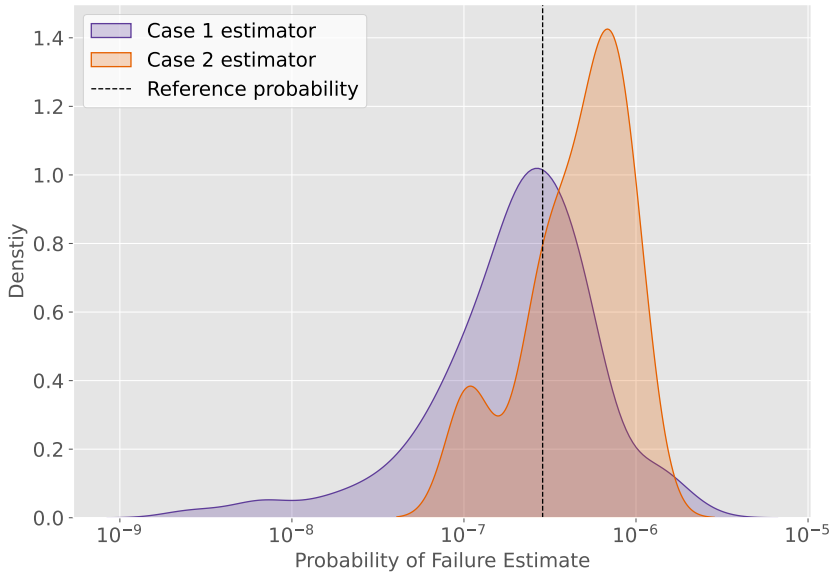


Figure 3.3: KDEs of the empirical distributions of two of the cases of SuS estimators for the asymmetrical performance function.

Figure 3.3 shows the empirical distributions of estimates associated with Case 1 and Case 2, approximated with KDE. It can be seen that the Case 2 estimator tends to overestimate the probability of failure by roughly a factor of 2. This is because SuS essentially assumes that performance function is symmetrical when in this case it is not. Clearly SuS, with these user-defined parameters, is not a suitable method for estimating the probability of failure for this reliability problem. This is because 13% of the time it will yield an estimate that is roughly double the probability of failure and 6% it will yield an estimate of 0.

Note that in Case 2, SuS does sample accurately from the failure density, but is unable to produce a good estimate for the probability of failure. This is an exact reversal of Case 2 in the previous example. In contrast, an intermediate method like iCE only relies on the being able to model the failure density accurately, so in Case 2 it would still yield an accurate estimate for the probability of failure.

3.2 Challenging reliability problems

In order to compare the performance of existing reliability methods against the novel methods that will be suggested in this work, a set of benchmark reliability problems are introduced alongside some evaluation criteria.

3.2.1 Benchmarks

This section introduces seven benchmark CRPs. Most have been taken from other sources and one has been contrived specifically for this work. Most of these reliability problems are low-dimensional, whereas one of the most important properties of a CRP is that it is high-dimensional. In order to account for this, the following procedure is used to raise their dimension. Given a performance function g with input dimension d it is possible to construct a performance function with identical probability of failure and a higher

input dimension d^*. Provided $d^*/d = s$ is an integer, the higher dimensional performance function is defined as

$$g^*(\boldsymbol{x}) = g(z_1, \ldots, z_d), \text{ where } z_i = \frac{1}{\sqrt{s}} \sum_{j=(i-1)s-1}^{i \cdot s} \boldsymbol{x}_j, \quad \text{for } 1 \leq i \leq d. \tag{3.49}$$

Note that all the benchmarks are assumed to be SNS, so each of the z_i will have a standard normal distribution.

The first two benchmarks are taken from [19]. Both have been multiplied by -1 in order to align with the conventions used in this work. The *piecewise linear function* is given as

$$g_{\text{pwl}}(\boldsymbol{x}) = -\min(g_1(x_1), g_2(x_2)), \text{ where}$$

$$g_1(x_1) = \begin{cases} 4 - x_1 & x_1 > 3.5, \\ 0.85 - 0.1x_1 & x_1 \leq 3.5, \end{cases} \tag{3.50}$$

$$g_2(x_2) = \begin{cases} 0.5 - 0.1x_2 & x_2 > 2, \\ 2.3 - x_2 & x_2 \leq 2. \end{cases}$$

A three dimensional surface plot of the piecewise linear function is shown in Figure 3.4a alongside the limit state surface and global design point. Note that there are many instantaneous changes in gradient. The BoAs are shown in Figure 3.4b with some trajectories of the update map. This reliability problem has two failure BoAs

Figures 3.4c and 3.4d show the result of a SuS run on the piecewise linear function. Figure 3.4c shows the state of SuS after generating and early level with intermediate threshold set as -0.28. The associated intermediate density has two MSRs, one high density and one low density. As expected SuS correctly populates the high density MSR and does not generate any samples in the low density MSR. Since at this point all the samples are confined to one BoA, all future levels are also confined to this BoA. This becomes a problem when sampling from the failure density. This is because, as shown in Figure 3.4d, the failure density also has two MSRs, but the position of the high density and low density MSR, with respect to the BoAs, has flipped. As a result the now high density MSR is not populated at all, which will ultimately result in an underestimate for the probability of failure. Note that in this case, it is it the sudden change in gradients that have led SuS astray.

The reference probability of the piecewise linear is easy to approximate due to the simple geometry of the failure region and safe region,

$$\mathcal{F} = \{(x_1, x_2) \in \mathbb{R}^2 : x_1 \geq 4 \text{ or } x_2 \geq 5\}, \tag{3.51}$$

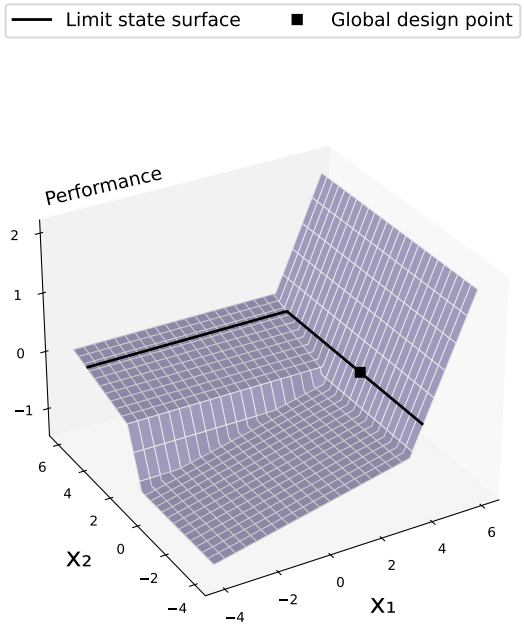
$$\mathcal{S} = \{(x_1, x_2) \in \mathbb{R}^2 : x_1 < 4 \text{ and } x_2 < 5\}. \tag{3.52}$$

It follows that

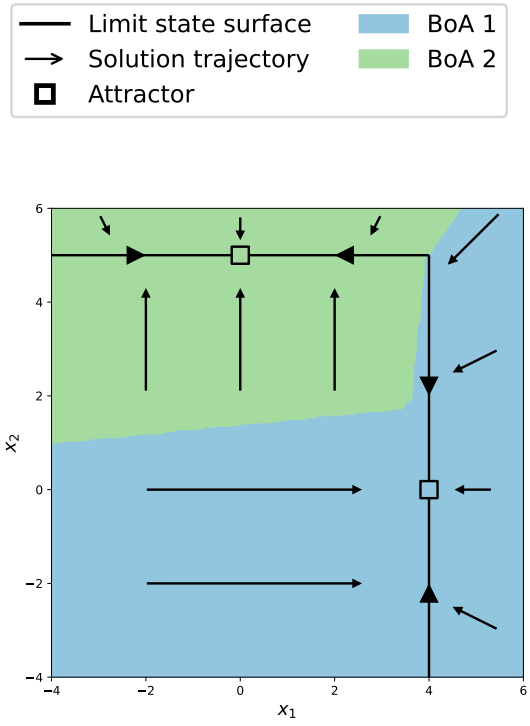
$$\mathbb{P}(\mathcal{F}) = 1 - \mathbb{P}(\mathcal{S}) \tag{3.53}$$

$$= 1 - \Phi(4) \cdot \Phi(5) \tag{3.54}$$

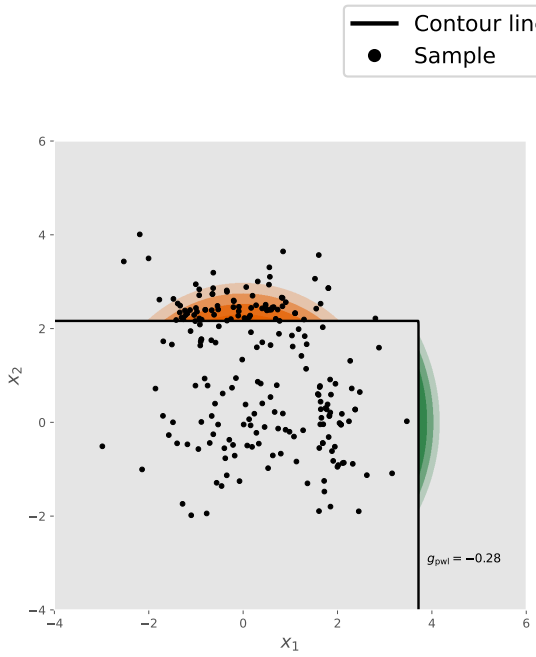
$$\approx 3.20 \times 10^{-5}. \tag{3.55}$$



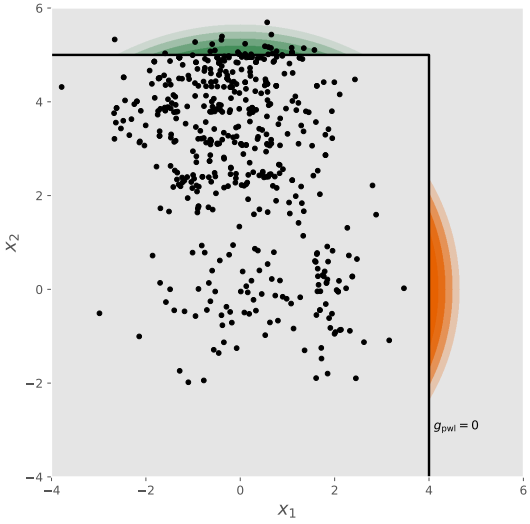
(a) Three dimensional surface plot.



(b) BoAs of the reliability problem.



(c) Intermediate level of subset simulation.



(d) Final level of subset simulation.

Figure 3.4: Analysis of subset simulation's behaviour when acting on the meatball function.

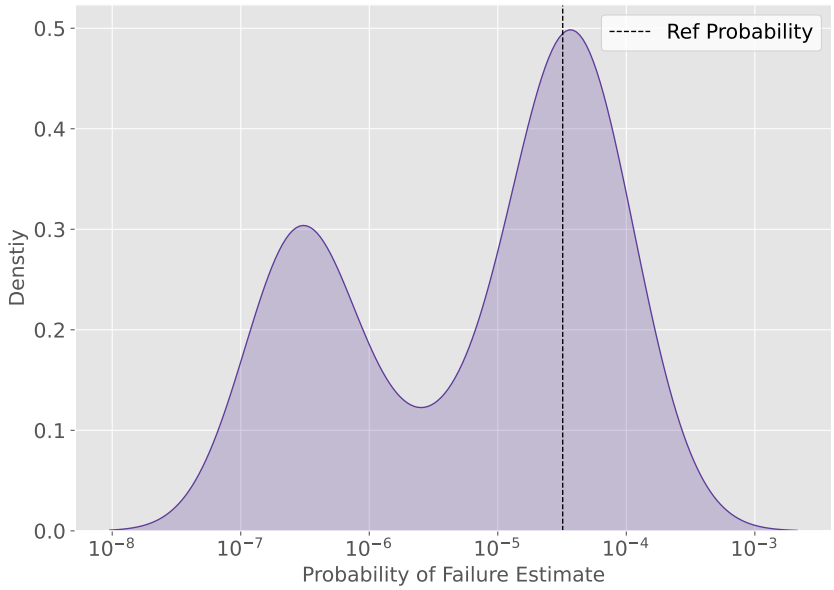


Figure 3.5: KDE of the empirical distribution of the SuS estimator for the piecewise linear function.

Because SuS is a stochastic algorithm, on some runs it will manage to populate the high density MSR of the failure density. However, in these cases, it has a tendency to overestimate the probability of failure. Figure 3.5 shows the estimator that is the result of SuS running on the piecewise linear function, with some fixed user-defined parameters, approximated by KDE. It can be seen that the estimator has a bimodal distribution, where the left mode corresponds to runs where the high density MSR is not populated and the right mode corresponds to runs where the high density MSR is populated. The mean of this empirical distribution is quite close to the reference probability. However, the variance of the estimator is so large that is practically unusable.

The second performance function taken from [19] is the meatball function, given as

$$g_{\text{mb}}(x_1, x_2) = -\frac{30}{\left(\frac{4(x_1+2)^2}{9} + \frac{x_2^2}{25}\right)^2 + 1} - \frac{20}{\left(\frac{(x_1-2.5)^2}{4} + \frac{(x_2-0.5)^2}{25}\right)^2 + 1} + 5. \tag{3.56}$$

Figure 3.6a shows a three dimensional surface plot of the meatball function and Figure 3.6b shows the BoAs. The meatball function has four failure BoAs. The geometry of the meatball function is difficult for intermediate methods to deal with since the global design point is separated from the origin by the neighbourhood of a local minima of the performance function. Since the geometry of the failure region is quite complex, it is not possible to approximate the probability of failure using analytical methods. Instead, the reference probability has been estimated as 1.13×10^{-5} using a MC estimator with a large sample size.

An early level of a run of SuS on the meatball function is shown in Figure 3.6c with associated intermediate threshold of -5.98. In this case, the intermediate density has two low density MSRs and two high density MSRs. The SuS algorithm is able to accurately

sample from the intermediate density by populating the two high density MSRs. However, at this point, two of the BoAs are not populated, including the one corresponding to the global design point. Figure 3.6b shows the SuS run after it has terminated alongside the failure density. Note that now there is only one high density MSR that has not been populated by SuS. Because the density of the BoA associated with the global design point is so low for early intermediate densities, it is very unlikely that SuS will be able to maintain samples within it. As a result, the meatball function is even more challenging for SuS than the piecewise linear performance function.

The third benchmark is the black swan function, originally posed in [18], and is given by

$$g_{\text{bs}}(x_1, x_2) = \begin{cases} x_1 - 3.5 & x_1 \leq 3.5, \\ x_2 - 3.5 & x_1 > 3. \end{cases} \tag{3.57}$$

Figure 3.7a depicts a three dimensional surface plot of the black swan function and Figure 3.7b shows its BoAs. In this case there is one failure BoA and one safe BoA. The high density region of the input dimension is dominated by the safe BoA. The discontinuity of the black swan function is what causes the multiple BoAs which results in issues for intermediate methods. Due to the simplicity of its failure region,

$$\mathcal{F} = \{(x_1, x_2) \in \mathbb{R}^2 : x_1 > 3, x_2 \geq 3.5\}, \tag{3.58}$$

the reference failure may be easily estimated,

$$\mathbb{P}(\mathcal{F}) = \Phi(-3) \cdot \Phi(-3.5), \tag{3.59}$$

$$\approx 3.14 \times 10^{-7}. \tag{3.60}$$

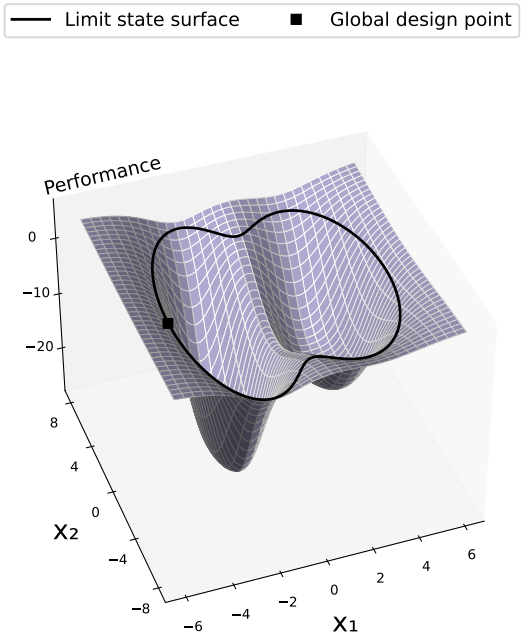
Figure 3.7c shows an early level of a SuS run on the black swan performance function., with corresponding intermediate threshold of -2.13. Despite the unusual shape of the intermediate failure region, the intermediate density is unimodal and unsurprisingly SuS is able to sample from it accurately. Note that the failure BoA is not populated at all. Figure 3.7d shows the final state of the SuS algorithm with the failure density. In this case, SuS has dramatically failed and has produced no failure samples at all. As a consequence it will incorrectly estimate the probability of failure to be 0. It should be noted that every intermediate distribution is actually unimodal in this case. The issue arises because that mode instantaneously jumps through the input space as the intermediate threshold rises. However, this is not true if SIS is used. In that case, the intermediate methods still experiences difficulties, but the intermediate densities are multimodal. Note that the algorithm is terminated in these cases when the level limit is reached.

The forth benchmark is original and has been created to explore how intermediate methods interact with the performance functions with multiple local maxima that are not in the failure region. Let $\varphi(\boldsymbol{x}; \mu)$ be a 2-dimensional normal PDF with mean μ and identity covariance matrix. The mixture performance function is defined as

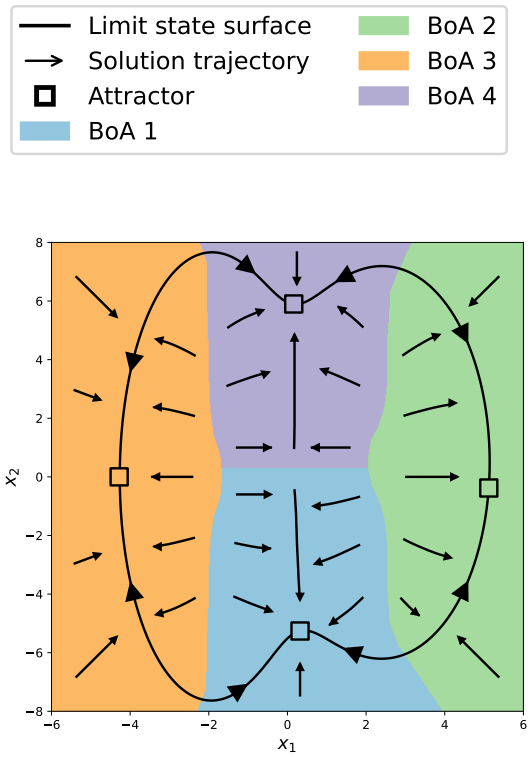
$$g_{\text{mix}}(\boldsymbol{x}) = \sum_{i=1}^{4} (w_i \varphi(\boldsymbol{x}; \mu_i)) - 0.04, \tag{3.61}$$

where

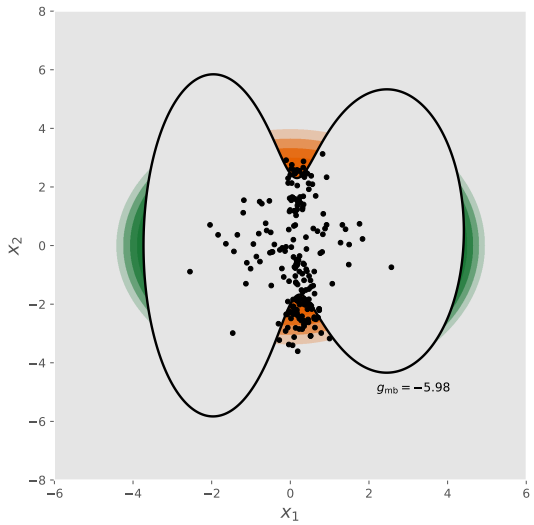
$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} = \begin{bmatrix} 0.4 \\ 0.2 \\ 0.2 \\ 0.2 \end{bmatrix} \text{ and } \begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \mu_4 \end{bmatrix} = \begin{bmatrix} 3 & 3 \\ 2 & -2 \\ -2 & 2 \\ -2 & -2 \end{bmatrix}. \tag{3.62}$$



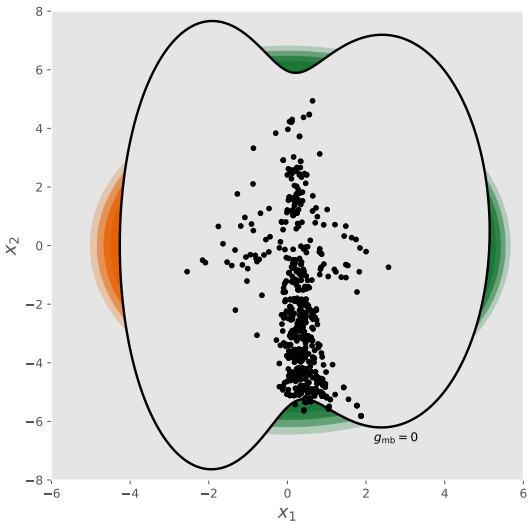
(a) Three dimensional surface plot.



(b) BoAs of the reliability problem.

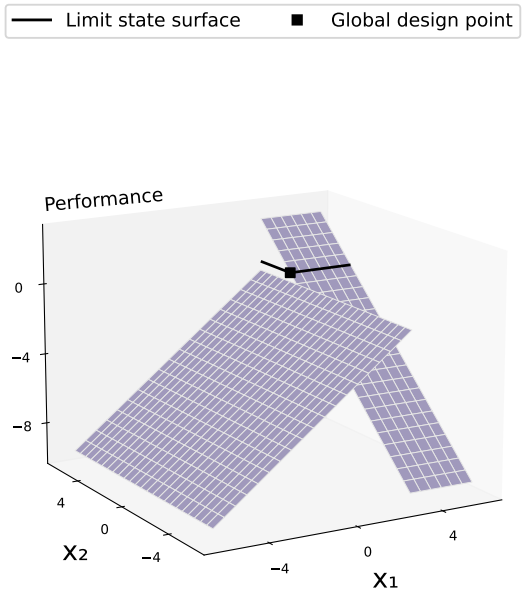


(c) Intermediate level of subset simulation.

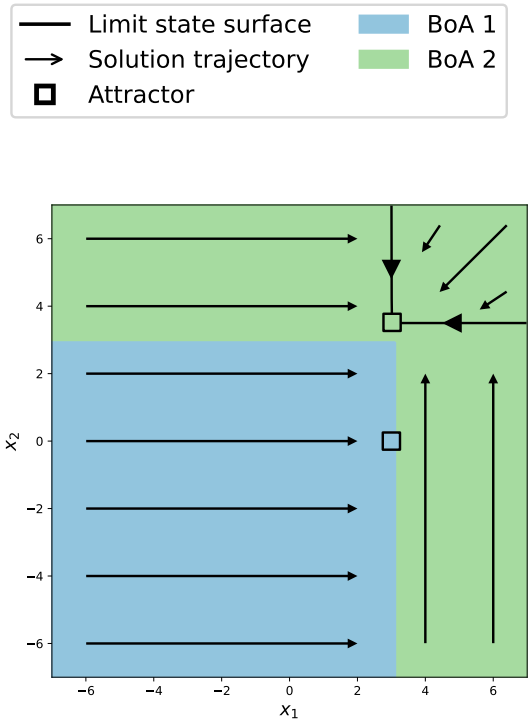


(d) Final level of subset simulation.

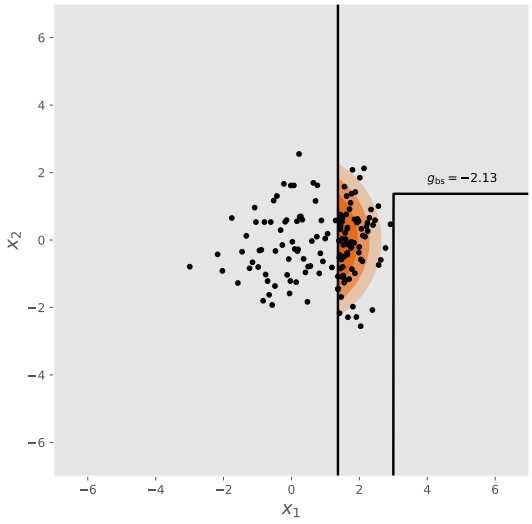
Figure 3.6: Analysis of subset simulation's behaviour when acting on the meatball function.



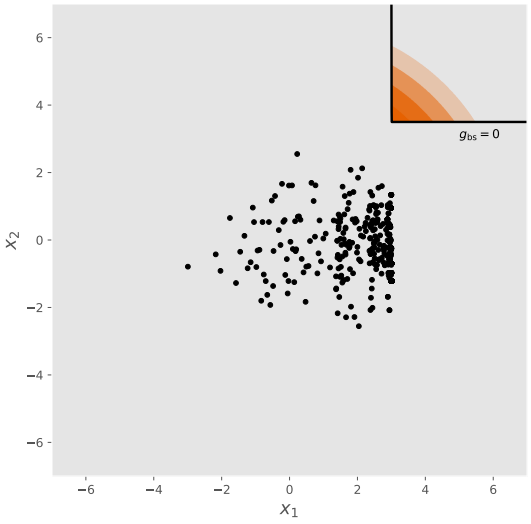
(a) Three dimensional surface plot.



(b) BoAs of the reliability problem.



(c) Intermediate level of subset simulation.



(d) Final level of subset simulation.

Figure 3.7: Analysis of subset simulation's behaviour when acting on the black swan function.

The 0.04 adjustment is only made so that the critical threshold is the conventional value of 0.

Figure 3.8a shows the three dimensional surface plot of the mixture performance function and Figure 3.8b shows the BoAs. Note that the failure region is circular. The radius of this circle, denoted as r, may be approximated,

$$\frac{0.4}{2\pi}\exp\left(-\frac{r^2}{2}\right) = 0.04 \implies r^2 = -2\ln(0.2\pi). \tag{3.63}$$

where it has been assumed that the other component of the mixture have negligible impact on the failure region. The probability of failure may then be approximated as

$$\mathbb{P}(\mathcal{F}) = \mathbb{P}((X_1 - 3)^2 + (X_2 - 3)^2 \leq r^2), \tag{3.64}$$

$$= F_{\chi^2_2(\lambda)}(r^2), \tag{3.65}$$

$$\approx 2.19 \times 10^{-4}. \tag{3.66}$$

where $F_{\chi^2_2(\lambda)}$ is the CDF of the non-central chi-square distribution with degrees of freedom set to 2 and the non-centrality parameter $\lambda = 18$.

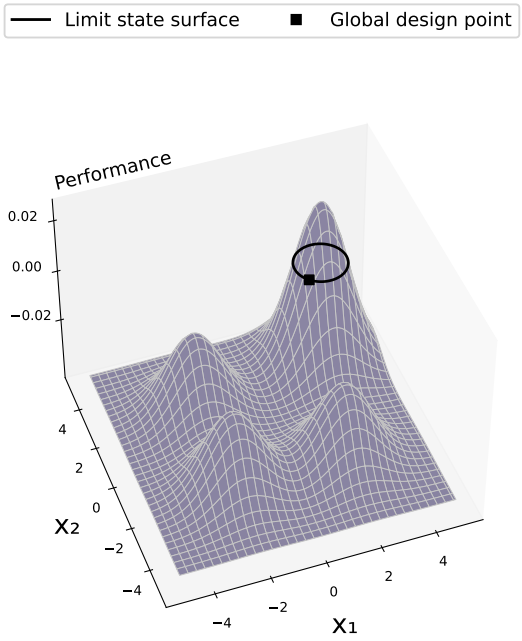
Figure 3.8c depicts an early level of a SuS run on the mixture performance function with intermediate threshold -0.03. At this level, there are three high density MSRs and one low density MSRs that corresponds to the only failure BoA. Note that at this level, the failure BoA has no samples in it at all. Figure 3.8c shows the all of the samples produced by the SuS after it has terminated alongside the failure density. Just as in the SuS run example on the previous benchmark, no failure samples are produced, and as a result the estimate for the probability of failure will incorrectly be given as 0. Note that, again, the failure density is unimodal in this case.

The previous four benchmarks have been contrived as counterexamples to intermediate methods specifically. The next two examples are more practical reliability problems. The fifth benchmark deals with the forced vibration of the two-degree-of-freedom (TDOF) mass spring system [11] depicted in Figure 3.9. The masses are $M_1 = M_2 = 2000$ kg, the modal damping ratios are $\eta_1 = \eta_2 = 0.02$, and forcing function acting on M_2 is given as $P(t) = 2000\sin(11t)$N. Let the stiffness parameters K_1 and K_2 have independent log-normal distributions with mean 2.5×10^5 and CoV 0.2. The standard normal input variables are transformed so that they have the required distribution, $K_1, K_2 = T(x_1, x_2)$. Let $r_1(t; K_1, K_2)$ be the displacement of the first mass at some time, given stiffness constants. The TDOF performance function is given as

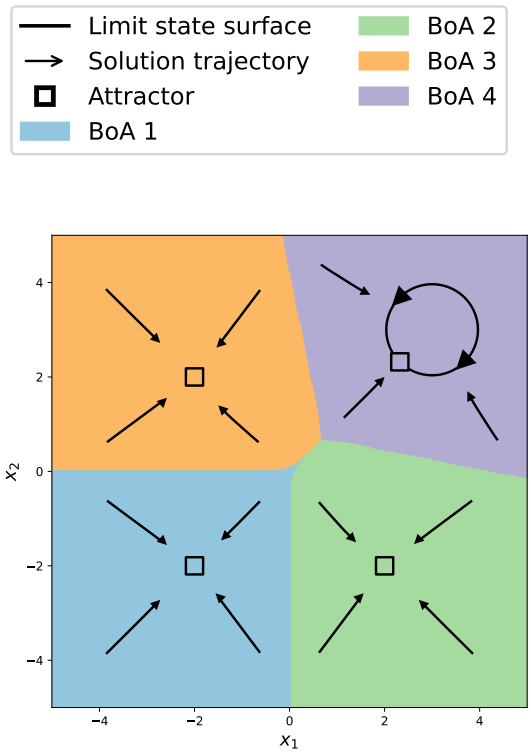
$$g_{\text{tdof}}(x_1, x_2) = \max_{0 \leq t \leq 20} r_1(t; K_1, K_2) - 0.024. \tag{3.67}$$

The failure region of this reliability problem has quite complex geometry an so it is not possible to approximate the reference probability of failure analytically. Instead, the reference probability has been estimated as 2.31×10^{-5} using an MC estimator with a large number of samples. Figure 3.10a shows a three dimensional surface plot of the TDOF performance function and Figure 3.10b shows its BoAs. Note that for clarity the many safe BoAs have been combined into one.

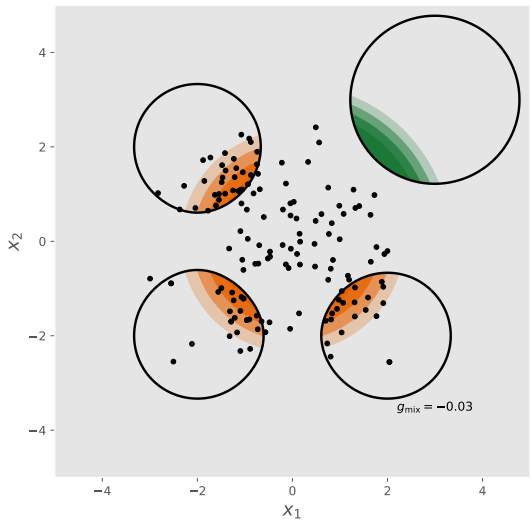
Figure 3.10c shows an early level of a SuS run on the TDOF performance function with intermediate threshold -0.01. The intermediate density at this level has high density



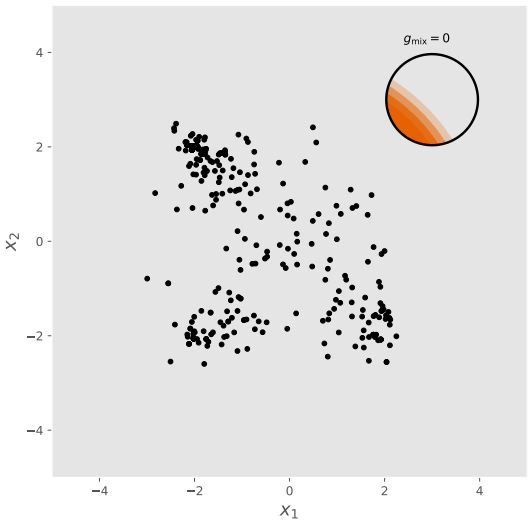
(a) Three dimensional surface plot.



(b) BoAs of the reliability problem.



(c) Intermediate level of subset simulation.



(d) Final level of subset simulation.

Figure 3.8: Analysis of subset simulation's behaviour when acting on the normal mixture function.

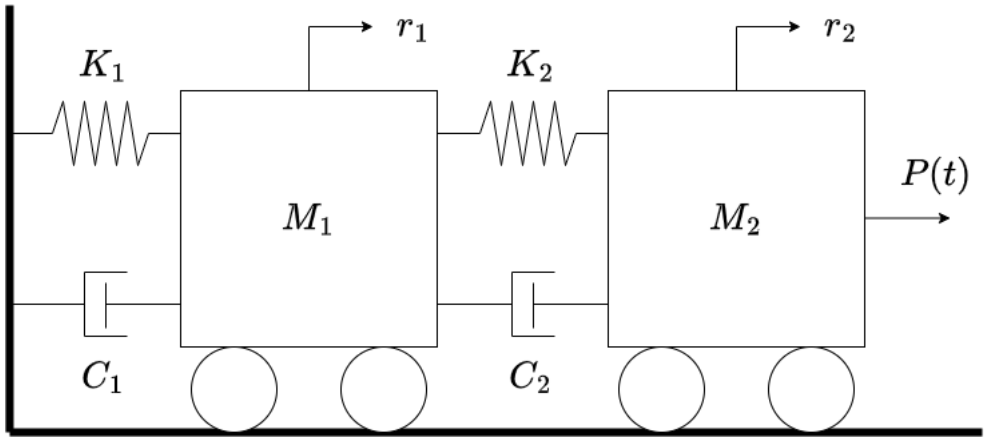


Figure 3.9: Two-degree-of-freedom mass spring system.

MSR, which is populated, and a low low density MSR, which is not. 3.10d shows all of the samples SuS has produced after termination and the failure density. The failure density has two high density MSR, only one of which is populated and thus the resulting estimate for the probability of failure will likely be an underestimate.

The sixth benchmark considers the passive vehicle suspension model depicted in Figure 3.11. The performance function, taken from [9], models the road-holding ability of a vehicle. The input variables are stiffness c (kg/cm), tire stiffness c_k (kg/cm) and damping coefficient k (kg/cms), all with normal distributions with means and standard deviations of $(424, 1480, 47)$ and $(10, 10, 10)$ respectively. The input standard normal variables are transformed to have the correct distributions, $T(x_1, x_2, x_3) = (c, c_k, k)$, and the vehicle performance function is defined as,

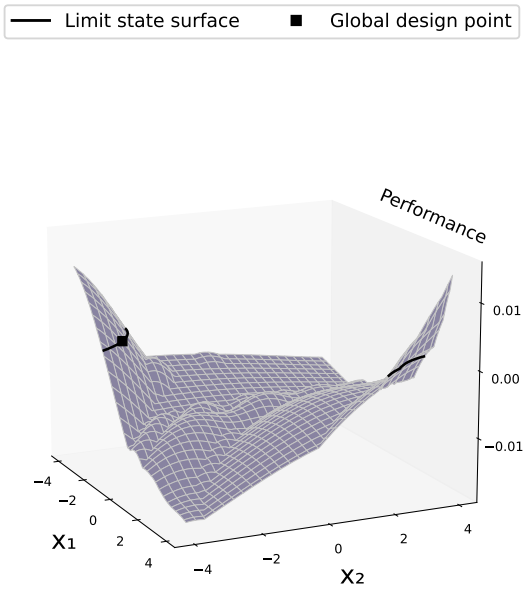
$$g_{\text{veh}}(x_1, x_2, x_3) = 1 - \left(\frac{\pi A V m}{b_0 G^2 k}\right)\left(\left(\frac{c_k}{M+m} - \frac{c}{m}\right)^2 + \frac{c^2}{Mm} + \frac{c_k k^2}{M^2 m}\right), \tag{3.68}$$

where $A = 1$ cm^2/cycle m, $b_0 = 0.27$, $V = 10$ m/s, $M = 3.2633$ kg s^2/cm, $G = 981$ cm/s^2 and $m = 0.8158$ kg s^2/cm.

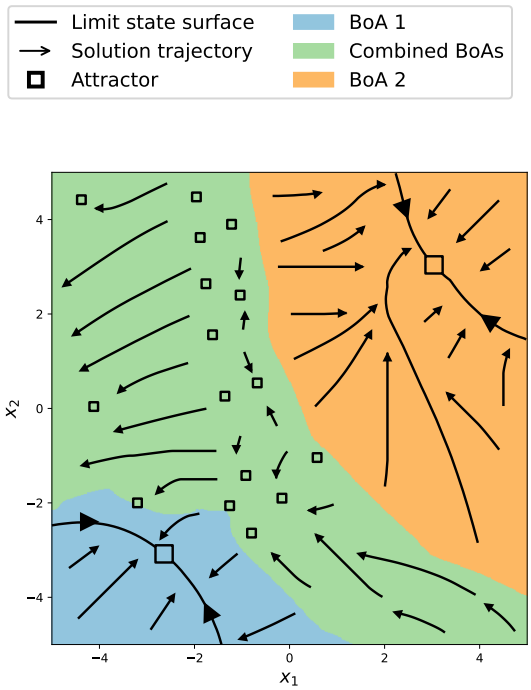
Figure 3.12a shows a three dimensional plot of vehicle performance function and Figure 3.12b shows its BoAs. In order to make the visualisation possible, a two dimensional version of the performance function is considered in these two figures. In particular, the second dimension is fixed as $x_2 = 0$. This lower-dimensional variant maintains the same challenging topology as the original function. Note that one of the attractors is not shown since it is located outside the bounds of the figure. The reference probability has been estimated as 1.30×10^{-6} using and MC estimator with a large sample size. Figure 3.12c shows an early level of a SuS run on the vehicle function with intermediate threshold -83. At this level there is one high density MSR that is populated and one low density MSR that is not. Figure 3.12d shows the final level of the SuS run with the failure density, which now only has one high density MSR which is not populated.

The final benchmark is a financial application [2]. Given a random variable $Z = (Z_1, \ldots, Z_n)$, Let the loss function be defined as

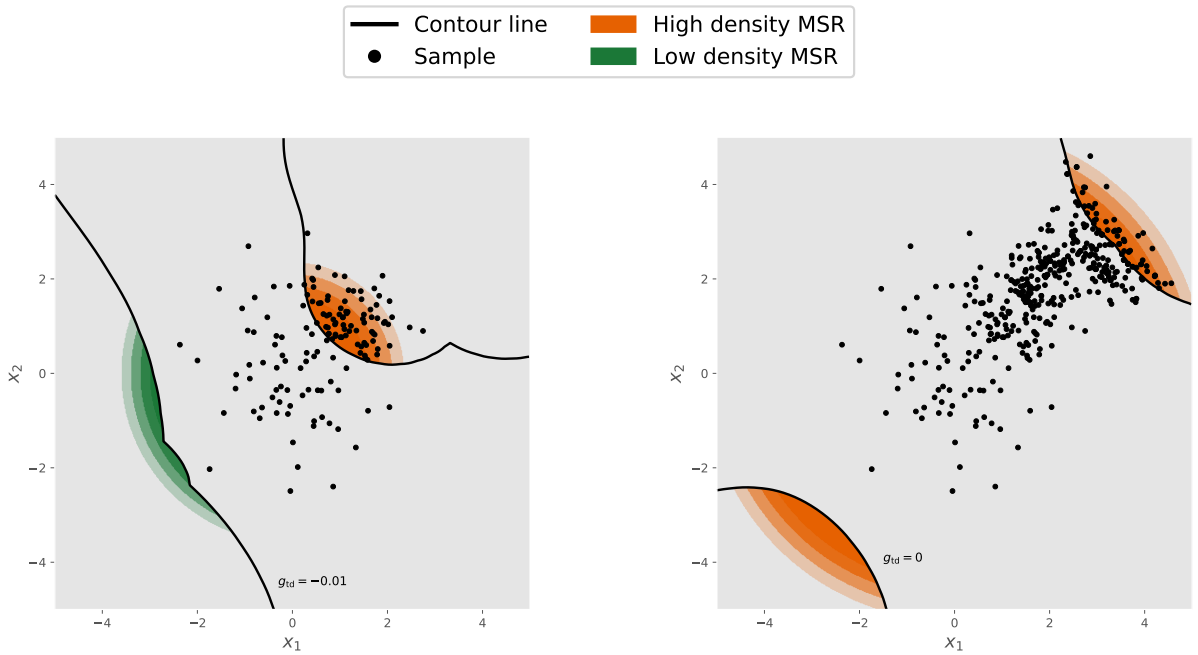
$$L(Z) = \sum_{j=1}^{n} \mathbb{1}_{\{Z_j \geq 0.5\sqrt{n}\}}. \tag{3.69}$$



(a) Three dimensional surface plot.



(b) BoAs of the reliability problem.



(c) Intermediate level of subset simulation.

(d) Final level of subset simulation.

Figure 3.10: Analysis of subset simulation's behaviour when acting on the two-degree-of-freedom function.

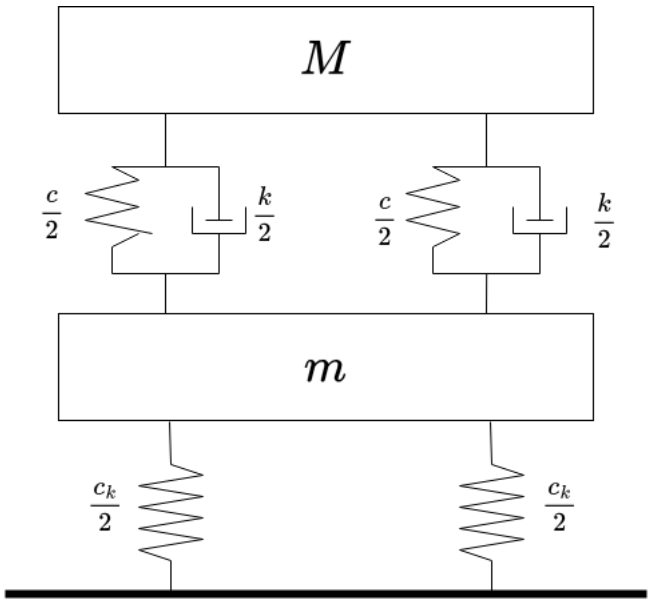


Figure 3.11: Passive vehicle suspension model.

The failure region is defined as the loss function being greater than bn, where two cases are considered: $b = 0.45, n = 30$; $b = 0.25, n = 100$. The random variable Z is defined as function of other random variables,

$$Z_j = \left(qU + (1-q^2)^{1/2}\eta_j\right)\mu^{-1/2} \tag{3.70}$$

where $U \sim \mathcal{N}(0,1)$, $\mu \sim \text{Gamma}(6,6)$, $\eta_j \sim \mathcal{N}(0,9)$ for $1 \leq j \leq n$, and $q = 0.25$. The same realisation of U and μ are used for all Z_j. Note that the dimension of this reliability problem is $d = n+2$. These random variables must be transformed into standard normal space. That is, given standard normal input $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$, Z may be rewritten as,

$$Z_j = \left(qx_1 + 3(1-q^2)^{1/2}x_{j+2}\right)\left[F_\Gamma^{-1}\left(F_\mathcal{N}(x_2)\right)\right]^{-1/2}, \tag{3.71}$$

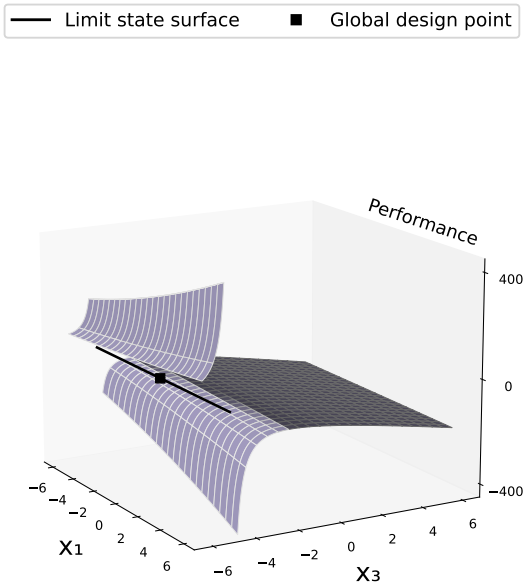
for $1 \leq j \leq n$, where $F_\mathcal{N}$, F_Γ are the cumulative distribution functions of $\mathcal{N}(0,1)$ and $\text{Gamma}(6,6)$ respectively. Finally, the loss performance function is defined,

$$g_{\text{loss}}(\boldsymbol{x}) = \sum_{j=1}^{n} \mathbb{1}_{\{Z_j \geq 0.5\sqrt{n}\}} - bn. \tag{3.72}$$

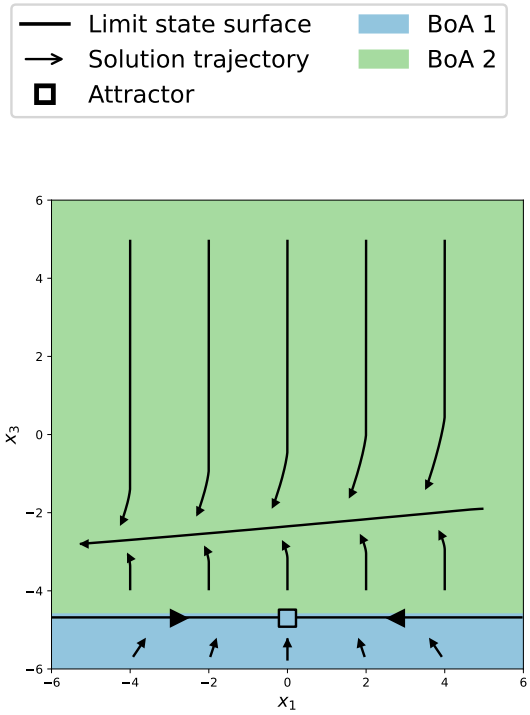
Note that a signifiant portion of the probability density of the failure region is represented by inputs which have a performance of exactly 0. Since the inequality used in the definition of the probability of failure in this work was not strict, whilst it is in [2], a small $\epsilon > 0$ is taken from the output of the performance functions in this numerical example. Clearly this benchmark is difficult to visualise due to it being inherently high dimensional. It should be noted that it is multimodal and has a lot of symmetries.

3.2.2 Evaluation

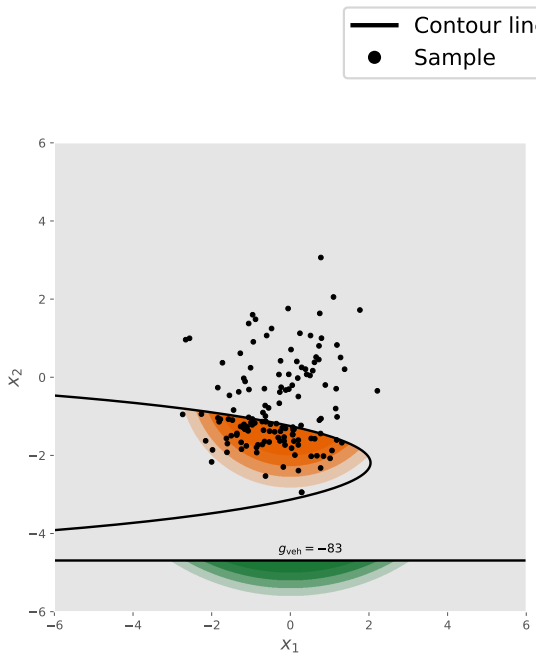
Some criteria is required in order to evaluate how well a reliability method performs on the established benchmarks. Each reliability method that is considered will be ran 100 times on each of the benchmarks, resulting in 100 estimates for the probability of failure,



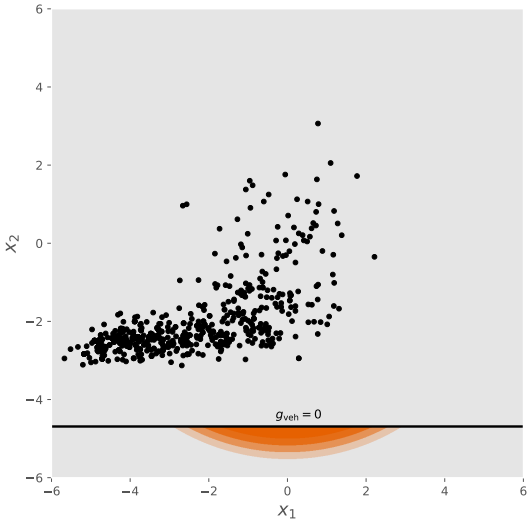
(a) Three dimensional surface plot.



(b) BoAs of the reliability problem.



(c) Intermediate level of subset simulation.



(d) Final level of subset simulation.

Figure 3.12: Analysis of subset simulation's behaviour when acting on the vehical suspension function.

denoted as $\hat{P}_1, \ldots, \hat{P}_{100}$. In each case, these estimate will be compared to the reference probability, denoted as $P_{\mathcal{F}}^{\text{ref}}$, which are summarised in Table 3.1 alongside the dimensions that will be considered using Equation 3.49.

g	Dimensions	Reference P_F
g_{pwl}	$2, 100$	3.20×10^{-5}
g_{veh}	$3, 99$	1.30×10^{-6}
g_{mix}	$2, 100$	2.19×10^{-4}
g_{bs}	$2, 100$	3.14×10^{-7}
g_{tdof}	$2, 100$	2.31×10^{-5}
g_{mb}	$2, 100$	1.13×10^{-5}
g_{loss}	32	4.29×10^{-3}
	102	1.82×10^{-3}

Table 3.1: Reference probabilities.

The most common measure of accuracy used in reliability analysis is the empirical CoV of the estimators. Note that this quantity does not use the reference probability in its calculation. Instead, the mean of the estimates is used, which in the case of an unbiased estimator, should be roughly equivalent to the reference probability anyway. However, when estimators have a very large variance, their mean is often a poor approximation of the reference probability, even when they are unbiased. Because the benchmarks in this work have been chosen deliberately to cause current reliability methods to dramatically fail, the resulting estimators will indeed often have a very large variance. As a result, the *normalised root mean squared error (NRMSE)* will be used instead of the CoV to measure the accuracy of the estimators, which is given as

$$\text{NRMSE} := \frac{\sqrt{\frac{1}{100}\sum_{i=1}^{100}(\hat{P}_i - P_{\mathcal{F}}^{\text{ref}})^2}}{P_{\mathcal{F}}^{\text{ref}}}. \tag{3.73}$$

Note that the NRMSE is identical to the CoV, except the reference probability is used instead of the empirical mean.

A lot of work in reliability analysis focuses on benchmarks for which existing methods already work quite well. The goal in such work is to make small but important improvements to the accuracy of of existing methods, and so the CoV or NRMSE are appropriate metrics to consider. Since many existing methods completely fail on the benchmarks chosen in this work, such metrics are not particularly informative, and the NRMSE has been mostly included to provide consistency with other work in reliability analysis. As will be seen, in many of the numerical experiments the NRMSE will be extremely large, and past a certain point, the degree to how large stops being relevant. That is, an estimator with an NRMSE of 1 is just as unviable for practical reliability analysis as an estimator with NRMSE of 100.

As a consequence, the main focus will be another custom metric, called the *non-degeneracy percentage (NDP)*, given as

$$\text{NDP} := \frac{1}{100}\sum_{i=1}^{100} \mathbb{1}\left[\frac{|\log(P_{\mathcal{F}}^{\text{ref}}) - \log(\hat{P}_i)|}{\log(P_F^{\text{ref}})} \leq 0.1\right] \tag{3.74}$$

When an estimate satisfies the condition in the Equation 3.74, it will be referred to as *non-degenerate* and otherwise it will be refereed to as *degenerate*. The idea behind

this definition is that a degenerate estimate is so far from the reference probability that something has gone dramatically wrong with the reliability method. The reason this metric uses logarithms is so underestimates and overestimates are treated in the same way.

Of course, an MC estimator could be used to easily achieve and accurate estimate for the probability of failure for any of the benchmarks. The issue is the computational cost, which is assumed in this work to be well approximated by the number of performance function evaluations. That is, a low NRMSE or a high NDP is only useful if the mean number of performance function evaluations is low. One approach is to combine an measure of accuracy with the mean number of performance function evaluations, resulting in a single scalar quantity that can be use to easily compare the performance of competing reliability methods. In this work the goal is to design methods that can achieve a 100% NDP with a comparable computational cost to existing methods. As a result the preciseness of combined statistic is not required and the mean number of performance function evaluations will be reported as a separate statistic.

Each reliability method will only use one set of fixed user-defined parameters. That is, a sensitivity analysis will not be provided. This omission should be justified in some way for two reasons. Firstly, the poor performance of existing methods may be exaggerated by deliberately choosing poor user-defined parameters. To account for this, for each existing method, the user-defined parameters have been set at values that are very commonly used in reliability analysis literature. In particular, since the benchmarks are considered black-box reliability problems, there should be no information available to tune the parameters anyway. In addition to this, many of the benchmarks have been chosen since they are widely regarded to be difficult for intermediate methods to tackle, and so it is unlikely that the poor performance is due to the particular set of user-defined parameters that are considered here. Secondly, the user-defined parameters of the novel methods may have been over-fit to the particular benchmarks that are being considered. To somewhat mitigate this issue, the same user-defined parameters will be used on all the benchmarks, which themselves are quite diverse.

There are other features of reliability methods that will not be explicitly tested but will be discussed throughout. Perhaps the most important of these features is the ability for the the reliability method to accurately sample from the failure density so that probabilistic failure analysis can be carried out. Most of the time, if the probability of failure can be accurately estimated, then an intermediate method will produce a good approximation of the failure density. However, as seen in Section 3.1.2, there are examples where this is not the case, especially when the performance function exhibits a lot of symmetry.

Some reliability methods naturally output additional information about the failure region and failure density which could be useful to a practitioner. This information is not captured by the evaluation metrics described above. For instance, the novel methods that will be introduced in this work will explicitly attempt to model the number of MSRs the failure region has, whereas a method like SuS does not naturally provide this type of information. On the other hand, SuS is able to generate estimates for values of the CCDF other than the critical threshold, whereas the novel methods presented in this work are not able to do this naturally.

Perhaps the most important property of a reliability method that is not explicitly measured by the stated evaluation metrics is whether the method gives any indication to the user as to when something might have gone wrong. In the examples of CRP that have been discussed in this chapter, when SuS dramatically fails to sample accurately from

the intermediate densities resulting in a poor estimate for the probability of failure, there is no way for the user to know that this has happened. This property will be referred to as *silent degeneracy*. The novel methods in this work will attempt to have some sort of system that lets the user know that estimate for the probability of failure it yields may be degenerate.

Since the CRP is a black-box reliability problem it is important to consider no free lunch theorems when evaluating the quality of competing the reliability methods. Originally conceived in the domains of optimisation and search [58], no free lunch theorems have also been derived for reliability analysis [59]. Informally, these theorems imply that given any two reliability methods it is always possible to construct a reliability problem for which one outperforms the other. Since all MC reliability methods essentially attempt to reduce the variance of a MC estimator by either explicitly or implicitly making some assumptions about the reliability problem, it is always possible to find a reliability problem that violates theses assumptions. It is for this reason that it has been ensured that the benchmarks in this work include realistic reliability problems as well as contrived examples. With that said, the set of benchmarks considered in this work is quite small and so just because a reliability method outperforms another on this set of benchmarks does not mean that it is necessarily a better method.

Now the results of the numerical experiments of SuS, SIS and iCE on the benchmarks are presented. The values of the user-defined parameters for each algorithm are given in Table 3.2, Table 3.3 and Table 3.4. The MCMC algorithm used for SuS was the pCN algorithm and for SIS it was a cMH algorithm with an adaptive proposal scale. The results of the numerical experiments are presented in Table 3.5, Table 3.6 and Table 3.7.

Both SuS and SIS have fairly similar results when looking at the NDP of each experiment. This is not particularly surprising, since they are quite similar algorithms. In both cases, for all benchmarks aside from the loss performance function, both have an unacceptably high NDP in both low dimensions and high dimensions. It should be noted that the good results in the case of the loss performance function are likely due to significant amount of symmetry that it possesses. for both these algorithms, the computational cost and NDP seem to be somewhat independent of the dimension of the problem. In these particular experiments, it appears that in general SuS has slightly higher NDP values than SIS, though this should not be read into too much, since this could easily be the result of the specific parameters that have been used for each or the different MCMC algorithms. The results of iCE are quite good for most of the problems in low dimensions. That is, the NDP in each case is reasonably close to 100%. The exception to this is the low dimensional black swan function, where the the NDP is unacceptably low. In the high-dimensional examples, iCE struggles with all of the benchmarks aside from the loss performance function.

Parameter	Value
Level size (N_{L})	2000
Level probability (p_{L})	0.1
Proposal scale (σ_{p})	0.5
Level limit (n_{limit})	10

Table 3.2: SuS user-defined parameter values.

Parameter	Value
Level size (N_{L})	2000
Level probability (p_{L})	0.1
Burn in (n_{burn})	0
Target CoV (δ_{target})	1.5
Level limit (n_{limit})	30

Table 3.3: SIS user-defined parameter values.

Parameter	Value
Level size (N_{L})	2000
Level limit (n_{limit})	30
Target CoV (δ_{target})	1.5
Initial components (k_{init})	4

Table 3.4: iCE user-defined parameter values.

3.3 Evolutionary multimodal optimisation

Problems of multimodality have been studied in fields outside of reliability analysis and so it follows that the ideas used in these fields to overcome such issues may be applicable to CRPs. In particular, this section introduces a class of optimisation methods called evolutionary algorithms which have been adapted for multimodal optimisation.

3.3.1 Evolutionary algorithms

Given an *objective function*, $g_{\mathrm{obj}} : \mathbb{R}^d \to \mathbb{R}$, let an *optimisation problem* be defined as finding

$$\boldsymbol{x}^* := \underset{\boldsymbol{x}\in\mathbb{R}^d}{\operatorname{argmax}}\, g_{\mathrm{obj}}(\boldsymbol{x}). \tag{3.75}$$

Locating the global design point of a reliability problem is an example of an optimisation problem, as defined by Equation 2.9. The type of method that is used to solve an optimisation problem is often determined by what information is available regarding the objective function. Many techniques, such as the HLRF algorithm, require access to gradient information. When the objective function is black-box, like the performance function in the CRP, alternative methods must be used.

In the context of optimisation, heuristics are methods which are not backed by mathematical rigour, but instead offer approximate solutions which are often found to be good enough in many practical contexts. Heuristics tend to assume very little about the objective function and so are often applied to black-box optimisation problems. *Evolutionary algorithms (EAs)* are a class of optimisation heuristics. More specifically, they are often referred to as metaheuristics. This is because it is more accurate to describe EAs as general frameworks for creating and combining heuristics rather than being heuristics themselves. Each EA refers to a collection of many different specific implementations of optimisation algorithms. This makes EAs highly flexible such that they can be tailored to particular problems.

EAs are metaheuristics inspired by biological processes and so often adopt biological terminology. An individual is a point $\boldsymbol{x} \in \mathbb{R}^d$ and a population is a finite set of individuals.

g	Dimension	Mean P_F	NRSME	NDP	Mean g evals
g_{pwl}	2	2.72×10^{-5}	0.97	51%	1.05×10^4
	100	2.85×10^{-5}	1.12	49%	1.05×10^4
g_{veh}	3	1.71×10^{-9}	1.00	0%	1.81×10^4
	99	2.98×10^{-6}	22.83	0%	1.80×10^4
g_{mix}	2	2.18×10^{-4}	0.47	89%	7.72×10^3
	100	2.10×10^{-4}	0.67	77%	7.89×10^3
g_{bs}	2	2.32×10^{-7}	1.51	34%	1.60×10^4
	100	3.04×10^{-7}	1.81	39%	1.56×10^4
g_{td}	2	2.51×10^{-5}	0.92	77%	9.65×10^3
	100	2.20×10^{-5}	0.67	84%	9.52×10^3
g_{mb}	2	1.23×10^{-5}	2.81	9%	1.35×10^4
	100	8.73×10^{-6}	1.99	7%	1.39×10^4
g_{loss}	32	4.26×10^{-3}	0.16	100%	5.60×10^3
	102	1.88×10^{-3}	0.16	100%	5.62×10^3

Table 3.5: SuS results on benchmarks.

The fitness of an individual is its value under the objective function, $g_{\text{obj}}(\boldsymbol{x})$. EAs can roughly unified under the following loose framework:

Evolutionary algorithm:

1. Create an initial population. Options for doing this include sampling from a distribution, using a explicit practitioner set starting position or Latin hypercube sampling schemes.

2. Evaluate the fitness for all the individuals in the population.

3. Check a stopping condition to see if the algorithm should terminate. Possible options include a fixed fitness evaluation budget, a time budget and convergence in either the maximum fitness found or the position of individuals.

4. Select some of the population to be parents. Typically, individuals with higher fitness should have a higher chance of being selected as a parent. For example, given a population $(\boldsymbol{x}_i)_{i=1}^N$, a roulette wheel selection strategy randomly selects the parents with replacement according to the probabilities

$$\frac{g_{\text{obj}}(\boldsymbol{x}_i)}{\sum_{j=1}^N g_{\text{obj}}(\boldsymbol{x}_j)}, \tag{3.76}$$

   for $1 \leq i \leq N$.

5. Generate offspring from parents through mutation and crossover procedures, where a mutation uses one parent and a crossover uses more than one parent. A simple example of mutation is to just add noise to a parent $\boldsymbol{x}$ to create an offspring,

$$\boldsymbol{x}' = \boldsymbol{x} + \mathcal{N}(\boldsymbol{0}_d, \boldsymbol{I}_d). \tag{3.77}$$

   A simple example of a crossover procedure, given two parents $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, is a linear crossover,

$$\boldsymbol{x}' = \frac{\boldsymbol{x}_1 + \boldsymbol{x}_2}{2}. \tag{3.78}$$

g	Dimension	Mean P_F	NRSME	ND%	Mean g evals
g_{pwl}	2	4.16×10^{-5}	3.65	20%	1.25×10^4
	100	2.95×10^{-5}	2.13	19%	1.26×10^4
g_{veh}	3	0	1.00	0%	1.65×10^4
	99	0	1.00	0%	1.67×10^4
g_{mix}	2	2.15×10^{-4}	1.04	50%	9.30×10^3
	100	2.04×10^{-4}	1.16	43%	9.42×10^3
g_{bs}	2	2.16×10^{-7}	3.36	11%	1.43×10^4
	100	4.35×10^{-7}	4.93	20%	1.41×10^4
g_{td}	2	2.15×10^{-5}	2.44	27%	1.00×10^4
	100	2.10×10^{-5}	2.82	21%	1.00×10^4
g_{mb}	2	8.01×10^{-6}	3.70	12%	1.47×10^4
	100	8.49×10^{-6}	3.05	2%	1.52×10^4
g_{loss}	32	4.17×10^{-3}	0.21	99%	7.86×10^3
	102	1.86×10^{-3}	0.22	99%	8.00×10^3

Table 3.6: SIS results on benchmarks.

The offspring are added to the population.

6. Remove some individuals from the population. In general, individuals with lower fitness should be more likely to be removed. Go to Step 3.

Examples of popular EAs include genetic algorithms [60,61], evolution strategies [62, 63] covariance matrix adaptation evolution strategies [64–66], differential evolution [67,68] and particle swarm optimisation [69].

It is important to note that SuS, SIS and iCE can all be described according to the above framework, which implies that they could be considered EAs. Indeed, SuS has been previously adapted for use as an optimisation algorithm where the performance function is analogous to the objective function [70]. This relationship between reliability analysis and EAs has also been utilised in the other direction. Many EAs have been used to locate the design point including particle swarm optimisation [71] genetic algorithms [72], artificial bee colony [73], Harris hawk optimisation [74] and slime mold optimisation [75].

3.3.2 Niching

Multimodal optimisation (MMO) is the problem of finding multiple locally optimal solutions to an optimisation problem. This is distinct from an optimisation problem with a multimodal objective function, since in that case, even though local optima my be explored, the goal is to converge to one global optima. Accordingly, optimisation algorithms are designed such that they do not try to maintain the locally optimum solutions, but instead converge to the global optima. Typically, optimisation algorithms must be adapted in some way so that they are suitable for MMO.

EAs are natural candidates for MMO, since the population of evolving individuals have the ability to explore the neighbourhoods of many local optima. Ultimately however, EAs are designed to converge to one optimal solution, given enough time. In *evolutionary multimodal optimisation (EMO)*, *niching* methods are applied to EAs in order to maintain sets of individuals, called *niches*, in the neighbourhoods of many different local optima. The name niching is also biologically inspired, where a niche is a role a specific species

g	Dimension	Mean P_F	NRSME	ND%	Mean g evals
g_{pwl}	2	3.10×10^{-5}	0.17	99%	5.20×10^{4}
	100	2.94×10^{-7}	0.99	0%	2.29×10^{4}
g_{veh}	3	1.20×10^{-6}	0.30	99%	1.52×10^{4}
	99	9.26×10^{-7}	2.88	9%	9.70×10^{3}
g_{mix}	2	2.19×10^{-4}	0.11	99%	2.75×10^{4}
	100	3.53×10^{-4}	11.37	0%	5.60×10^{4}
g_{bs}	2	2.07×10^{-7}	2.02	39%	5.68×10^{4}
	100	5.16×10^{-7}	5.49	5%	6.00×10^{4}
g_{td}	2	2.24×10^{-5}	0.22	95%	3.78×10^{4}
	100	7.03×10^{-6}	0.70	1%	1.94×10^{4}
g_{mb}	2	1.10×10^{-5}	0.13	99%	3.13×10^{4}
	100	1.66×10^{-6}	1.69	0%	6.00×10^{4}
g_{loss}	32	4.26×10^{-3}	0.08	100%	5.94×10^{4}
	102	1.84×10^{-3}	0.06	100%	1.07×10^{4}

Table 3.7: iCE results on benchmarks.

plays in an environment. Niching techniques are often initially suggested with respect to a specific evolutionary algorithm. However, since all evolutionary algorithms mostly follow the same general framework, it is often possible to adapt niching algorithms for use with many different evolutionary algorithms.

The first niching method to be developed was *preselection* [76]. The idea behind this technique is that only the individuals used in process of creating an offspring could be replaced by that offspring. This means that offspring should only replace similar individuals in the population, and so niches should be able to be maintained. A variation of preselection is *crowding* [77]. The idea is very similar. The difference is that offspring are not only restricted to replacing their parents, but they can also replace other similar individuals which are not necessarily parents. The degree similarity tends to be defined using the Euclidean metric.

In *fitness sharing* [78], the fitness values of individuals are adjusted for the selection phase of the algorithm. Individuals which are close to other individuals, where is close is defined using the Euclidean distance, then the fitness value is lowered. Alternatively, if an individual is relatively isolated, its fitness value is increased. Since the selection step selects individuals for the next population based on the fitness values, this will help to maintain niches. *Clearing* [79] is similar to fitness sharing in that it adjusts the fitness values so the niching happens in the selection section of the algorithm. Rather than sharing the fitness amongst the niche, in clearing, a winner is selected for each niche that takes all the collective fitness of that niche. The rest of the individuals in that niche are assigned zero fitness. Again, the niches are determined using the Euclidean distance.

The idea of *speciation* [80] is to label all the individuals in a population with a species. Then, throughout the process of creating the generation, it is ensured via various means that at least one of each species survives into the next generation. *Islanding* [81] is similar to speciation, but instead the species are separated into distinct populations that are updated separately. Often after a specified amount of generations the species will be returned into one total population.

In particle swarm optimisation it is common to use different *index based topologies*, that is, not Euclidean based, such that only individuals linked in the topology can com-

municate. With the correct topology, this process alone can be seen as a niching method. It has been found that the ring topology is able to maintain niches [82]. Of course, index based topologies can also be applied to other evolutionary algorithms. Clustering algorithms, such as K-means have also been used for niching [83]. The idea is to cluster the individuals, where each cluster is a niche.

Despite a reliability method like SuS having the same basic form as an evolutionary algorithm, there are some difficulties to consider when attempting to incorporate niching techniques. An evolutionary algorithm has a great amount of freedom with how samples are selected and created. In contrast, SuS has to generate samples that are distributed according to the chosen intermediate densities. This implies, that in order to maintain niches, somehow SuS should change which intermediate densities are chosen. A simple general strategy of this kind could be as follows: label all the samples of a level according to what niche they belong to with a niching technique and then adjust the level probability until at least one member of every niche has been selected as a seed.

An example of how this strategy could be employed is now described with a specific niching technique. *Nearest better clustering (NBC)* [84] is another niching technique that assigns a topology to the individuals using the Euclidean distance. First, the samples of a level are labelled

$$(\boldsymbol{x}_i)_{i=1}^N \quad \text{such that} \quad g(\boldsymbol{x}_i) \geq g(\boldsymbol{x}_{i+1}) \quad \text{for} \quad 1 \leq i \leq N-1. \tag{3.79}$$

Next the *NBC update map* is defined as follows,

$$\Psi^{\text{NBC}}(i) = \underset{1 \leq j \leq i}{\operatorname{argmin}} \|\boldsymbol{x}_i - \boldsymbol{x}_j\| \quad \text{for} \quad 2 \leq i \leq N. \tag{3.80}$$

Given a user-defined *niche radius* $r^{\text{NBC}} > 0$, the set of *NBC attractors* is defined as

$$\mathcal{A}^{\text{NBC}} = \{1\} \cup \{2 \leq i \leq N : \|\boldsymbol{x}_i - \boldsymbol{x}_j\| > r^{\text{NBC}} \text{ where } j = \Psi^{\text{NBC}}(i)\}. \tag{3.81}$$

Finally, each member of the level is assigned a label with the *NBC repeated update map*

$$\Psi_*^{\text{NBC}}(i) = \begin{cases} i & i \in \mathcal{A}^{\text{NBC}} \\ \Psi_*^{\text{NBC}}(\Psi^{\text{NBC}}(i)) & \text{otherwise.} \end{cases} \tag{3.82}$$

Notice the similarity of this definition to the definition of the BoAs of a reliability problem. Once the labels have been assigned to all samples of a level, the level probability that determines the next intermediate density should be increased until the corresponding intermediate failure region contains at least one seed with each label.

There are a few issues with this proposed approach. The most important is that it relies on the Euclidean metric with loses utility in high dimensions [85], making this algorithm unsuitable for a CRP. Additionally, the performance of this niching technique is highly sensitive to the choice of niche radius, which must be chosen by the user a priori. Finally, making the level probability larger will increase the mean number of levels required for a given reliability problem, which in turn increase the average number of performance function evaluations.

A reliability method that can perform well on a CRP will require a niching technique that does not depend on the Euclidean metric. Hill valley tests are a class of niching method that detect valleys in the objective function between two samples [86]. The idea is that two points belong to two separate niches if they exist on two hills of an objective

function, separated by a valley. Most hill valley tests are able to perform well in high dimensions, however, this comes at the cost of additional objective function evaluations. Possibly the simplest example of a hill valley test, and the most computationally cheap, is the following test defined for two samples $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$,

$$\mathrm{HVT}(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} 1 & \text{if } g_{\mathrm{obj}}\left(\frac{\boldsymbol{x}+\boldsymbol{y}}{2}\right) \geq \min(g_{\mathrm{obj}}(\boldsymbol{x}), g_{\mathrm{obj}}(\boldsymbol{y})), \\ 0 & \text{otherwise.} \end{cases} \tag{3.83}$$

Note that a 1 is returned when no valley is detected. This simple hill valley test is the fundamental component of the novel methods that will be presented in the following chapters.

4 Niching ratio methods

This chapter introduces novel algorithms that integrate niching techniques into ratio methods for reliability analysis. In particular, hill valley tests are used to increase the robustness of SuS when dealing with a CRPs. It should be noted that because SuS and SIS are such similar algorithms, the ideas presented here could be adapted for use with SIS as well. It should also be noted that like SuS and SIS, there is no fundamental reason why the techniques presented should be limited to reliability problems in SNS, like a first-order reliability method for example. So long as there exists a MCMC algorithm that can sample efficiently from the required intermediate densities, a reliability problem with any input density may be tackled. The choice to focus on SuS and SNS here has been made mostly for concreteness and ease of exposition.

Two approaches will be discussed in this chapter. The first approach, introduced in Section 4.1, is separated into a distinct niching stage and estimation stage. Section 4.2 explores how this method performs on the benchmarks. The second technique, presented in Section 4.3 interweaves the niching and estimation methods throughout the run of the algorithm.

4.1 Niching decomposition

This section introduces the niching decomposition subset simulation (NDSuS) algorithm. The idea of this algorithm is to use niching techniques to decompose a CRP into multiple reliability problems that are hopefully less challenging for SuS. Once this has been done, SuS may be used to solve the easier constituent reliability problem, and the probability of failure estimates may be combined to produce an estimate for the probability of failure for the CRPs. This is an attractive approach since SuS may be used with only very minor changes. This means existing implementations of SuS in different software packages may be used and also any enhancements or improvements that have been suggested for SuS in the literature may be applied directly to this algorithm. The niching stage is itself comprised of two stages: niching initial sampling (NInS) and smoothing.

4.1.1 Niching initial sampling

The purpose of an initial sampling procedure in reliability analysis is to locate the failure region. The most robust initial sampling procedure is to sample from the input distribution and to reject samples not in the failure region [87]. Of course, for small failure probabilities this approach will be inefficient. Though not explicitly a distinct initial sampler, the first step in [17] entails sampling from a scaled input distribution. This is only effective if the scaling is able to increase the probability of failure. The optimisation method in [88] can efficiently locate multiple modes of the failure density by adding a

bulge to the limit state function. However, it does require gradient information. The technique in [16] is gradient-free, though in the case of multiple niches, it requires prior expert knowledge regarding the performance function.

More recently, it is common to use an intermediate algorithm for the initial sampling. For instance, [53] uses SuS for initial sampling and [89] uses a density extrapolation approach combined with SuS [90]. As has been discussed previously, SuS can fail to populate all of the important regions of the input space. Another approach taken in [89], specifically used to deal with the meatball performance function, is to sample uniformly within some predetermined bounded area. This would not be a viable strategy for a reliability problem with a high-dimensional input space.

Before describing the NInS algorithm, a variant of the SuS algorithm is required. Given a subset of the input space called the *conditional set*, $A \subset \mathbb{R}^d$, let the *conditional probability of failure* be given as $\mathbb{P}(F|A)$. Such problems will be called *conditional reliability problems*. It is possible to compute such a probability with a slightly adjusted version of SuS which will be referred to as *conditional subset simulation (cSuS)*. To construct the product of ratios estimator, cSuS uses the following sequence of intermediate densities,

$$q_i^{\text{cSuS}}(\boldsymbol{x}) \propto \mathbb{1}_{F_i \cap A}(\boldsymbol{x}) f(\boldsymbol{x}) \text{ for } 0 < i < m, \tag{4.1}$$

$$q_0^{\text{cSuS}} \propto \mathbb{1}_A(\boldsymbol{x}) f(\boldsymbol{x}), \tag{4.2}$$

$$q_m^{\text{cSuS}} \propto \mathbb{1}_{F \cap A}(\boldsymbol{x}) f(\boldsymbol{x}). \tag{4.3}$$

To sample from these densities, two small adjustments are made to SuS. Firstly, the MCMC algorithm that is used at each level should reject candidate samples that lie outside of $F_i \cap A$, instead of only considering F_i. Secondly, some way of producing the *initial level*, that is level 0, such that the samples are distributed according to $f|A$ must be used. Different ways of producing such an initial level will be discussed later. Note that these definitions are really just a generalisation of a regular reliability problem and the standard SuS algorithm, which can be seen when $A = \mathbb{R}^d$.

The idea of this section is to rewrite the probability of failure in terms of conditional probability of failures with respect to some derived performance functions. That is, the target reliability problem gets decomposed into multiple, hopefully simpler, conditional reliability problems. This is an attractive approach since these conditional reliability problems may be estimated with the cSuS algorithm, which is essentially just the SuS algorithm.

The NDSuS algorithm is governed by two related heuristics. The first is that if no valley is detected between two points in the input space, with respect to a hill valley test defined by the performance function, then those two points are likely to be in the same BoA of the reliability problem. This will be useful for efficiently searching the input space for parts of the failure region. This is because hill valley tests that use failure samples that have already been located may be used to define regions of the input space that no longer need to be explored. This idea is central to the NInS algorithm. The second heuristic is that if no valley is detected between two failure samples, then they are likely to be in the same MSR of the failure density with respect to some MCMC algorithm. Of course, this heuristic can not be formally proven, because it always depends on precisely what MCMC algorithm is being used. This will be useful since the NDSuS algorithm explicitly attempts to model the number of MSRs a reliability problem has.

The decomposition of the reliability problem is done by identifying a set of points in the input space called *generators* $((\boldsymbol{y}_{ij})_{i=1}^{n_j})_{j=1}^{K}$, where $K \in \mathbb{N}$ will be the number of MSRs that NDSuS predicts the reliability problem has and $n_j \in \mathbb{N}$ is the number of generators required to model the ith MSR. How this is done will be discussed later, but for now it will be assumed they have been found. The generators have an important property that requires some definitions to establish. First, denote the *bounded performance function* as

$$\bar{g}(\boldsymbol{x}) := \min(g(\boldsymbol{x}), 0). \tag{4.4}$$

The bounded performance function is considered because the definition of the BoAs of a reliability problem is not influenced by the performance function in the failure. Put another way, the behaviour of SuS is not affected by the value of the performance function in the failure region, so long as the failure region is correctly defined. If the regular performance function were used instead in the following definitions, then valleys could be potentially identified in the failure which SuS has no problems traversing.

Let the *hill-valley performance functions* be given as,

$$g_{ij}^{\mathrm{HV}}(\boldsymbol{x}) := \bar{g}((\boldsymbol{x} + \boldsymbol{y}_{ij})/2), \tag{4.5}$$

for $1 \leq i \leq n_j$ and $1 \leq j \leq K$. That is, the performance function has been transformed into new performance functions that are re-centred around the generators using a hill valley test. The *joint hill-valley performance functions* are defined as

$$g_j^{\mathrm{JHV}}(\boldsymbol{x}) := \max_{1 \leq i \leq n_j} g_{ij}^{\mathrm{HV}}(\boldsymbol{x}). \tag{4.6}$$

for $1 \leq j \leq K$. Now each joint hill-valley performance function has an associated failure region

$$\mathcal{F}_j^{\mathrm{JHV}} := \{\boldsymbol{x} \in \mathbb{R}^d : g_j^{\mathrm{JHV}}(\boldsymbol{x}) \geq 0\}. \tag{4.7}$$

for $1 \leq j \leq K$. Finally, if the generators have been chosen appropriately, they should imply the following property,

$$\mathbb{P}(\mathcal{F}) \approx \sum_{j=1}^{K} \mathbb{P}(\mathcal{F}_j^{\mathrm{JHV}}) \mathbb{P}(\mathcal{F} | \mathcal{F}_j^{\mathrm{JHV}}). \tag{4.8}$$

That is the, probability of failure may approximately decomposed into $2K$ conditional probabilities of failure where K of them are conditional on the entire input space.

The generators must be chosen such that this approximate decomposition can be made and so that it is likely that the resulting conditional reliability problems are easier for SuS than the original reliability problem. The intention is that conditional reliability problem has a failure density with only one MSR and ideally no safe BoAs that the SuS algorithm can get trapped in. If these properties can be ensured, it is much less likey that SuS will adaptively choose a sequence of intermediate densities that are intermediate multimodal, thus avoiding the associated challenges,

The purpose of the NInS algorithm is, given a reliability problem, to locate a suitable set of generators, according to above criteria. Importantly, it does no require any other information regarding the performance function aside from input-output information and its computational cost is largely independent of the dimension of the reliability problem. The NInS algorithm requires a few important definitions. Given a performance function g and a generator $\boldsymbol{y} \in \mathbb{R}^d$, define a *hill valley basin of attraction (HVBoA)* as

$$\mathcal{H}(\boldsymbol{y}, g) := \{\boldsymbol{x} \in \mathbb{R}^d : \bar{g}(\boldsymbol{x}) \leq \min\left(\bar{g}((\boldsymbol{x} + \boldsymbol{y})/2)\right), \bar{g}(\boldsymbol{y}))\}. \tag{4.9}$$

The purpose of a HVBoA is to model the BoA that $\boldsymbol{y}$ lies inside, where y is assumed to be some sort of approximation of an attractor. Given a performance function g and a finite set of generators, $\mathcal{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n\}$, define the *non-admissible region* as

$$\mathcal{H} := \bigcup_{\boldsymbol{y} \in \mathcal{Y}} \mathcal{H}(\boldsymbol{y}, g). \tag{4.10}$$

Let the *admissible region* be denoted as $\mathcal{H}^c := \mathbb{R}^d \backslash \mathcal{H}$. The admissible region will be useful, since give the generators, which represent attractors, that have already been located, the admissible region should approximately contain all the attractors that have not yet been located.

The NInS algorithm constitutes three components: *seed generation*, *Markov chain optimisers (MCOs)* and *generator location*. All three components depend on the currently located generators $\mathcal{Y}$ and the resulting admissible region $\mathcal{H}^c$, which are updated iteratively as the algorithm proceeds. At the start of the algorithm, the set generators is empty, that is $\mathcal{Y} = \emptyset$ and so $\mathcal{H}^c = \mathbb{R}^d$. The idea of the seed generation algorithm is to simply sample from the input distribution until a sample is generated in the admissible region. Sometimes the high-density region of the input distribution may lie entirely inside the non-admissible region, despite there still being important regions of the failure density to populate. For this reason, after every failed attempt at generating a seed in the the admissible region, an increasing amount of noise is added to the input distribution for the next attempt. If, after some fixed number attempts, the algorithm still can not produce a sample in the admissible region, then it is assumed that the admissible region covers the entire input space and the NInS algorithm is stopped. Seed generation is formally described by the following steps:

Seed generation:

1. Set $\sigma_{\text{noise}} = 0$ and $c_{\text{rej}} = 0$.
2. Let n_{rej} be a user-defined parameter called the *rejection limit*. If $c_{\text{rej}} > n_{\text{rej}}$ stop the algorithm and return no seed.
3. Sample $\boldsymbol{x} \sim f$ and $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}_d, \boldsymbol{I}_d)$. Let $\boldsymbol{x}' = \boldsymbol{x} + \sigma_{\text{noise}} \cdot \boldsymbol{z}$.
4. If $\boldsymbol{x}' \in \mathcal{H}^c$ then terminate the algorithm and return $\boldsymbol{x}'$ as a seed.
5. Update and $c_{\text{rej}} \leftarrow c_{\text{rej}} + 1$ and $\sigma_{\text{noise}} \leftarrow \sigma_{\text{noise}} + \sigma_{\text{step}}$ where σ_{step} is a user-defined parameter. Go to Step 2.

Once a seed is generated, it is used to start a MCO, which consists of two steps. First, chain of length n_s is created using a MCMC algorithm with stationary distribution $f|\mathcal{H}^c$ starting at the seed. This single chain is then then used as an initial level of a cSuS algorithm with $\mathcal{H}^c$ as the conditional set and in which every level only consists of one chain, that is $N_{\text{L}} = p_{\text{L}}^{-1} = n_s$. The resulting associated conditional probability of failure estimate will be very poor for two reasons. Firstly each level is only constructed using one chain. Secondly, the seed used for the initial level may have had a lot of noise applied to it, and so will not be distributed according to $f|\mathcal{H}^c$. This is not a problem, because cSuS is not being used here to estimate the conditional probability of failure, but is instead being used to locate the failure region, hence the name of a MCO.

Once a MCO has terminated, generator location takes place. If the MCO never created any failure samples, then the sample with the highest performance from the the last level of the cSuS algorithm, say $\boldsymbol{y}$, is added to list of generators, $\mathcal{Y} \leftarrow \mathcal{Y} \cup \boldsymbol{y}$. This will be referred to as a safe generator. A safe generator may be thought of as approximating the attractor of a safe BoA. Otherwise, the failure sample that was last to be generated during cSuS is selected as a seed. This seed will be used with a MCMC algorithm to construct a chain, referred to as a *generator chain*, of length n_s^{gen} with the failure density as its stationary distribution, where n_s^{gen} is a user-defined parameter called the *generator chain length*. Because of the noise added in the seed generation procedure, there is no guarantee that the seed is distributed according to the failure density, and so some burn-in phase may be required. To account for this, the first half of the generator chain is discarded.

The generator chain is now randomly sampled from with replacement n_{rep} times, where n_{rep} is a user-defined parameter called the representative number. These samples are meant to serve as an approximation of the entire generator chain. The reason this is done, instead of simply considering the entire chain is because it will reduce the amount of performance function evaluations required later. The resulting samples are added to a set called the representatives, denoted by $\mathcal{R}$, which is the empty set at the start of NInS procedure. Now the set $\mathcal{R} \setminus \mathcal{H}$ is considered. This set indicates the parts of the failure region that are not currently covered by the non-admissible region. If it is empty then the NInS starts again from the seed generation phase. If not, a member of this set is randomly selected, say $\boldsymbol{y} \in \mathcal{R} \setminus \mathcal{H}$, and then added to the set of representatives, $\mathcal{Y} \leftarrow \mathcal{Y} \cup \boldsymbol{y}$. This process is repeated until $\mathcal{R} \setminus \mathcal{H}$ is empty.

Consider a graph called the *joint hill valley graph*, denoted by G^{JHV}, where the vertex set is $\mathcal{Y}$. Put an edge between two vertices $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ if

$$\mathcal{H}(\boldsymbol{y}_1, g) \cap \mathcal{H}(\boldsymbol{y}_2, g) \cap \mathcal{R} \neq \emptyset. \tag{4.11}$$

The control flow of the algorithm is mainly governed by a user-defined parameter called the *discover limit*, n_{dsc}, which regulates the number of consecutive iterations the algorithm can make without increasing the number of connected components of G^{JHV}. To make the control flow of the NInS procedure more clear, it is now summarised.

Niching initial sampling:

1. Set $c_{\text{dsc}} = 0$.
2. If $c_{\text{dsc}} = n_{\text{dsc}}$ then terminate the algorithm.
3. Generate seed $\boldsymbol{x}$ with the seed generation procedure. If no seed is returned then terminate the algorithm.
4. Run a MCO using the seed $\boldsymbol{x}$ and with $\mathcal{H}^c$ as the conditional set.
5. If the MCO produces failure samples, carry on. Otherwise, take the sample with the highest performance from the MCO, say $\boldsymbol{y}$, and add it to the generators, $\mathcal{Y} \leftarrow \mathcal{Y} \cup \boldsymbol{y}$. Update $c_{\text{dsc}} \leftarrow c_{\text{dsc}} + 1$. Go to Step 2.
6. Use the last failure sample to be generated by the MCO as a seed to create the generator chain with an MCMC algorithm of length n_s^{gen}. Discard the first half of the generator chain.

7. Randomly sample n_{rep} times from the generator chain add the result to the set of representatives,
$$\mathcal{R} \leftarrow \mathcal{R} \cup \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n_{\text{rep}}}\}. \tag{4.12}$$

8. Update $c_{\text{dsc}} \leftarrow c_{\text{dsc}} + 1$.

9. If $\mathcal{R} \setminus \mathcal{H} = \emptyset$ then go to Step 2.

10. Randomly select $\boldsymbol{y} \in \mathcal{R} \setminus \mathcal{H}$, and then update set of representatives, $\mathcal{Y} \leftarrow \mathcal{Y} \cup \boldsymbol{y}$. If the number of connected components of G^{JHV} increases after this addition, then set $c_{\text{dsc}} = 0$. Go to Step 9.

From this point on, after a NInS procedure has terminated, it will be assumed that all the safe generators have been removed from $\mathcal{Y}$. This is because the generators will now be used to model the failure region, and so they are not required. The central idea behind this algorithm is that $\mathcal{H}$ has been constructed in such a way that it is impossible to produce a sample from the failure density with a MCMC algorithm that does not lie in $\mathcal{H}$, regardless of starting seed. Due to this construction, it is reasonable to assume that

$$\mathbb{P}(\mathcal{F}) \approx \mathbb{P}(\mathcal{F} \cap \mathcal{H}). \tag{4.13}$$

This is useful as now $\mathcal{H}$ may be used to in some way approximate the failure region, or more precisely, the high density parts of the failure region.

It is possible to decompose $\mathcal{H}$ into smaller sets. Assuming that G^{JHV} has K connected components where the jth component has n_j vertices, label the generators as $((\boldsymbol{y}_{ij})_{i=1}^{n_j})_{j=1}^{K}$ such that $\boldsymbol{y}_{ij}$ and $\boldsymbol{y}_{i'j}$ belong to the same connected component for all i, i' and j. The heuristic mentioned above, that failure samples with no valley in-between them belong to the same MSR, motivates this definition. If this heuristic is true, then the generators should now be sorted according to which MSR they belong to.

By definition, it is true that

$$\bigcup_{j=1}^{K} \bigcup_{i=1}^{n_j} \mathcal{H}(\boldsymbol{y}_{ij}, g) = \mathcal{H}. \tag{4.14}$$

Also note, by the definitions given in Equations 4.7, the following also holds,

$$\bigcup_{i=1}^{n_j} \mathcal{H}(\boldsymbol{y}_{ij}, g) \cap \mathcal{F} = \mathcal{F}_j^{\text{JHV}} \cap \mathcal{F}. \tag{4.15}$$

for $1 \leq j \leq K$. Equation 4.11 implies that the joint hill-valley failure regions should have very little overlap in the failure region. This implies that the following should be approximately true

$$\mathbb{P}(\mathcal{F}_j^{\text{JHV}} \cap \mathcal{F}_{j'}^{\text{JHV}} \cap \mathcal{F}) \approx 0, \tag{4.16}$$

for $1 \leq j < j' \leq K$. Combining all these statements allow for the required decomposition

to be made

$$\mathbb{P}(\mathcal{F}) \approx \mathbb{P}(\mathcal{F} \cap \mathcal{H}) \tag{4.17}$$

$$= \mathbb{P}\left(\mathcal{F} \cap \bigcup_{j=1}^{K} \bigcup_{i=1}^{n_j} \mathcal{H}(y_{ij}, g)\right) \tag{4.18}$$

$$= \mathbb{P}\left(\mathcal{F} \cap \bigcup_{j=1}^{K} \mathcal{F}_j^{\mathrm{JHV}}\right) \tag{4.19}$$

$$\approx \sum_{j=1}^{K} \mathbb{P}\left(\mathcal{F} \cap \mathcal{F}_j^{\mathrm{JHV}}\right) \tag{4.20}$$

$$= \sum_{j=1}^{K} \mathbb{P}(\mathcal{F}_j^{\mathrm{JHV}}) \mathbb{P}(\mathcal{F} | \mathcal{F}_j^{\mathrm{JHV}}). \tag{4.21}$$

There are some small implementation details of NInS algorithm to consider. Firstly, when evaluating the membership of the admissible region, care should be taken so that the procedure is not needlessly computationally expensive. That is, each generator should be considered in turn, and if at any point no valley is detected, there is no need to carry on with the rest of the generators. This should be done since each hill valley test requires the evaluation of the performance function. The same logic applies to evaluating the joint hill valley performance functions for determining the membership of the intermediate failure regions. It is possible for the algorithm to never produce a failure sample, and so the above decomposition may not be able to be made. To avoid this, a condition could be added such the algorithm keeps on running until a failure sample is produced, regardless of other stopping conditions. However, in the end this may just be a sign that the true probability of failure is actually zero.

4.1.2 Smoothing

The next stage of NDSuS is smoothing. The principal idea behind smoothing is to attempt to remove safe BoAs from the conditional reliability problems. Given a joint hill valley performance function, denoted by g^{JHV}, and associated failure region, denoted by $\mathcal{F}^{\mathrm{JHV}}$, smoothing creates a new joint hill valley performance function, g_*^{JHV}, and associated failure region, $\mathcal{F}_*^{\mathrm{JHV}}$, such that

$$\mathbb{P}(\mathcal{F}^{\mathrm{JHV}}) \approx \mathbb{P}(\mathcal{F}_*^{\mathrm{JHV}} \cap \mathcal{F}^{\mathrm{JHV}}), \tag{4.22}$$

$$= \mathbb{P}(\mathcal{F}_*^{\mathrm{JHV}}) \mathbb{P}(\mathcal{F}^{\mathrm{JHV}} | \mathcal{F}_*^{\mathrm{JHV}}). \tag{4.23}$$

That is, $\mathcal{F}_*^{\mathrm{JHV}}$ contains the high density region of $\mathcal{F}^{\mathrm{JHV}}$. In some sense, the smoothing procedure represents a type of backwards SuS, that starts at the failure region and creates a sequence of subsets that increasingly have a higher probability with respect to the input density.

The smoothing process consists of two parts. The first checks if the smoothing is necessary, and then if it is, the second creates the new performance function. The checking

procedure is done by running SuS n_{check} times, where the number of checks is a user-defined parameter. Each SuS run uses g^{JHV} as its performance function and each level consists of only one chain. This step could be interpreted as using MCOs where the entire input space is the conditional set. If any of the SuS runs do not populate the failure region, then the second step is performed, and otherwise it is not.

The second step of the smoothing process is very similar to the seed location procedure used during NInS. To begin, one of the generators of g^{JHV} is selected at random as a seed. This seed is the used by a MCMC algorithm to generate a generator chain of length n_s^{gen} with stationary distribution $f|\mathcal{F}^{\text{JHV}}$. The first half of the chain is discarded to account for the burn-in phase. Then, n_{rep} samples are randomly chosen and are called collectively the set of representatives, denoted as $\mathcal{R}$. Let the set of generators, $\mathcal{Y}$, begin as the empty set and define the associated non-admissible set $\mathcal{H}$. Whilst $\mathcal{R} \setminus \mathcal{H} \neq \emptyset$, randomly add samples from $\mathcal{R}$ to $\mathcal{Y}$. When this process is complete, use all of the generators in $\mathcal{Y}$ to define a *smoothed joint hill valley performance function*, g_*^{JHV}, and associated failure region $\mathcal{F}_*^{\text{JHV}}$. The approximation used in Equation 4.22 is justified by the process of construction.

The entire smoothing process can now be applied to the smoothed joint hill valley performance function, and so on, until the checking procedure determines that no more smoothing is required. After NInS has completed, the smoothing process is carried out on each of the $\mathcal{F}_j^{\text{JHV}}$ for $1 \leq j \leq K$. The smoothing process is now summarised.

Smoothing:

1. Set $j = 1$.
2. If $j > K$, terminate the algorithm.
3. Set $g^{\text{JHV}} \leftarrow g_j^{\text{JHV}}$ and $\mathcal{F}^{\text{JHV}} \leftarrow \mathcal{F}_j^{\text{JHV}}$.
4. Run SuS n_{check} times with $N_{\text{L}} = p_{\text{L}}^{-1}$ and performance function g^{JHV}. If all SuS runs populate the failure region, update $j \leftarrow j + 1$ and go Step 2. Otherwise, carry on.
5. Randomly select a generator of g^{JHV}, use it as a seed with a MCMC algorithm to create a generator chain of length n_s^{gen} with stationary distribution $f|\mathcal{F}^{\text{JHV}}$. Discard the first half of the chain.
6. Randomly select n_{rep} samples from the generator chain and denote that collection $\mathcal{R}$. Set the set of generators as $\mathcal{Y} = \emptyset$ and denote the associated non-admissible region as $\mathcal{H}$.
7. If $\mathcal{R} \setminus \mathcal{H} \neq \emptyset$ then carry on. Otherwise, use the generator $\mathcal{Y}$ to define smoothed joint hill valley performance function, g_*^{JHV}, and associated failure region $\mathcal{F}_*^{\text{JHV}}$. Update $g^{\text{JHV}} \leftarrow g_*^{\text{JHV}}$ and $\mathcal{F}^{\text{JHV}} \leftarrow \mathcal{F}_*^{\text{JHV}}$. Go to Step 4.
8. Randomly select $\boldsymbol{y} \in \mathcal{R} \setminus \mathcal{H}$, and then update set of representatives, $\mathcal{Y} \leftarrow \mathcal{Y} \cup \boldsymbol{y}$. Go to Step 7.

For the jth joint hill valley performance function, the smoothing procedure generates a sequence of s_j new smoothed performance functions and associated failure regions. These new failure regions are labelled as $\mathcal{F}_i^{(j)}$ for $1 \leq i \leq s_j$ in reverse order of creation. That

is, $\mathcal{F}_1^{(j)}$ was the last to be created and $\mathcal{F}_{s_j}^{(j)}$ was the first. Let $\mathcal{F}_0^{(j)} = \mathbb{R}^d$, $\mathcal{F}_{s_j+1}^{(j)} = \mathcal{F}_j^{\text{JHV}}$ and $\mathcal{F}_{s_j+2}^{(j)} = \mathcal{F}$. Let $g_i^{(j)}$ be the associated joint hill valley performance functions for $1 \leq j \leq K$ and $1 \leq i \leq s_j + 1$. By combining the approximations made in Equation 4.8 and in Equation 4.22, the following approximation may be made for the probability of failure

$$\mathbb{P}(\mathcal{F}) \approx \sum_{j=1}^{K} \prod_{i=0}^{s_j+1} \mathbb{P}\left(\mathcal{F}_{i+1}^{(j)} | \mathcal{F}_i^{(j)}\right) \tag{4.24}$$

Each conditional probability of failure, $\mathbb{P}\left(\mathcal{F}_{i+1}^{(j)} | \mathcal{F}_i^{(j)}\right)$ may be estimated using the cSuS algorithm with performance function $g_{i+1}^{(j)}$ and conditional set $\mathcal{F}_i^{(j)}$. This results in the following estimators,

$$\mathbb{P}\left(\mathcal{F}_{i+1}^{(j)} | \mathcal{F}_i^{(j)}\right) \approx \hat{P}_i^{(j)} \tag{4.25}$$

for $1 \leq j \leq K$ and $0 \leq i \leq s_j+1$. Combining these estimators gives the NDSuS estimator for the probability of failure,

$$P_{\mathcal{F}} \approx \hat{P}_{\mathcal{F}}^{\text{NSuS}} := \sum_{j=1}^{K} \prod_{i=0}^{s_j+1} \hat{P}_i^{(j)} \tag{4.26}$$

The only issue that remains is to create an initial level for each of the cSuS runs with the correct distribution. Note that for the cSuS runs associated with the estimators $\hat{P}_0^{(j)}$ for $1 \leq j \leq K$, the conditional set is the input space and the initial level may be simply sampled from the input distribution. Also note that if cSuS has just been ran targetting the conditional probability of failure $\mathbb{P}\left(\mathcal{F}_{i+1}^{(j)} | \mathcal{F}_i^{(j)}\right)$, then the final level should contain samples distributed according to $f|\mathcal{F}_{i+1}^{(j)}, \mathcal{F}_i^{(j)}$. These samples should also be approximately distributed according to $f|\mathcal{F}_{i+1}^{(j)}$. And so, if these samples are resampled uniformly at random with replacement n_c times, they can be used as seeds to generate Markov chains of length n_s with stationary distribution $f|\mathcal{F}_{i+1}^{(j)}$. This is then precisely the initial level required for the cSuS targetting $\mathbb{P}\left(\mathcal{F}_{i+1}^{(j)} | \mathcal{F}_i^{(j)}\right)$. In this way the final level of one cSuS can be used to construct the initial level of the next. One way of interpreting this is that there are K runs of SuS like algorithms, that each consist of $s_j + 1$ cSuS algorithms that are linked together. The final NDSuS is now summarised.

Niching decomposition subset simulation:

1. Run the NInS procedure.
2. Run the smoothing procedure using the output of the NInS procedure.
3. Set $j = 1$.
4. If $j > K$, terminate the algorithm.
5. Set $i = 1$. Sample initial level, L_0, of size N_{L} from the input distribution.
6. Run cSuS with initial level L_0, conditional set $\mathcal{F}_i^{(j)}$ and performance function $g_{i+1}^{(j)}$. Update $i \leftarrow i + 1$.
7. If $i > s_j + 1$, Update $j \leftarrow j + 1$ and go to Step 4.

8. Generate initial level L_0 of size N_{L} distributed according $f|\mathcal{F}_i^{(j)}$ using final level of previous cSuS and a MCMC algorithm. Go to Step 6.

All MC reliability methods make some assumptions about the reliability problem in order to reduce the variance of their estimator in comparison to the MC estimator for the probability of failure. The NDSuS estimator is no different. In particular, NDSuS is implicitly assuming a couple of different important properties. Firstly it is assuming that the number of MSRs of the failure density is relatively low. If this were not the case, then a large number of generators would be required to model the failure region accurately and a large number of SuS runs would be required to accurately estimate the probability of failure. In this case, the resulting algorithm would be so computationally expensive that the MC estimator may as well be used instead. Additionally, it is being assumed that the hill valley test can be used to accurately model each MSR. If this is not the case, the algorithm may split up one MSR into multiple different reliability problems, again wasting computational resources.

Fortunately, in both of these cases, the algorithm can to some extent determine when these assumptions are being violated and the user can then act on this information. For example, if the NInS algorithm identifies too many generators then the algorithm could stopped. If it is ever determined that different samples from one generator chain belongs to two separate groups of generators then the algorithm may also be stopped. This second condition is considered in the implementation that is used in the numerical examples in this work. When this condition is triggered, the algorithm reverts to just simply running SuS. This condition is only ever triggered by the loss performance function, due to its complex geometry that is difficult for the generators to model accurately.

As will be seen in the next section, the NDSuS algorithm is able to perform extremely well the on the selected benchmarks. Indeed, the general strategy of running some type of process that decomposes the reliability problem in simpler reliability problems is attractive, since those simpler reliability problems may be tackled by existing methods, as has been done here with SuS. Additionally, the hill valley test seems to be a powerful general tool for reliability analysis since it does not deteriorate in higher dimensions like the Euclidean metric. However, the separate NInS and smoothing stages are slightly awkward, could be easier to implement and introduce quite a few user-defined parameters. Ideally, the NInS procedure and smoothing procedure could be combined into one more streamlined algorithm. This would be an interesting topic for future work.

One alternative approach that was attempted during the course of this work was to consider the all hill valley performance functions simultaneously as is done in generalised subset simulation [91, 92]. The idea is that each intermediate failure region is the union of intermediate failure regions defined by the hill valley performance functions. This approach has a somewhat simplified control flow, since only one run of SuS is ultimately needed and there is no need to sort the generators into groups. However, this strategy would require all of the hill valley performance functions to be evaluated for all of the samples, even when a sample is in part of the input space that is only relevant to one of the hill valley performance functions. In contrast NDSuS, by grouping the generators and running separate SuS runs for each group, is able to more efficiently evaluate the hill valley performance functions. One aspect of NDSuS that could be explored more is that some of the MSRs that have been modelled may be very low density and thus will not contribute much to the overall probability of failure. This NDSuS has the potential to waste an entire run of SuS on a region of the failure density that is not important. Unfortunately, there is no way of knowing this before the SuS run is started. One strategy

could be to update a level of each of the K sequences of cSuS runs at a time, rather than completing one of them fully and then moving onto the next. In this way, when one of the sequences reaches the failure region, the others may all be stopped as well, in order to prevent further irrelevant levels.

4.2 Numerical Examples

In this section, the performance of the NDSuS algorithm on the benchmarks is analysed. Before discussing the overall results of NDSuS on the benchmarks, its behaviour on specific reliability problems is explored in order to give a deeper understanding of how it works. First the piecewise linear function is considered in depth, and then the black swan performance function.

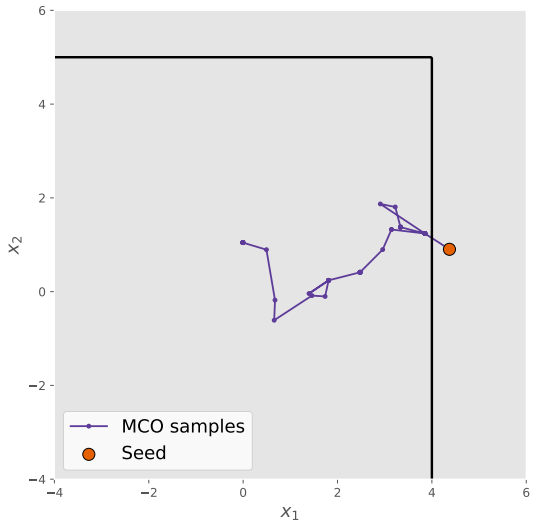
4.2.1 Piecewise linear function

Recall that the piecewise linear function has two BoAs, where one contains a high density MSR of the failure density and the other contains a low density MSR of the failure density. The original SuS algorithm struggles to consistently populate the high density MSR due to the misleading geometry.

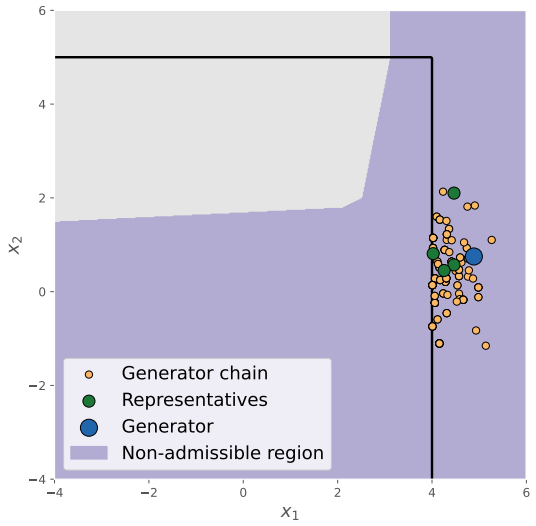
Figure 4.1 shows a run of NInS on the piecewise linear function. Since the admissible region is the entire input space at the start of the algorithm, the first seed that is suggested by the seed generation process is automatically accepted with no noise applied to it. This seed is then used to run a MCO which is shown in Figure 4.1a. This MCO manages to reach the failure region and so a seed is selected to create a generator chain. The generator chain is then created, the representatives are chosen, and the generators are chosen from those representatives. As can be seen in Figure 4.1b, on this occasion only one generator is required in order to model the MSR. This is due to the relatively simple geometry of the piecewise linear function.

Once the first generator has been selected, the non admissible region is no longer the empty set, but instead attempts to model the basin of attraction that the generator lies in, as can be seen in 4.1b. Now seed generation begins again. This time, some of the seeds may be rejected since the non-admissible region covers a large part of the high density region of the input density. As a result, noise is added until a sample outside of this region can be produced. Once this sample is produced, it is used to run the second MCO, which is shown in Figure 4.1c. Note that the MCO never steps outside of the admissible region due to it being a cSuS run with its conditional set as the admissible region. Again, this MCO manages to reach to failure region and so a new seed is selected for the generator chain. Using that seed a new generator chain is sampled as shown in Figure 4.1d. Note that the seed on this occasion happens to be quite far from the high density region of the MSR. This is why the burn-in phase is required for the generator chain. Now new representatives are randomly sampled, and again, only one generator is required to model the MSR. Now the non-admissible region is the entire input space, and so it is impossible for the seed generation algorithm to generate a seed. After n_{rej} attempts at doing so, the entire NInS algorithm is terminated.

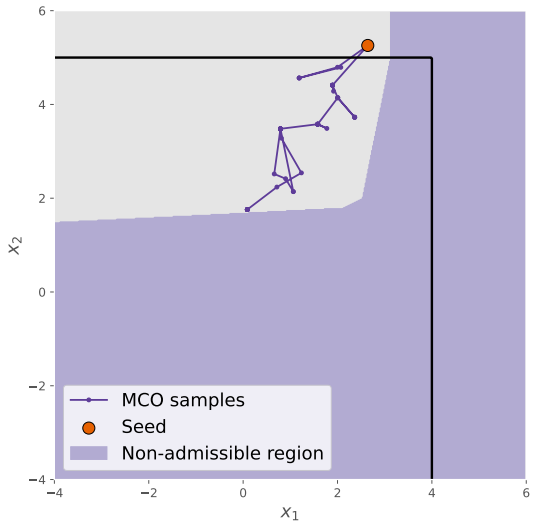
On this particular occasion, no smoothing is required. This is determined by the NDSuS by running SuS on the newly created joint hill valley performance functions that are defined by the generators that were found by NInS. Since SuS is able to reach the



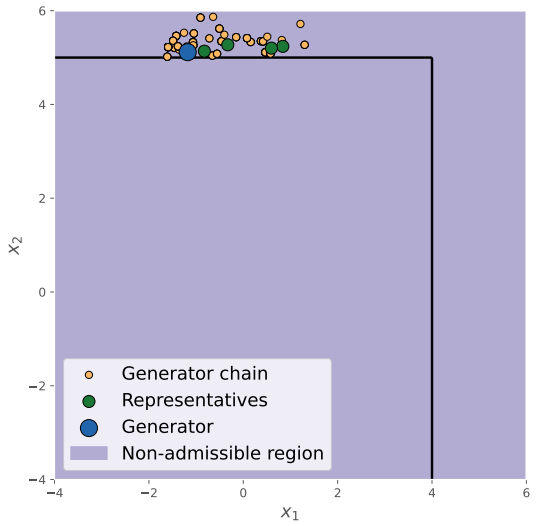
(a) First MCO.



(b) Locating the first generator.



(c) Second MCO.



(d) Locating the second generator.

Figure 4.1: A run of NInS on the piecewise linear function.

failure regions, the smoothing procedure is not used. This is quite typical for the the piecewise linear function, because it has no safe BoAs to begin with. However, despite this, it is possible that the newly created conditional reliability problems do have safe BoAs which tend to have to be smoothed over.

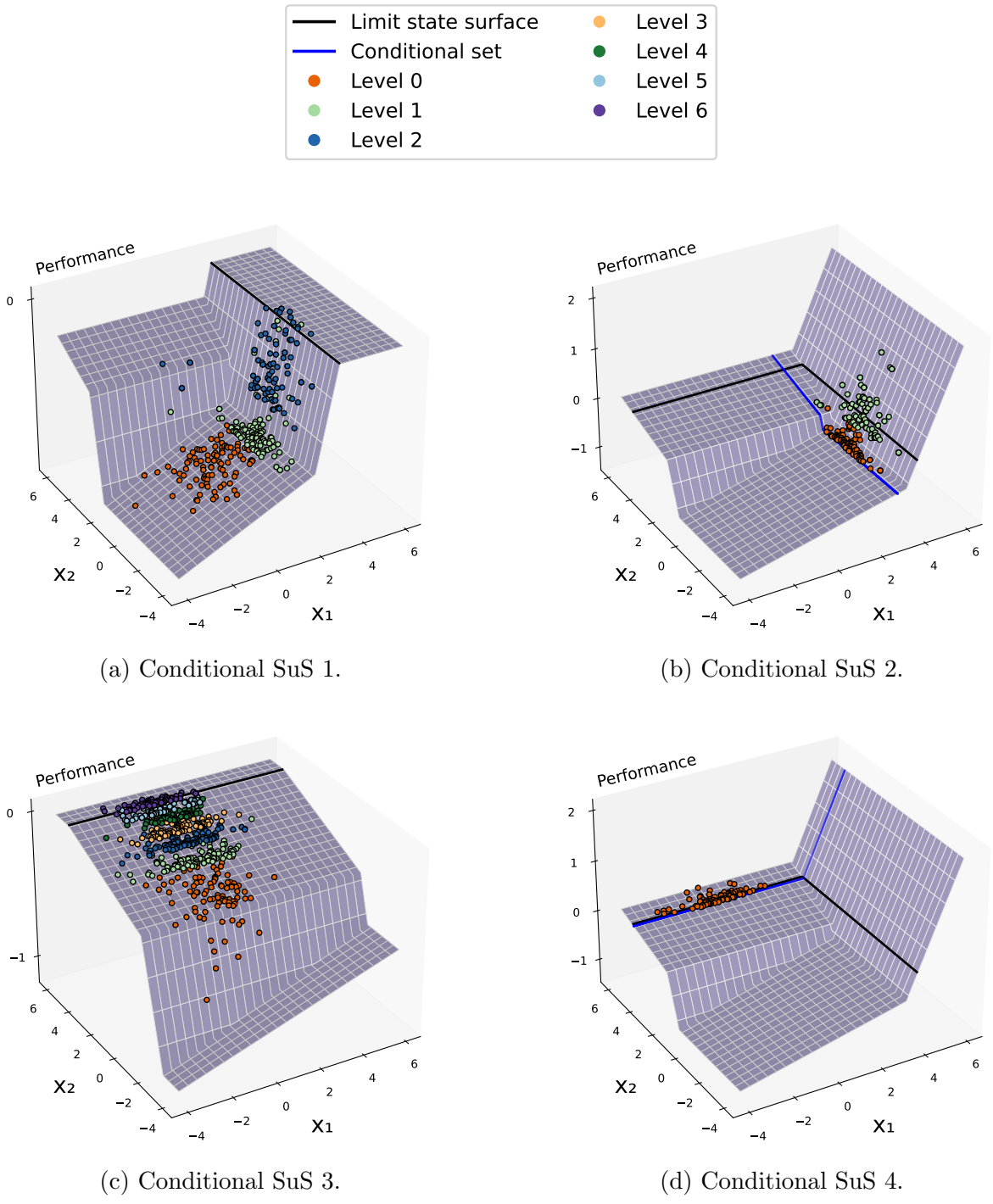
Figure 4.2 shows the four runs of cSuS on the four conditional reliability problems the NInS procedure has decomposed the piecewise linear function into. Figure 4.2a shows the first such run. Note that the conditional set in the first run of a sequence is simply the entire input space, so it may be treated like a regular reliability problem. This conditional reliability problem is associated with the high density MSR. In this case, the geometry of this new reliability problem guides the cSuS samples in the desired direction. Figure 4.2b shows the second and final conditional reliability problem in this sequence. Note that the performance function is now the piecewise linear function itself. The final conditional reliability problem always use the original performance function. However, this is conditional reliability problem, and so cSuS is restricted to the failure region of the previous performance function. The high density region of the failure density is successfully populated by the samples.

Figure 4.2c starts a new sequence of conditional reliability problems, this time associated with the second generator that was located by the NInS procedure. Now it can be seen that the cSuS are guided by the geometry of the joint hill valley performance function in other direction. Finally, Figure 4.2d shows how the low density MSR of the failure density becomes populated. To estimate the probability of failure the two estimates from the two sequences are added together, which themselves are each products of two estimates. The sequence associated with the low density MSR produces a negligible estimate compared to the estimate of the sequence associated with the high density MSR. Ideally there would be some way of knowing this a priori, since the a priori, since the entire sequence is not really required. On the other hand, if a practitioner wants to know the individual probabilities of all the MSRs for some practical reason, then this sequence would still be useful.

4.2.2 Black swan function

Recall that the black swan function has two BoA, one safe and one failure, and that its failure density has one MSR. Figure 4.3 shows a run of NInS on the black swan function. The first MCO is shown in Figure 4.3a. It is important to note that the this MCO does not manage to reach the failure region. This is because the safe BoA dominates the the high density part of the input density. In this case, the seed is chosen as the sample with the highest performance. However, as shown in Figure 4.3b, no chain is generated from this seed. Instead it is used as a safe generator. Note how important it is that the HVBoA is defined in such a way that the HVBoA of a safe generator never contains any of the failure region. The non-admissible region now covers all of the high density region of the input space.

Figure 4.3c shows the run of the second MCO. Notice that a lot of noise had to be applied in the seed generation step in order to escape form the non-admissible region in this case. This MCO is able to reach the failure region and produce a failure sample seed. Figure 4.3d shows the generator chain, representatives and the selected generator that models the MSR of the failure density. Finally, the non-admissible region covers the entire input space and so the no more seeds can be generated, and the algorithm is terminated. Note that now going forward, only the failure generator will be considered



(a) Conditional SuS 1.

(b) Conditional SuS 2.

(c) Conditional SuS 3.

(d) Conditional SuS 4.

Figure 4.2: Runs of conditional SuS on the four conditional reliability problems that the piecewise linear problem has been decomposed into.

whilst the safe generator is discarded.

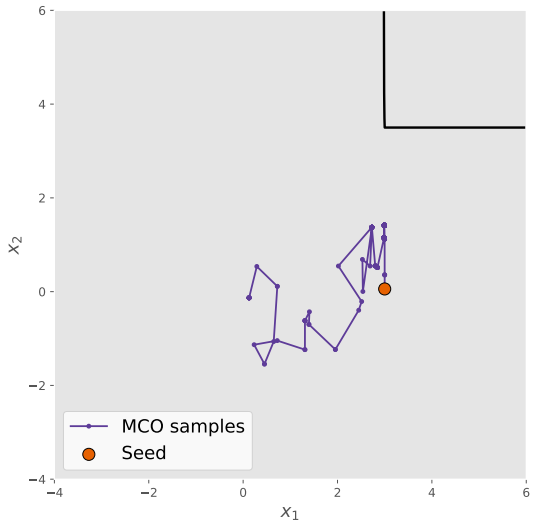
In this case, smoothing will almost certainly be required. This is due to the safe BoA of the black swan function. A checking runs of SuS will be made and the failure region of new joint hill valley performance function will likely not be populated. At this point, the smoothing procedure begins. The generator chain is ran and a new joint hill valley performance function is found. This process repeats until the checks are passed. Figure 4.4 shows how this process looks for the black swan function, where three rounds of smoothing have taken place in this particular run. Notice that the new failure regions, which are coloured to match the corresponding generator chains, progressively approach the high density region of the input space. That is, they should become easier to sample from after successive rounds of smoothing.

Figure 4.5 shows the runs of cSuS on the conditional reliability problems that result from the the NInS and smoothing procedure. In this case there is one sequence of four conditional reliability problems. Figure 4.5a shows the first conditional reliability problem, in which most of the progress is made towards the failure region of the original reliability problem. The next two cSuS runs are shown in Figures 4.5b and 4.5c. Finally, the failure region of the black swan performance function is populated in Figure 4.5d.

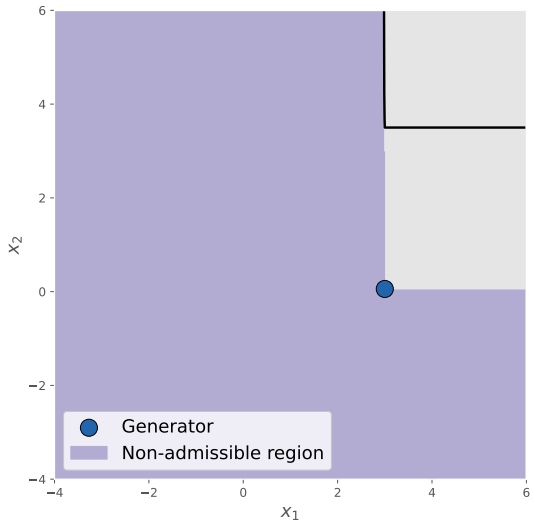
Table 4.1 shows the values of the user-defined parameters used for NInS and table 4.2 shows the rest of the parameters used for NDSuS for the numerical experiments on the benchmarks. Note that the pCN algorithm was used through out as the MCMC algorithm. The results of the numerical examples are shown in Table 4.3. These results show that NDSuS has a very robust performance on the benchmark functions, with the NDP being 100% for nearly all of the experiments. Note that for both the low and high dimensional version of the black swan performance function 1 out of 100 of the NDSuS estimators are degenerate. Despite this not being ideal, NDSuS does offer something in these cases. This is because in both these cases, the failure region is not reached by the cSuS algorithms, despite the NInS algorithm finding the failure region. This lets the user know that something must have gone wrong. That is, in this case, NDSuS is not silently degenerate. This is one of the advantages of explicitly modelling the failure region. Having been alerted to this problem, a user could takes some action, such as running more smoothing steps. Note that, as expected, the number of performance function evaluations does not increase with the dimension of the problem. This makes NDSuS a suitable algorithm for very high dimensional problems.

Parameter	Value
Proposal scale (σ_{p})	0.5
Generator chain length (n_s^{gen})	400
Discover limit (n_{dsc})	3
Representative number (n_{rep})	5
Rejection limit (n_{rej})	100
Noise step (σ_{step})	0.04
Level limit (n_{limit})	100
Level probability (p_{L})	0.1

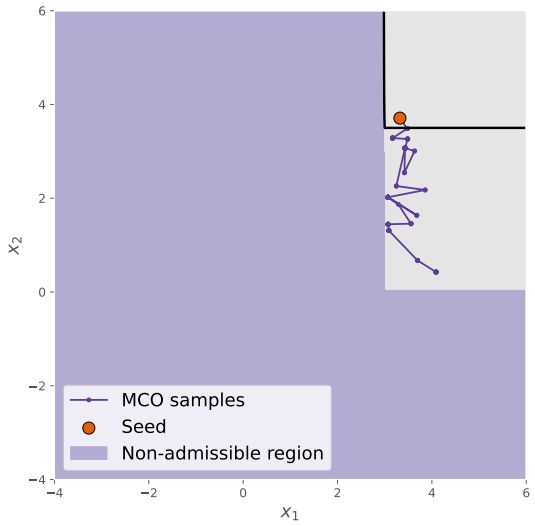
Table 4.1: NInS user-defined parameter values.



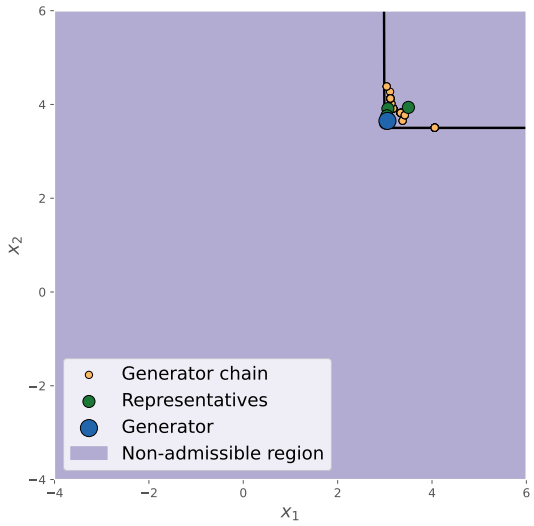
(a) First MCO.



(b) Safe generator.



(c) Second MCO.



(d) Location failure generator.

Figure 4.3: A run of NInS on the black swan function.

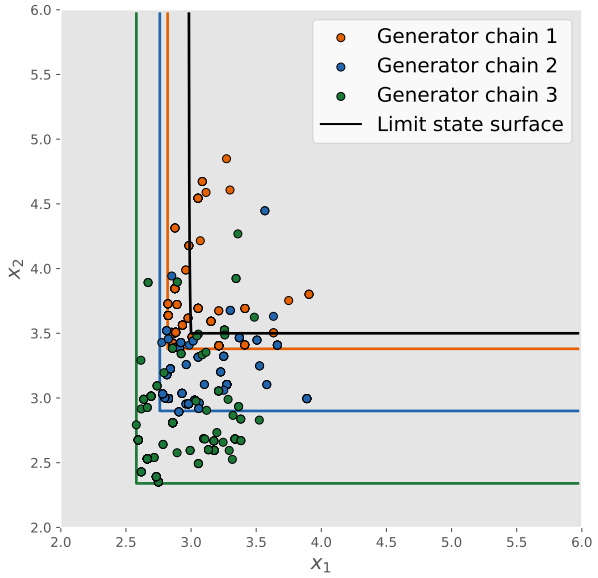


Figure 4.4: Smoothing the black swan function.

Parameter	Value
Level size (N_{L})	2000
Level probability (p_{L})	0.1
Level limit (n_{limit})	10
Check number (n_{check})	2

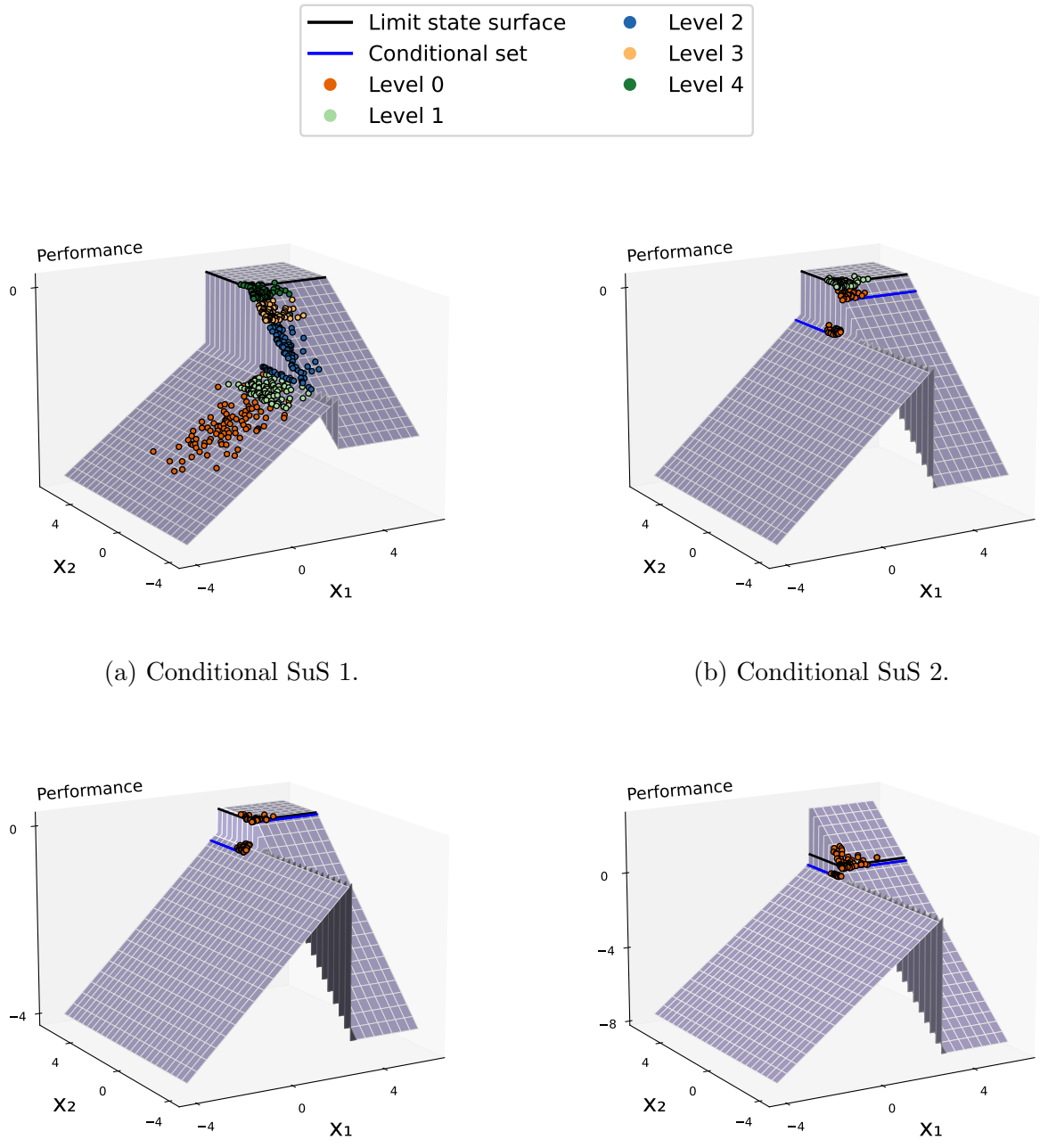
Table 4.2: NDSuS user-defined parameter values.

4.3 Niching Subset Simulation

This section introduces an alternative novel approach of integrating niching techniques into SuS, called niching subset simulation (NSuS). During the course of this work, NSuS was developed before NDSuS and is the subject of a published paper [20]. The NSuS algorithm has some deficiencies, and does not perform well on all the benchmarks considered in this work. The NDSuS algorithm is in fact the result of trying to remedy these deficiencies, which resulted in a better more robust algorithm, hence why it was presented first here. However, there are some novel ideas used within NSuS that could be of some merit in their own right and might be potentially useful for other algorithms. Additionally, it is important to understand why NSuS does not perform as well as NDSuS.

4.3.1 Hill valley graph

In some sense, intermediate methods replace the need for an initial sampling procedure using the intermediate densities. That is, intermediate methods simultaneously locate the failure region and build an estimate for the probability of failure, whereas other reliability methods first locate the failure region, and then, in a distinct step, estimate the probability of failure. This makes the NDSuS algorithm slightly unusual, since it is an intermediate method that uses a separate initial sampling procedure. In this aspect, the NSuS is a more natural algorithm, since it does not use an initial sampling procedure. Instead it uses niching techniques of the levels produced by SuS to attempt to identify



(a) Conditional SuS 1.

(b) Conditional SuS 2.

(c) Conditional SuS 3.

(d) Conditional SuS 4.

Figure 4.5: Runs of conditional SuS on the four conditional reliability problems that the black swan function has been decomposed into.

g	Dimension	Mean P_F	NRSME	ND%	Mean g evals
g_{pwl}	2	3.25×10^{-5}	0.30	100%	3.19×10^4
	100	3.20×10^{-5}	0.29	100%	3.17×10^4
g_{veh}	3	1.30×10^{-6}	0.32	100%	8.09×10^4
	99	1.33×10^{-6}	0.31	100%	9.41×10^4
g_{mix}	2	2.34×10^{-4}	0.24	100%	1.65×10^4
	100	2.22×10^{-4}	0.18	100%	1.66×10^4
g_{bs}	2	3.15×10^{-7}	0.55	99%	2.33×10^4
	100	3.11×10^{-7}	0.44	99%	2.28×10^4
g_{td}	2	2.29×10^{-5}	0.23	100%	3.11×10^4
	100	2.38×10^{-5}	0.22	100%	3.11×10^4
g_{mb}	2	1.07×10^{-5}	0.24	99%	9.97×10^4
	100	1.12×10^{-5}	0.26	100%	1.02×10^5
g_{loss}	32	4.35×10^{-3}	0.15	100%	6.50×10^3
	102	1.90×10^{-3}	0.21	100%	6.19×10^3

Table 4.3: NDSuS results on numerical experiments.

BoAs as the intermediate sequence of densities progresses.

After every level is created in NSuS, a process called the *hill valley graph partitioner* is ran on that level and outputs a partition of the input space. This is done in three steps. First, an object called the *hill valley graph (HVG)* is constructed using the levels of the sample. Next, a *community detection* algorithm is ran on the graph, that assigns labels to the samples of the level. Finally, a classifier is ran on the now labelled samples to produce a partition of the input space. The idea is that the sets in the partition should be an approximation of the BoAs of the reliability problem.

The HVG is constructed in the following way. Firstly, n_g samples are randomly selected without replacement from the level and labelled as $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n_g}$ where n_g is user-defined parameter called the graph size. These samples will be referred to as *graph samples* and are the nodes of the HVG that will be denoted by G^{HVG}. The adjacency matrix of the HVG is given as

$$A_{ij} := \text{HVT}(\boldsymbol{x}_i, \boldsymbol{x}_j), \tag{4.27}$$

for $1 \leq i, j \leq n_g$. The idea of the HVG is that samples in the same BoA should be more likely to share an edge. Note that this process does not depend on the dimension of the reliability problem, and so is a useful way of understanding the topology of a high dimensional space by converting it into graph space. Of course, this comes at the cost of additional performance function evaluations.

Once the HVG has been constructed, it may now be analysed. A *community* of a graph is a set of vertices which is internally densely connected and sparsely connected to the other vertices in the graph. Due to the manner in which the HVG has been constructed, vertices in the same community should correspond to samples in the same BoA, and so the problem of deciding niches has been transformed into community detection on a graph. Community detection algorithms can be used to label the samples according to what niche they belong to. There are many community detection algorithms that can be chosen from in order to complete this task.

One important consideration when deciding between community detection algorithms is the parameters they require the user to define. Some community detection algorithms,

such as Louvain Community Detection algorithm [93], require the user to specify a resolution parameter. The higher the resolution parameter the more attention that is paid to fine-grained details of the graph and so smaller communities are encouraged. The lower the resolution parameter the more the algorithm focuses on large generalised structures of the graph resulting in larger communities. Despite the fact that 1 is often used a default value for resolution, there is still the potential for the performance of NSuS to be sensitive to such a parameter. Other options, like the Fluid Communities algorithm [94], require the user to a priori specify the number of communities to be identified. However, in a black-box reliability problem there is no way of knowing how BoAs there are a priori.

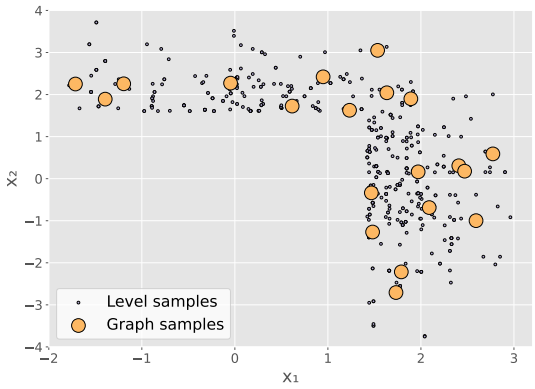
With the above in mind, the asynchronous label propagation (ALP) algorithm [95] is an attractive option since it does not require the user to specify any parameters. For this reason, ALP was used in the numerical examples in [20]. The ALP algorithm works in the following way. Denote a graph as $G = (V, E)$, with vertices $V = \{x_1, \ldots, x_n\}$. ALP updates the labels of the vertices iteratively through different discrete time steps. Let $C_{x_i}(t)$ denote the label of x_i at time t. At the beginning of the algorithm each sample is assigned its own unique label, that is $C_{x_i}(0) = i$. To obtain the labels at step t given the labels at step $t-1$, first the vertices are given a random ordering in which they shall be updated. Let n_i be the number of vertices adjacent to x_i. In the specified order, node x_i is updated with following rule:

$$C_{x_i}(t) := h(C_{x_{i1}}(t), \ldots, C_{x_{im}}(t), C_{x_{i(m+1)}}(t-1), \ldots, C_{x_{in_i}}(t-1)), \tag{4.28}$$

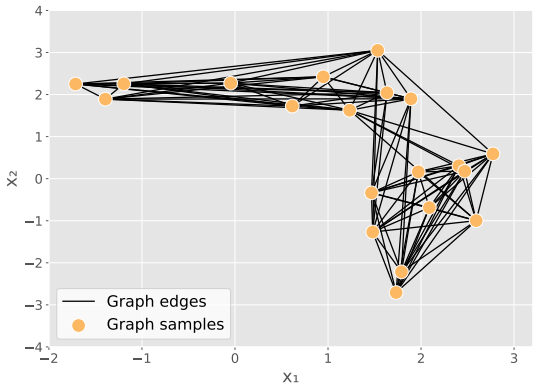
where $x_{i1}, \ldots, x_{im}$ are the vertices adjacent to x_i that have already been updated and $x_{i(m+1)}, \ldots, x_{in_i}$ are the vertices adjacent to x_i that are still awaiting an update for this iteration, and h is function that returns the most common label. In the event of a tie, h picks a label amongst those tied uniformly at random. Due to this tie breaking procedure it is inappropriate to have a stopping condition that looks for no label changes in an iteration, since it is always possible for some labels to change if there is a tie. Consequently, instead the algorithm stops if every vertex has a label that is amongst those labels that have caused a tie.

Once the labels have been assigned to the graph samples, a classification problem in the original input space has been defined. A classifier may now be trained on the labelled samples in order to define a partition of the input space. Note that if the community detection algorithm only identifies one niche, then no classification algorithm is required. Instead the trivial partition, containing one set that is the entire input space, is returned instead. The classification pipeline can be arbitrarily sophisticated, potentially consisting of steps such as data preprocessing, dimension reduction and hyperparameter optimisation by splitting the data up into train and test sets. More details of a specific classification pipeline will be discussed in Section 5.2.2.

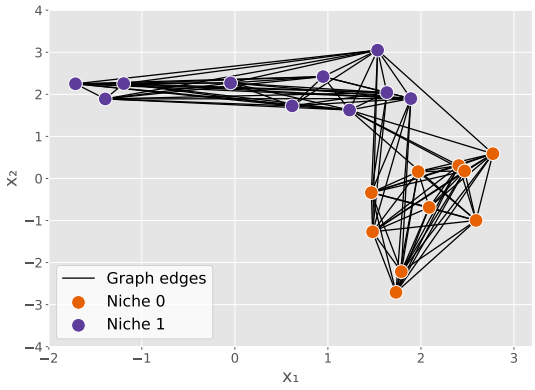
Figure 4.6 depicts the entire hill valley graph partitioner procedure on level 1 of a SuS run on the piecewise linear function. To begin, the graph samples are randomly selected from the level, as shown in Figure 4.6a. Next, Figure 4.6b illustrates how the HVG is constructed using the hill valley test. After that, the community detection algorithm is ran on the HVG to identify the niches, as shown in Figure 4.6c. In this case, the algorithm correctly finds two different niches, corresponding to the two BoA of the piecewise linear function. Finally the classifier is ran on the the now labelled samples resulting in a partition of the input space, as can be seen in Figure 4.6d.



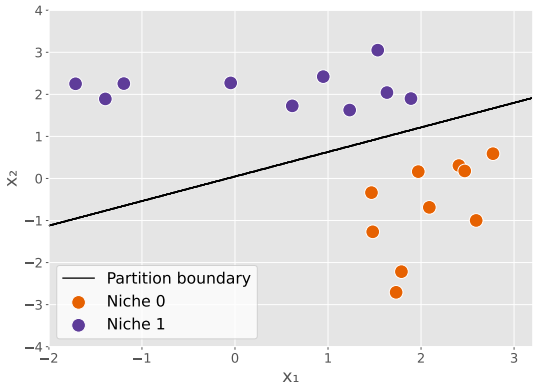
(a) Selecting the graph samples.



(b) Constructing the hill valley graph.



(c) Performing community detection on the hill valley graph.



(d) Fitting a classifier to the labelled samples.

Figure 4.6: The hill valley graph partitioner acting on level 1 of SuS on the piecewise linear function.

4.3.2 Algorithm

The NSuS algorithm, similar to NDSuS consists of a sequence of cSuS runs. It begins with a cSuS run where the conditional set is the entire input space and the initial level is sampled from the input distribution, so identical to a regular SuS run. However after every level has been generated, the hill valley graph partitioner is then ran on that level. If the trivial partition containing one set is returned, then the cSuS algorithm simply carries on and creates the next level. However if more than one set is returned then the cSuS algorithm is stopped and *branching* occurs. This means that a new cSuS algorithm is started in each set of the partition. That is, the conditional set of each of these new cSuS runs is the conditional set of the current cSuS run, combined with the corresponding set of the partition and the current intermediate failure region. The initial level of each of the cSuS algorithm may be constructed with an MCMC algorithm and the current level samples. This process then continues recursively, where all of these cSuS algorithms are run, but on each level the hill valley graph partitioner is used to decide if some branching should happen. This entire process is more formally defined by the following steps.

Niching subset simulation:

1. Sample N_{L} times from the input distribution, and denote the resulting collection L_0. Set $A = \mathbb{R}^d$. Set $\mathcal{T} = \{(L_0, A)\}$.

2. If $\mathcal{T} = \emptyset$, then terminate the algorithm.

3. Choose a member of the set $\mathcal{T}$, label it (L_0, A), and remove it from $\mathcal{T}$.

4. Initialise a cSuS algorithm with initial level L_0 and confining set A. Call this cSuS algorithm the current cSuS algorithm.

5. Generate one level of the current cSuS algorithm and call it L. If a stopping condition of the current cSuS algorithm is triggered by this level go to Step 2.

6. Run the hill valley graph partitioner on L and label the resulting partition $A_1, \ldots, A_{n_p}$. If $n_p = 1$, go to Step 5.

7. Let $\mathcal{F}'$ denote the intermediate failure region associated with L. Define the following conditional sets,
$$A'_i := A \cap A_i \cap \mathcal{F}', \tag{4.29}$$
for $1 \leq i \leq n_p$.

8. For $1 \leq i \leq n_p$, generate initial levels $L_0^{(i)}$ of size N_{L} distributed according $f|A'_i$ using samples from $L \cap A'_i$ as seeds for a MCMC algorithm.

9. For $1 \leq i \leq n_p$, $\mathcal{T} \leftarrow \mathcal{T} \cup (L_0^{(i)}, A'_i)$. Go to Step 2.

At the end of the algorithm, the resulting cSuS runs can be sorted into a tree structure. Let the first cSuS run be the root of the tree. Given any two cSuS runs, say cSuS 1 and cSuS 2, then cSuS 2 is a child of cSuS 1 if it was created by a branching during the running of cSuS 1. Now the estimates of these cSuS runs will be labelled according to this tree structure. Note firstly that when a cSuS algorithm branches, it can be used to estimate conditional probabilities of the confining sets of each its children. Suppose the tree has

K leaves, that is cSuS runs without children. Label the leaves 1 to K. For $1 \leq j \leq K$, consider the path from the root of the tree to the jth leaf, and label cSuS algorithms on this path from 1 to s_j. For the ith cSuS run on the path let $\hat{P}_i^{(j)}$ denote the estimator for the conditional probability of the confining set of the next cSuS algorithm on the path. Let $\hat{P}_{s_j}^{(j)}$ be the conditional probability of failure estimate of the leaf. Putting this all together, the NSuS estimator for the probability of failure is given as

$$P_{\mathcal{F}} \approx \hat{P}_{\mathcal{F}}^{\text{NSuS}} := \sum_{j=1}^{K} \prod_{i=1}^{s_j} \hat{P}_i^{(j)}. \tag{4.30}$$

Notice that this has an identical form to the NDSuS estimator. Both algorithm decompose the reliability problem into easier conditional reliability problems.

There some important problems with the NSuS algorithm. The first is that confining sets are defined using a classifier. This is is an issue since the the higher the dimension of a problem, the more samples the classifier needs in order to correctly partition the input space, ultimately resulting in a higher computational cost. This defeats the main purpose of the ratio methods in this work which is that they are supposed to be suitable for very high dimensional problems. This issue could be fixed by simply adjusting the level probability of each level so that each niche has a sample selected as a seed, instead of the using the partition. As mentioned previously, this simpler approach has a major drawback in that a high level probability will greatly increase the computational cost of the algorithm.

The more fundamental issue is that it is very possible that during the entire algorithm run, no samples at all are generated in a BoA that contains an important part of the failure region. In this case, the HVG has no hope of detecting the BoA. For example, the BoA that contains the design point of the meatball function has a very low probability with respect to the input density, and so the initial level is very unlikely to ever produce any samples within it. The same could be said for the black-swan function as well. Ultimately in these cases, there is not enough information locally for the algorithm to extrapolate what the failure region will look like. This is why it seems that a strategy involving an initial sampling routine that can provide global information should probably be preferred in general.

5 Niching model methods

This chapter introduces a novel framework that allows niching techniques to be combined with methods that explicitly model the failure density or failure region, such as importance sampling, bridge sampling and line sampling. It will be assumed again in this chapter the reliability problem being considered is in SNS. However, as with the previous chapter, so long as an appropriate MCMC algorithm and parametric family of densities can be identified, there is no reason these techniques could no be used on reliability problems that are not in or can not be transformed into SNS.

Section 5.1 introduces the general modular framework that will be considered and allows for the use of niching techniques with model methods. Failure models, that is models that attempt to model the failure density or failure region, are discussed Section 5.2. Finally, Section 5.3, explores how well the proposed methods perform on the selected benchmarks.

5.1 Framework

Methods that attempt to model the failure density or failure region typically first employ and initial sampling procedure in order to generate failure samples that may then be used to then fit the parameters of some model. Finally that model is used in some manner to estimate the probability of failure. Although a method such iCE does not explicitly use distinct initial sampling, fitting and estimating stages, it can certainly be interpreted in this way. In particular, all the levels that are produced prior to the final level are essentially an initial sampling scheme, attempting to locate the failure region. Then, the final model is fit using the penultimate level, which will likely contain a significant number of failure samples. Then the final model is sampled from to produce the final level, which is used as samples for an IS estimator.

There is another deficiency of iCE that has not yet been discussed, that is aside from the challenges that basin multimodal methods present. As has been noted previously [53, 54], the importance weights that are used in the fitting of the models in high dimensions can become degenerate. That is, the importance weights can become very large for very few samples, and so only very few points contribute to the estimation of the parameters. One approach to solving this problem is to use MCMC algorithms to sample from the failure density and those use those samples for fitting, as was done in [53].

However, this strategy has another issue that has been central to the work of this thesis. The failure density could be metastable multimodal with respect to whichever MCMC algorithm is being used. In this case, even if all the high density MSRs are populated with seeds, which itself is not a trivial task, it is difficult to known a priori what the relative sizes of the MSRs are with respect to failure density and thus how many seeds should be used in each one. In these cases, sampling with an ensemble of MCMC

algorithms could lead to a poor approximation of the failure density. Put another way, in these cases, the failure density could be thought of as a mixture distribution, with one component for each MSR. However, when the incorrect amount of seeds are assigned for each MSR, it is as though the ensemble of MCMC algorithms are sampling from this mixture distribution, but with the incorrect component weights.

The idea of the framework that will be presented in this section is to fix this issue with the MCMC approach. To populate all the MSRs of a reliability problem, the NInS algorithm will be used. The generator chains that are created during theNInS procedure will then be used to fit models for the failure density and failure region. However, there is no guarantee that each MSR contains the correct relative amount of samples from the generator chains, according to their weight with respect to the failure density. The idea in this work will be to fit a mixture model to the generator chains, but then to employ a novel component weight correction routine.

The component weight correction routine will itself rely another another novel concept. When a MCMC algorithm attempts to sample from a failure density, many of the proposed samples will be rejected for lying outside of the failure region. The process of deciding whether a sample belongs in the failure region has a computational cost, that is, the performance function needs to be evaluated at the proposed sample. However, typically the knowledge of the value of the performance function of these points is not used again in reliability methods. That is, samples that are proposed but rejected for not lying within the failure region, which will be referred to as *proposal samples*, are often simply discarded. The proposed framework makes used of these proposal samples, for multiple different reasons.

5.1.1 Proposal models

Consider the following generating process. Run a MCMC algorithm with the failure density as it stationary distribution. Instead of returning the Markov chain samples however, instead replace repeated samples that were the result of rejecting a proposal because it lay outside the failure region, with that proposal sample. This generating process will be referred to as sampling from a *proposal failure density* with a PDF denoted as f_{prop} and the set of samples will be referred as *proposal failure samples*. Of course, in general, the form of the proposal density will not be known, and it depends on the failure density as well as the specific MCMC algorithm that is being used. In particular, the proposal density a MCMC algorithm uses heavily influences the proposal density. It should be noted that for MCMC algorithms such as MM, candidate samples may be rejected even if they do lie in the failure region. These are not proposal samples. There are other MCMC algorithms such pCN where all the rejected candidates are proposal samples.

This section will focus on attempting to model the proposal failure density, rather than the failure density. There are a few reasons that motivate this approach. The first reason the proposal failure density is going to be considered is that it is often easier to model than the failure density. This is because the failure density typically has very sudden changes in density, from high density to zero, due to the shape of the failure region. The result is that the failure density looks like a type of truncated density which can be difficult for many standard parametric families, such as a family of normal distributions, to model correctly. In contrast, the proposal failure density will typically have a more natural shape. Ultimately the goal will be to model the failure density, and so, as will

be seen, the model of the proposal failure density will be used as an intermediary step. Another reason to consider the proposal failure density is that there are more samples available for fitting. That is, the repeated samples from the failure density get replaced with the proposal samples, providing more information. Finally, as will be seen in later sections, some of the models that will be considered for the failure density will require safe samples as well as failure samples in their fitting process.

Now the mixture model that will be used to model the proposal failure density will be described. In this work, a parametric family of densities will be chosen that naturally align with the important ring in SNS. This is done so that relatively fewer parameters are required to describe each density, and as a result, fewer samples are required to fit those densities. Note that if the reliability problem is not in SNS, or can not be transformed into SNS, some other parametric family would be required. This work follows the approach suggested in [57], of fitting a *von Mises-Fisher-Nakagami mixture (vMFNM)* model. This is the model that iCE uses. It should be noted that this approach is a generalisation of the strategy used in [54], where a *von Mises-Fisher mixture (vMFM)* model is considered. The vMFM model assumes that the failure samples have radius $\sqrt{d}$, which becomes inaccurate in low dimensions. Conversely, the vMFNM distribution explicitly models the radius of the failure samples. The vMFNM model is most naturally described in polar coordinates. Each $\boldsymbol{x} \in \mathbb{R}^d$ can be rewritten as $\boldsymbol{x} = r\boldsymbol{a}$ where $r \in \mathbb{R}$ is a scalar radius and $\boldsymbol{a} \in \mathbb{R}^d$ is a unit direction vector. The direction and radius are modelled separately, by the *von Mises-Fisher (vMF)* distribution and *Nakagami* distribution respectively.

The vMF distribution is given by,

$$f_{\text{vMF}}(\boldsymbol{a}; \boldsymbol{\mu}, \kappa) := C_d(\kappa) \exp(\kappa \boldsymbol{\mu}^T \boldsymbol{a}), \tag{5.1}$$

where $\boldsymbol{\mu}$ is the the mean direction, with $\|\boldsymbol{\mu}\| = 1$, and $\kappa \geq 0$ is the concentration parameter. The normalising constant $C_d(\kappa)$ is defined as

$$C_d(\kappa) := \frac{\kappa^{d/2-1}}{(2\pi)^{d/2} I_{n/2-1}(\kappa)}, \tag{5.2}$$

where I_k is the modified Bessel function of the first kind with order k. The Nakagami distribution is given by

$$f_{\text{N}}(r; m, \Omega) := \frac{2m^m}{\Gamma(m)\Omega^m} r^{2m-1} \exp\left(-\frac{m}{\Omega} r^2\right), \tag{5.3}$$

where $m \geq 0.5$ is the shape parameter, $\Omega > 0$ is the spread parameter and Γ is the gamma function. Together, these distributions define the von Mises-Fisher-Nakagami (vMFN) distribution,

$$f_{\text{vMFN}}(r, \boldsymbol{a}; m, \Omega, \boldsymbol{\mu}, \kappa) := f_{\text{N}}(r; m, \Omega) \cdot f_{\text{vMF}}(\boldsymbol{a}; \boldsymbol{\mu}, \kappa). \tag{5.4}$$

Finally, in order to be capable of modelling failure densities and the corresponding proposal failure densities with multiple MSRs, a mixture distribution is defined. This requires a set of $K \times 5$ parameters,

$$\boldsymbol{\nu} = \{\boldsymbol{\nu}_k := (\pi_k, m_k, \Omega_k, \boldsymbol{\mu}_k, \kappa_k) : 1 \leq k \leq K\}, \tag{5.5}$$

where the π_k are the component weights with $\sum_k \pi_k = 1$ and K is the number of components. Let a vMFN component be defined as

$$f_{\text{vMFNc}}(r, \boldsymbol{a}; \boldsymbol{\nu}_k) := \pi_k \cdot f_{\text{vMFN}}(r, \boldsymbol{a}; m_k, \Omega_k, \boldsymbol{\mu}_k, \kappa_k). \tag{5.6}$$

The vMFNM distribution is given as

$$f_{\text{vMFNM}}(r, \boldsymbol{a}; \boldsymbol{\nu}) := \sum_{k=1}^{K} f_{\text{vMFNc}}(r, \boldsymbol{a}; \boldsymbol{\nu}_k). \tag{5.7}$$

The first step of this framework is the to run the NInS procedure. Once that has been completed, Denote the union of the proposal failure samples of all the generator chains as $(\boldsymbol{x}_i)_{i=1}^N$. Now let these sample be given by the polar coordinates $(r_i \boldsymbol{a}_i)_{i=1}^N$. These samples will be used to fit the mixture distribution. The parameters can be fit to the samples by minimising the relative cross entropy between the parametric family and the proposal failure density. Following the cross entropy approach in Section 2.3.3, an approximate objective function can be derived,

$$L(\boldsymbol{\nu}) := \frac{1}{N} \sum_{i=1}^{N} \ln(f_{\text{vMFNM}}(r_i, \boldsymbol{a}_i; \boldsymbol{\nu})). \tag{5.8}$$

Of course, this optimisation problem is equivalent to maximum likelihood estimation for a vMFNM distribution. An *expectation-maximisation (EM)* algorithm, which is an iterative algorithm where each iteration comprises an expectation and maximisation step, is typically employed to optimise these types of objective functions.

This section follows closely the EM approach detailed in [57]. However, since the samples are sampled directly from the proposal failure density, no importance weights appear in the approximate cross entropy objective function. This means that the task can be considered a regular maximum likelihood estimation problem, rather than a weighted maximum likelihood estimation problem. This is significant, it avoids the issue of degenerate weights in high dimensions.

On the l^{th} iteration of the EM algorithm, the estimated parameters are denoted as,

$$\hat{\boldsymbol{\nu}}^{(l)} := \{\hat{\boldsymbol{\nu}}_k^{(l)} := (\hat{\pi}_k^{(l)}, \hat{m}_k^{(l)}, \hat{\Omega}_k^{(l)}, \hat{\boldsymbol{\mu}}_k^{(l)}, \hat{\kappa}_k^{(l)}) : 1 \leq k \leq K\}. \tag{5.9}$$

In the l^{th} expectation step, the objective function is evaluated, $L(\boldsymbol{\nu}^{(l)})$. Note that the objective function is an approximation of an expectation. Next, the latent component label of each sample is estimated. This assignment can be done with the *posterior probabilities*,

$$\gamma_k(r, \boldsymbol{a}; \boldsymbol{\nu}) := \frac{f_{\text{vMFNc}}(r, \boldsymbol{a}; \boldsymbol{\nu}_k)}{f_{\text{vMFNM}}(r, \boldsymbol{a}; \boldsymbol{\nu})}. \tag{5.10}$$

Let $\gamma_{i,k}^{(l)} := \gamma_k(r_i, \boldsymbol{a}_i; \hat{\boldsymbol{\nu}}^{(l)})$ for $1 \leq k \leq K$ and $1 \leq i \leq N$. On iteration number $l+1$, the maximisation step attempts to maximise the objective function, given the posterior probabilities from iteration l. Using auxiliary variables,

$$\bar{R}_k := \min\left(\frac{\| \sum_{i=1}^N \gamma_{i,k}^{(l)} \cdot \boldsymbol{a}_i \|}{\sum_{i=1}^N \gamma_{i,k}^{(l)}}, 0.95 \right), \tag{5.11}$$

$$\mu_{j,k} := \frac{\sum_{i=1}^N \gamma_{i,k}^{(l)} \cdot r_i^j}{\sum_{i=1}^N \gamma_{i,k}^{(l)}}, \tag{5.12}$$

the following update rules were derived in [57]:

$$\hat{\pi}_k^{(l+1)} := \frac{1}{N}\sum_{i=1}^{N}\gamma_{i,k}^{(l)}, \tag{5.13}$$

$$\hat{\boldsymbol{\mu}}_k^{(l+1)} := \frac{\sum_{i=1}^{N}\gamma_{i,k}^{(l)}\cdot\boldsymbol{a}_{i,k}}{\|\sum_{i=1}^{N}\gamma_{i,k}^{(l)}\cdot\boldsymbol{a}_{i,k}\|}, \tag{5.14}$$

$$\hat{\kappa}_k^{(l+1)} := \frac{\bar{R}_k(d-\bar{R}_k^2)}{1-\bar{R}_k^2}, \tag{5.15}$$

$$\hat{\Omega}_k^{(l+1)} := \mu_{2,k}, \tag{5.16}$$

$$\hat{m}_k^{(l+1)} := \frac{\mu_{2,k}^2}{\mu_{4,k}-\mu_{2,k}^2}, \tag{5.17}$$

for $1 \leq k \leq K$. The Lagrangian multiplier method can be used to analytically derive the rules for π, μ and Ω. This is not possible for m and κ, thus the updating rules are approximated using estimators that have been shown to give empirically good results [96,97]. The upper limit is placed on $\bar{R}$ for numerical stability. At each step, if $|L(\hat{\boldsymbol{\nu}}^{(l+1)}) - L(\hat{\boldsymbol{\nu}}^{(l)})| < 10^{-5}\cdot|L(\hat{\boldsymbol{\nu}}^{(l+1)})|$, then the algorithm is stopped, and the final parameter estimator is relabelled as

$$\hat{\boldsymbol{\nu}}^{(l+1)} = \hat{\boldsymbol{\nu}} := \{\hat{\boldsymbol{\nu}}_k := (\hat{\pi}_k, \hat{m}_k, \hat{\Omega}_k, \hat{\boldsymbol{\mu}}_k, \hat{\kappa}_k) : 1 \leq k \leq K\}. \tag{5.18}$$

For the sake of use in the next section, the final posterior probabilities are also relabelled $\gamma_{i,k} := \gamma_{i,k}^{(l+1)}$.

Note that since a set of parameters is required to compute the posterior probabilities, and the posterior probabilities are required to estimate a set of parameters, some initial set of posterior probabilities must be decided upon before starting the EM algorithm. Determining the initial posterior probabilities is not typically straightforward and the performance of the EM algorithm can be sensitive to this choice. Defining the initial posterior probabilities requires the number of components, K, to also be decided upon. If K is too small, then some important regions of the target density may be missed. However, if K is too large, there may not be enough samples to fit all the parameters accurately. In some cases, prior information regarding the reliability problem, such as the number of components, can be used to aide in this process. However, in this work, it is assumed no such information is available.

Clustering methods like DBSCAN [98] or k-means [99] could be used to decide the initial posterior portabilities. However, these algorithms have their own user-defined parameters, and will often perform poorly in high dimensions because of their reliance on the Euclidean metric [85]. However, because of the use of NInS, there is a natural choice for the number of components. That is the number of connected components of G^{JHF}, as was defined in Section 4.1.1. The posterior probabilities could also be informed by the NInS algorithm. In particular samples from the same generator chains should likely be assigned to the same initial component. However, in testing, this type of approach seemed to offer no benefit over just randomly assigning each sample to a component. This random approach is must simpler and so that was used instead.

Because of the ergodicity issues that metastable multimodal failure densities can cause for MCMC algorithms, the component weight estimate from the previous section is likely to be inaccurate. In this section, one last adjustment is made to the component weight estimate, whilst the other estimators are left as they are. The cross entropy optimisation problem that was used to define the objective function in Equation 5.8 is considered with respect to the proposal failure density and with an additional importance weight term,

$$\boldsymbol{\nu}' := \underset{\boldsymbol{\nu}}{\operatorname{argmax}}\, \mathbb{E}_{f_{\text{prop}}}\left[\ln(h(\boldsymbol{x};\boldsymbol{\nu}))\frac{f(\boldsymbol{x})\mathbb{1}_{\mathcal{F}}(\boldsymbol{x})}{f_{\text{prop}}(\boldsymbol{x})}\right]. \tag{5.19}$$

The parameter set fit in the previous section can be used to approximate the proposal failure density, $f_{\text{prop}}(\cdot) \approx f_{\text{vMFNM}}(\,\cdot\,;\hat{\boldsymbol{\nu}})$. Note that if instead of modelling the proposal failure density, the failure density was modelled by say $f_{\text{vMFNM}}(\,\cdot\,;\boldsymbol{\nu}_{\mathcal{F}})$, then at this stage the approximation $f_{\mathcal{F}} \approx f_{\text{vMFNM}}(\,\cdot\,;\boldsymbol{\nu}_{\mathcal{F}})$ would have to be used. Due to the more difficult shape of the failure density, this approximation tends to be considerably worse than the one that is being used. This is the primary motivation behind the use of the proposal failure density.

This approximation, together with the proposal failure samples, may be used define the following to approximate optimisation problem,

$$\hat{\boldsymbol{\nu}}' := \underset{\boldsymbol{\nu}}{\operatorname{argmax}}\, \frac{1}{N}\sum_{i=1}^{N} \ln(f_{\text{vMFNM}}(r_i, \boldsymbol{a}_i;\boldsymbol{\nu}))w_i, \tag{5.20}$$

with $w_i = f(r_i, \boldsymbol{a}_i)\mathbb{1}_{\mathcal{F}}(r_i, \boldsymbol{a}_i)/f_{\text{vMFNM}}(r_i, \boldsymbol{a}_i;\hat{\boldsymbol{\nu}})$ where the input distribution and indicator function are defined in polar coordinates. The Lagrangian multiplier method may be then used again to derive a weighted updating rule for the component weights,

$$\hat{\pi}'_k := \frac{\sum_{i=1}^{N} w_i \cdot \gamma_{i,k}}{\sum_{i=1}^{N} w_i}, \tag{5.21}$$

for $1 \leq k \leq K$.

These new component weights could be used directly in the model for the failure proposal density. However, sometimes a component may be assigned a negligible amount of weight, when in reality, the proposal failure density does assign a significant amount of weight to the region of the input space that that component covers. This type of error can be corrected later on the framework by sampling more from this region so that the model can be improved. However, the amount of computational resources that the framework assigns to each section of the input space for sampling is determined by these weights. If a weight is so small the the framework assigns no computational resources to sample from there, then it could be the case that the type of correction required never occurs. For this reason, the framework is made more robust by smoothing the weights.

Let w_{smooth} be a user-defined parameter called the *smooth weight*. Now let

$$\hat{\pi}_k^{\text{smooth}} := \begin{cases} \hat{\pi}'_k & \hat{\pi}'_k > w_{\text{smooth}} \\ w_{\text{smooth}} & \hat{\pi}'_k \leq w_{\text{smooth}}, \end{cases} \tag{5.22}$$

for $1 \leq k \leq K$. Now normalise the smoothed weights,

$$\tilde{\pi}_k^{\text{smooth}} := \frac{1}{\sum_{k'=1}^{K} \hat{\pi}_{k'}^{\text{smooth}}} \cdot \hat{\pi}_k^{\text{smooth}}. \tag{5.23}$$

Let $\hat{\boldsymbol{\nu}}^c$ denote the parameter set that is the result of replacing $\hat{\pi}_k$ with $\tilde{\pi}_k^{\text{smooth}}$ in $\hat{\boldsymbol{\nu}}$ for $1 \leq k \leq K$. That is, the final model for the proposal failure density, which will be called the *proposal model* is given as $f_{\text{vMFNM}}(\,\cdot\,;\hat{\boldsymbol{\nu}}^c)$.

It should be noted that the importance weights here can actually be used to estimate the probability of failure with an estimator that will be referred to as the *proposal estimator*,

$$P_{\mathcal{F}} \approx P_{\mathcal{F}}^{\text{prop}} := \frac{1}{N}\sum_{i=1}^{N} w_i. \tag{5.24}$$

This can be thought of as a sort of approximate IS estimator. That is, if the samples were truly sampled from $f_{\text{vMFNM}}(\,\cdot\,;\hat{\boldsymbol{\nu}})$ rather than the proposal failure density, then it would just be an IS estimator. In numerical experiments, it was found that this estimator can often give surprisingly good results for the probability of failure despite not actually being an IS estimator. However, it is not robust enough to be used as a reliability method.

5.1.2 Control flow

The general framework that will now be described will be referred to as the *niching model framework (NMF)*. The NMF consists of two for loops, and outer and inner loop. The purpose of the outer loop is to update a dataset which will be referred as the *NMF dataset*. The purpose of the inner loop is to use the *NMF dataset* to fit the proposal model, and then ultimately fit a model for the failure density, which will be referred to as a *failure model*. The failure model is sampled from and the samples are used to estimate the probability of failure. The sampling from the failure model is done in small batches, rather than all at once. The size of these batches is controlled by a user-defined parameter called the sample size as is denoted by n_{size}. This is done so that it may be monitored how well the failure model approximates the failure density. If it is determined that failure model is a poor approximation of the failure density, the sampling process stops, and the NMF returns to outer loop so that more samples may be added to NMF dataset, so that a new failure model that is a better approximation of the failure density may be fit.

The *NMF dataset* is a collection of Markov chains, with more chains being added with each iteration of the outer loop. Initially, immediately after NInS has terminated, the *NMF dataset* consists of the generator chains that NInS generates. On every iteration of the outer loop the NMF selects new seeds from NMF dataset and then uses MCMC algorithms with the failure density as their stationary distribution to generate new chains, which are then added to the NMF dataset. Note that there also exists an associated dataset, called the *proposal NMF dataset* that consists of all the proposal failure samples of the all the chains in the NMF dataset.

The addition of new chains to the NMF dataset is done in the following way, given a proposal model. Let $(\boldsymbol{x}_i)_{i=1}^N$ be union of all the samples of the chains in the NMF dataset. Let $\gamma_{i,k}^c$ denote the posterior probabilities of these samples with respect to the proposal model. Now define the following sets of indices,

$$I_k = \{1 \leq i \leq N : k = \operatorname*{argmax}_{1\leq k'\leq K} \gamma_{i,k'}^c\} \tag{5.25}$$

for $1 \leq k \leq K$. From each of these sets, uniformly at random select one of the members and denote the results as $i_1, i_2, \ldots, i_K$. Define the budget for sampling as $n_{\text{budget}} =$

$M_{\text{budget}} \cdot d$, where M_{budget} is a user-defined parameter called the budget multiplier. Now, for $1 \leq k \leq K$, run a MCMC algorithm to generate a chain with the failure density as the stationary distribution, $\boldsymbol{x}_{i_k}$ as the seed and of length $\lceil n_{\text{budget}} \cdot \tilde{\pi}_k^{\text{smooth}} \rceil$. The idea is that, assuming the component weights of the proposal model are roughly correct, the correct number of samples should be produced in each MSR, with respect to their relative probabilities under the failure density.

Once the NMF dataset has been updated, a new proposal model may be fit using the associated proposal NMF dataset. The new proposal model will then be used to fit the failure model. The precise details of specific failure models will be covered in the next section. All that is required to understand the control flow of the NMF is that a failure model must have following properties:

- Can be fit given a proposal model and a NMF dataset.
- Can be sampled from, where the resulting samples may be used to estimate the probability of failure
- Can provide an estimate of the CoV of the probability of failure estimate.
- Can estimate a *projected CoV*, that is a estimate for the CoV of the estimate for the probability of failure that would result from sampling some fixed number of additional times.

The control flow of the NMF is governed by a user-defined parameter called *max inner iterations* which is denoted by n_{outer} and a user-defined parameter called *target CoV* that is denoted by δ_{target}. The inner loop of the NMF will be referred to as *failure model estimation*. The NMF is given by the following steps:

Niching model framework:

1. Run the NInS procedure.
2. Set $c_{\text{outer}} = 0$ and $\delta = \infty$.
3. If $c_{\text{outer}} = n_{\text{outer}}$ or $\delta < \delta_{\text{target}}$ then terminate algorithm.
4. Update $c_{\text{outer}} \leftarrow c_{\text{outer}} + 1$.
5. If $c_{\text{outer}} = 1$, then update the generator chains where each chain is increased by length $\lceil n_{\text{budget}} \cdot n_{\text{gen}} \rceil$ where n_{gen} is the number of generator chains. Otherwise use the proposal model fit in the previous iteration to update the NMF dataset.
6. Run failure model estimation, and then update δ as the estimate of the CoV of the probability of failure estimate. Go to Step 3.

The inner loop, failure model estimation, requires another user-defined parameter. Let n_{inner} be the max inner iterations. The failure model estimation procedure requires access to the variable c_{outer} from the outer loop. Failure model estimation is summarised by the following steps:

Failure model estimation:

1. Fit a proposal model using the proposal NMF data.
2. Fit a failure model using the proposal model and NMF data.
3. Set $c_{\text{inner}} = 0$, $\delta_{\text{proj}} = \infty$ and $\delta = \infty$.
4. If ($\delta_{\text{proj}} > \delta_{\text{target}}$ and $c_{\text{outer}} < n_{\text{outer}}$) or $c_{\text{inner}} = n_{\text{inner}}$ or $\delta < \delta_{\text{target}}$ terminate algorithm.
5. Update $c_{\text{inner}} \leftarrow c_{\text{inner}} + 1$.
6. Set $n_{\text{proj}} = (n_{\text{inner}} - c_{\text{inner}}) \cdot n_{\text{size}}$.
7. Sample from failure model n_{size} times. Now update the estimates for the CoV and projected CoV given n_{proj} more samples and denote them as δ and δ_{proj} respectively. Go to Step 4.

5.2 Failure models

This section gives examples of failure models that have the required properties for the NMF. As well being modular and interchangeable components in the NMF, these failure models are defined with respect to general parametric families and so they are modular frameworks themselves. These failure models are fit by making use of the latest fit proposal model in the NMF.

5.2.1 Niching importance sampling

One approach to fitting a model for the failure density could be to consider the collective samples of the NMF dataset, denoted as $(\boldsymbol{x}_i)_{i=1}^N$, and directly fit a mixture distribution to them using an EM algorithm. In particular, the precise EM algorithm that was used to fit vMFNM model to the proposal failure samples could be used with respect to the NMF dataset instead. The issue with this strategy, as previously mentioned, is that components weights will likely be incorrect due to the ergodicity issues of the chains. This is one of the motivations for fitting the proposal model first instead.

The proposal model offers a way of fitting a mixture distribution to the failure samples without the need for an additional EM algorithm to be run. Denote the posterior probabilities of the samples under the proposal model as $\gamma_{i,k}^c$ and the let the component weights of the posterior model be given as $\tilde{\pi}_k^{\text{smooth}}$ for $1 \leq k \leq K$. The posterior probabilities may be used to define K different weighted datasets, that all use the same samples but with different weights. A parametric family of densities may now be independently fit to each of the K weighted data sets, resulting in K different densities. These densities may be combined into one mixture density by assigning them the associated component weights $\tilde{\pi}_k^{\text{smooth}}$ for $1 \leq k \leq K$. All that is required for this fitting procedure is a parametric family with parameters that can be fit with weighted samples. The numerical examples in this work will use the vMFN density and the parameters may be fit to the weighted samples using Equations 5.14–5.17.

The correcting of the component weights is not the only motivation for fitting the proposal distribution first. The above procedure that is used to fit a mixture distribution without the need for an additional EM algorithm has another benefit. As will be explored

in the next section, there exists parametric families of densities with methods for fitting their parameters to weighted samples that do not have a natural EM algorithm for fitting a mixtures. The above approach then offers a way fitting mixtures of this type of parametric family. It should be noted that of course EM algorithms tend to have nice theoretical properties, depending on the nature of the updating rules they use, that allow certain guarantees to be made about the parameter set that is ultimately returned. The approach for fitting a mixture distribution described above is a heuristic approach, and so the same guarantees may not necessarily hold.

Given a failure model that has now been fit the most natural approach to estimating the probability of failure is to use it as an importance density. That is, sample from the failure model and use the samples to construct and IS estimator. The details of the IS estimator for the probability of failure, including an estimator for its CoV, were given in Section 2.2.4. All that remains of the required abilities of a failure model in the NMF is that it can make an estimate for the projected CoV, denoted by $\delta^{\text{IS}}_{\text{proj}}$. That is, the CoV of the IS estimator with respect to some greater number of samples than it is currently using, denoted by N_{proj}. The projected CoV may be estimated by

$$\delta^{\text{IS}}_{\text{proj}} \approx \hat{\delta}^{\text{IS}}_{\text{proj}} := \hat{\delta}^{\text{IS}}_{\mathcal{F}} \cdot \sqrt{\frac{N_{\text{IS}}}{N_{\text{proj}}}}, \tag{5.26}$$

where N_{IS} is the number of samples the IS estimator is currently using and $\hat{\delta}^{\text{IS}}_{\mathcal{F}}$ is the estimator for the CoV of an IS estimator. This specific type of implementation of the NMF that uses an IS estimator will be referred to as niching importance sampling (NIS).

5.2.2 Niching bridge sampling

Bridge samplings (BSs) is another technique that may be used to estimate normalisation constants, similar to IS. It is a reasonably common and well known technique in the field of Bayesian inference, but somewhat surprisingly, it does not appear to be common technique that is used in reliability analysis. Indeed, in a recent extensive review of MC methods for reliability analysis the term BS does not appear once [44]. The only reliability analysis paper that has been found during the course of this work that references BS is [100], in which bridge sampling is used to estimate the ratio of normalising constants in an intermediate method. The estimation of a ratio of normalisation constants is the most standard application of BS. However, as discussed in [101] for example, it is possible to use BS to estimate a singular normalisation constant. This approach will now be recast in the context of reliability analysis.

The probability of failure, which is the normalisation constant of the failure density, may be expressed in terms of expectations in the following way:

$$P_{\mathcal{F}} = \mathbb{E}_f[\mathbb{1}_{\mathcal{F}}(\boldsymbol{x})] \tag{5.27}$$

$$= \mathbb{E}_f[\mathbb{1}_{\mathcal{F}}(\boldsymbol{x})] \cdot \frac{\mathbb{E}_q[\mathbb{1}_{\mathcal{F}}(\boldsymbol{x}) f(\boldsymbol{x}) h(\boldsymbol{x})]}{\mathbb{E}_q[\mathbb{1}_{\mathcal{F}}(\boldsymbol{x}) f(\boldsymbol{x}) h(\boldsymbol{x})]} \tag{5.28}$$

$$= \frac{\mathbb{E}_q[\mathbb{1}_{\mathcal{F}}(\boldsymbol{x}) f(\boldsymbol{x}) h(\boldsymbol{x})]}{\mathbb{E}_{f_{\mathcal{F}}}[q(\boldsymbol{x}) h(\boldsymbol{x})]}, \tag{5.29}$$

where $h(\boldsymbol{x}) : \mathbb{R}^d \to \mathbb{R}$ is called the *bridge function*, and q is a PDF that will be referred to as the importance density, since it plays a very similar role as the importance density in IS. The *optimal bridge function* is given as

$$h(\boldsymbol{x}) := \frac{1}{\mathbb{1}_{\mathcal{F}}(\boldsymbol{x}) f(\boldsymbol{x}) + P_{\mathcal{F}} \cdot q(\boldsymbol{x})}. \tag{5.30}$$

where the term optimal refers to the fact that it will minimise the MSE of final estimator that is yet to be described [102].

Given N independent samples from the importance density, $\boldsymbol{x}_1^q, \ldots, \boldsymbol{x}_N^q \sim q$, and N MCMC samples from the failure density $\boldsymbol{x}_1^{\mathcal{F}}, \ldots, \boldsymbol{x}_N^{\mathcal{F}} \sim f_{\mathcal{F}}$, define the the following auxiliary variables:

$$b_i^q = \frac{\mathbb{1}_{\mathcal{F}}(\boldsymbol{x}_i^q) f(\boldsymbol{x}_i^q)}{q(\boldsymbol{x}_i^q)} \quad \text{and} \quad b_i^{\mathcal{F}} = \frac{q(\boldsymbol{x}_i^{\mathcal{F}})}{\mathbb{1}_{\mathcal{F}}(\boldsymbol{x}_i^{\mathcal{F}}) f(\boldsymbol{x}_i^{\mathcal{F}})} \tag{5.31}$$

for $1 \leq i \leq N$. By replacing the expectations in Equation 5.29 with their corresponding MC and MCMC estimators, rearranging the result slightly and the substituting in the the auxiliary variables, the following estimator for the probability of failure may be derived,

$$P_{\mathcal{F}} \approx \frac{\sum_{i=1}^N \frac{b_i^q}{b_i^q + P_{\mathcal{F}}}}{\sum_{i=1}^N \frac{b_i^{\mathcal{F}}}{1 + P_{\mathcal{F}} \cdot b_i^{\mathcal{F}}}}. \tag{5.32}$$

There is of course a major issue with this estimator because it requires access to probability of failure, which is the quantity it is attempting to estimate in the first place. To remedy this problem, an iterative scheme is used based on this estimator. That is, define

$$\hat{P}_{t+1}^{\text{BS}} := \frac{\sum_{i=1}^N \frac{b_i^q}{b_i^q + \hat{P}_t^{\text{BS}}}}{\sum_{i=1}^N \frac{b_i^{\mathcal{F}}}{1 + \hat{P}_t^{\text{BS}} \cdot b_i^{\mathcal{F}}}}, \tag{5.33}$$

for $1 \leq t \leq T - 1$, where T is the iteration where the iterative scheme terminates due to some predefined convergence criteria. The initial guess for the probability of failure use in this work is $\hat{P}_1^{\text{BS}} = 1$. Finally, the BS estimator for the probability of failure is given as

$$P_{\mathcal{F}} \approx \hat{P}_{\mathcal{F}}^{\text{BS}} := \hat{P}_T^{\text{BS}}. \tag{5.34}$$

Let the CoV of the BS estimator be denoted as $\delta_{\mathcal{F}}^{\text{BS}}$. Given the following auxiliary functions,

$$f_1(\boldsymbol{x}) = 2 \cdot \frac{\mathbb{1}_{\mathcal{F}}(\boldsymbol{x}) f(\boldsymbol{x})}{\mathbb{1}_{\mathcal{F}}(\boldsymbol{x}) f(\boldsymbol{x}) + \hat{P}_{\mathcal{F}}^{\text{BS}} \cdot q(\boldsymbol{x})} \tag{5.35}$$

$$f_2(\boldsymbol{x}) = 2 \cdot \frac{\hat{P}_{\mathcal{F}}^{\text{BS}} \cdot q(\boldsymbol{x})}{\mathbb{1}_{\mathcal{F}}(\boldsymbol{x}) f(\boldsymbol{x}) + \hat{P}_{\mathcal{F}}^{\text{BS}} \cdot q(\boldsymbol{x})} \tag{5.36}$$

the following estimator for the CoV of the BS estimator was derived in [103],

$$\delta_{\mathcal{F}}^{\text{BS}} \approx \hat{\delta}_{\mathcal{F}}^{\text{BS}} := \sqrt{\hat{\delta}_q \left[f_1(\boldsymbol{x})\right]^2 + \hat{\delta}_{f_{\mathcal{F}}} \left[f_2(\boldsymbol{x})\right]^2} \tag{5.37}$$

Similar to the IS case, the projected CoV with respect to some additional number of samples N_{proj}, is denoted by $\delta^{\text{BS}}_{\text{proj}}$. The projected CoV may be estimated by

$$\delta^{\text{BS}}_{\text{proj}} \approx \hat{\delta}^{\text{BS}}_{\text{proj}} := \hat{\delta}^{\text{BS}}_{\mathcal{F}} \cdot \sqrt{\frac{N}{N_{\text{proj}}}}. \tag{5.38}$$

An implementation of the NMF that uses the BS estimator will be refereed to as niching bridge sampling (NBS).

The importance density q should model the failure density. The exact same process that was used in the previous section, using the proposal model and the NMF dataset, may be used to fit a vMFNM model as the importance density. The set of Markov chains that were just used to fit the proposal model may be used again now to generate the MCMC samples required for the BS estimator. Firstly, the chains are updated according to the correct ratios with total collective additional n_{size} samples. This first update is done to reduce the correlation between the samples that will eventually be used in BS estimator, and the samples that are used to fit the importance density. This initial update is only done on the the first iteration of the inner loop. Then that process is simply repeated to get n_{size} MCMC samples.

The BS estimator has a nice property, which is in fact the main reason it is being considered here. Note that for an IS estimator, if at some point in the input space $f_{\mathcal{F}}(\boldsymbol{x}) > 0$, for the IS estimator to be well defined, the importance density must also have $q(\boldsymbol{x}) > 0$. This is not the case for the importance density in the BS estimator. This makes it easier to consider another type of mixture distribution for modelling the failure density. Let a *classifier mixture distribution* be given as

$$f_{\text{class}}(\boldsymbol{x}) = \sum_{i=1}^{K} \pi_i \frac{\mathbb{1}_{\hat{\mathcal{F}}_i}(\boldsymbol{x}) f(\boldsymbol{x})}{P_{\hat{\mathcal{F}}_i}}, \tag{5.39}$$

where the $\hat{\mathcal{F}}_i$ are approximations of the failure region defined by classifiers, the π_i are the component weights and the normalisation constants are given as

$$P_{\hat{\mathcal{F}}_i} = \mathbb{P}_f(\hat{\mathcal{F}}_i) \tag{5.40}$$

for $1 \leq i \leq K$. The problem with using this type of mixture distribution in importance sampling is that it might not assign non-zero density to all the points in the input space, and so may violate the above condition.

The classifier mixture distribution will be fit in the following way. The proposal NMF dataset will be used in this case, since it contains both the failure and safe samples. The proposal model may be used to generate the posterior probabilities of all the samples in the proposal NMF dataset. The result is K weighted binary classification datasets, where the labels indicate failure and safe samples. A classifier is then fit to each of the K datasets, which in turn defines $\hat{\mathcal{F}}_i$ for $1 \leq i \leq K$. The weights of the classifier mixture are simply the corresponding weights of the proposal model. Note that the normalisation constants may be estimated with a MC estimator, since there is no computational cost associated with evaluating the classifier since it does not require performance function evaluations. Note also that the classifier mixture may be sampled from with simple rejection sampling using the input distribution.

The binary classifier used in the numerical examples in this work is the linear support vector classifier (LSVC). Let a weighted binary classifier dataset be given as $(\boldsymbol{x}_i, y_i)_{i=1}^{N}$

where $y_i = 1$ if $\boldsymbol{x}_i$ is in the failure region and $y_i = -1$ otherwise, with weights $w_1, \ldots, w_N$. Now the following optimisation problem is solved for the coefficients $\boldsymbol{\alpha} \in \mathbb{R}^d$ and intercept $\alpha_0 \in \mathbb{R}$,

$$\min_{\boldsymbol{\alpha},\alpha_0} \frac{1}{2}\|\boldsymbol{\alpha}\|^2 + C_{\text{reg}} \sum_{i=1}^{N} w_i \max(0, 1 - y_i(\boldsymbol{\alpha}^T \boldsymbol{x}_i + \alpha_0))^2, \tag{5.41}$$

where C_{reg} is a user-defined regularisation parameter. Within the context of a general linear support vector classification optimisation problem, the specific type considered here uses a squared hinge loss function and an L2 penalty function. The only preprocessing of the samples used is that the dataset is normalised such that it has 0 mean and unit variance.

5.2.3 Niching line sampling

The final type of failure model that will be discussed will use a LS estimator. When a NMF implementation uses a LS estimator, it will be referred to as niching line sampling (NLS). This implementation starts identically to the NIS procedure and fits a vMFNM to the samples of the NMF dataset using the posterior probabilities and component weights of the proposal model. Denote the mean directions of the vMF distribution of each component as $\boldsymbol{\mu}_k$ for $1 \leq k \leq K$. These unit directions are perfect candidates for the important directions that will be used for LS estimators. Recalling the notation from Section 2.2.5, let the important direction be given as $\boldsymbol{\alpha}_k := \boldsymbol{\mu}_k$ for $1 \leq k \leq K$.

The idea will be to use K line sampling estimates,, corresponding to the K important directions, and sum the results together to estimate the probability of failure. However, as has been noted before [34], there is an issue with this approach. If any of the important directions happen to be relatively similar, there is potential that the parts of the failure region they are sampling from might overlap, leading to overestimate for the probability of failure. To overcome this problem, the posterior probabilities of the vMFNM model may be considered. That is, let $\gamma_k(\boldsymbol{x})$ be a function that gives the posterior probability of $\boldsymbol{x}$ of the kth component for $1 \leq k \leq K$.

Given an important direction α and a sample that lies in its orthogonal complement $\boldsymbol{x}_{\boldsymbol{\alpha}}$, define the reliability index of the associated one dimensional reliability problem β as the solution to following constrained optimisation problem,

$$\begin{aligned} \min_{t>0} \quad & t \\ \text{s.t.} \quad & \boldsymbol{x}_{\boldsymbol{\alpha}} + t\boldsymbol{\alpha} \in \mathcal{F}. \end{aligned} \tag{5.42}$$

Now a one dimensional reliability problem with respect to a posterior probability function is defined and approximated in a similar way to which the approximation is made in first-order reliability methods,

$$P_{\mathcal{F}}^{\boldsymbol{\alpha}_k}(\boldsymbol{x}_{\boldsymbol{\alpha}_k}) := \int_{\mathbb{R}} \mathbb{1}_{\mathcal{F}}(\boldsymbol{x}_{\boldsymbol{\alpha}_k} + t\boldsymbol{\alpha}_k)\gamma_k(\boldsymbol{x}_{\boldsymbol{\alpha}_k} + t\boldsymbol{\alpha}_k)\phi(t)dt. \tag{5.43}$$

$$\approx \gamma_k(\boldsymbol{x}_{\boldsymbol{\alpha}_k} + \beta\boldsymbol{\alpha}_k)\Phi(-\beta), \tag{5.44}$$

for $1 \leq k \leq K$. Using this approximation, and the fact that

$$\sum_{k=1}^{K} \gamma_k(\boldsymbol{x}) = 1, \tag{5.45}$$

for all $\boldsymbol{x} \in \mathbb{R}^d$, the probability of failure may be approximated in the following way

$$P_{\mathcal{F}} := \sum_{k=1}^{K} \mathbb{E}_f \left[\gamma_k(\boldsymbol{x}\mathbb{1}_{\mathcal{F}}(\boldsymbol{x})\right] \tag{5.46}$$

$$= \sum_{k=1}^{K} \mathbb{E}_{\boldsymbol{X}_{\boldsymbol{\alpha}_k}} \left[P_{\mathcal{F}}^{\boldsymbol{\alpha}_k}\left(\boldsymbol{x}_{\boldsymbol{\alpha}_k}\right)\right] \tag{5.47}$$

$$\approx \sum_{k=1}^{K} \mathbb{E}_{\boldsymbol{X}_{\boldsymbol{\alpha}_k}} \left[\gamma_k(\boldsymbol{x}_{\boldsymbol{\alpha}_k} + \beta\boldsymbol{\alpha}_k)\Phi(-\beta)\right]. \tag{5.48}$$

Each of the expectations in the sum may be estimated by a MC estimator, which results in the *NLS estimator* for the probability of failure

$$P_{\mathcal{F}} \approx \hat{P}_{\mathcal{F}}^{\text{NLS}} := \sum_{k=1}^{K} \hat{\mu}_{\boldsymbol{X}_{\boldsymbol{\alpha}_k}} \left[\gamma_k(\boldsymbol{x}_{\boldsymbol{\alpha}_k} + \beta\boldsymbol{\alpha}_k)\Phi(-\beta)\right]. \tag{5.49}$$

The CoV of the NLS estimator, denoted by $\delta_{\mathcal{F}}^{\text{NLS}}$, may be estimated using CoV estimators for each constituent MC estimator,

$$\delta_{\mathcal{F}}^{\text{NLS}} \approx \hat{\delta}_{\mathcal{F}}^{\text{NLS}} := \sqrt{\sum_{k=1}^{K} \left(w_i \cdot \hat{\delta}_{\boldsymbol{X}_{\boldsymbol{\alpha}_k}} \left[\gamma_k(\boldsymbol{x}_{\boldsymbol{\alpha}_k} + \beta\boldsymbol{\alpha}_k)\Phi(-\beta)\right]\right)^2} \tag{5.50}$$

where

$$w_i = \frac{\hat{\mu}_{\boldsymbol{X}_{\boldsymbol{\alpha}_k}} \left[\gamma_k(\boldsymbol{x}_{\boldsymbol{\alpha}_k} + \beta\boldsymbol{\alpha}_k)\Phi(-\beta)\right]}{\hat{P}_{\mathcal{F}}^{\text{NLS}}} \tag{5.51}$$

for $1 \leq k \leq K$. Finally, the projected CoV, denoted as $\delta_{\text{proj}}^{\text{NLS}}$, may be estimated in the following way,

$$\delta_{\text{proj}}^{\text{NLS}} \approx \hat{\delta}_{\text{proj}}^{\text{NLS}} := \hat{\delta}_{\mathcal{F}}^{\text{NLS}} \cdot \sqrt{\frac{N}{N_{\text{proj}}}}. \tag{5.52}$$

where N is the total number of samples of all the MC estimators added together and N_{proj} is the projected number of samples.

It would desirable if more computational resources were assigned to the important directions that are believed to be pointing towards the relatively high regions of the failure region. The component weights of the the vMFNM model, which are the same as the weights of proposal model, can help fulfil precisely that criteria. That is, when n_{size} samples are added to the NLS estimator, they should be done in such a way that each of the K MC estimators gets a proportion of the samples that is equal to the corresponding component weight.

All that remains to be described is a method for solving the constrained optimisation problem for determining the reliability index of the one dimensional reliability methods. A crude but robust method is used here. To begin, a grid is defined from 0 to 15 using steps of length 0.1. The algorithm starts at one of these points on the grid, say $t = 7$, and evaluates the performance function to check if $\boldsymbol{x}_{\boldsymbol{\alpha}} + t\boldsymbol{\alpha} \in \mathcal{F}$. If it is, then the algorithm starts adjusting t by -0.1 in a series of steps. It continues making these steps

until it leaves the failure region. Otherwise the opposite happens, the algorithm starts adjusting t by 0.1 and keeps going until enters the failure region. Either way the end result will be two consecutive points in the grid $t_{\mathcal{S}}$ and $t_{\mathcal{F}}$ with $\boldsymbol{x}_{\mathcal{S}} := \boldsymbol{x}_{\boldsymbol{\alpha}} + t_{\mathcal{S}}\boldsymbol{\alpha} \in \mathcal{S}$ and $\boldsymbol{x}_{\mathcal{F}} := \boldsymbol{x}_{\boldsymbol{\alpha}} + t_{\mathcal{F}}\boldsymbol{\alpha} \in \mathcal{F}$. Linear interpolation is then applied to find the solution,

$$\hat{\beta} := t_{\mathcal{F}} - g(\boldsymbol{x}_{\mathcal{F}})\frac{t_{\mathcal{S}} - t_{\mathcal{F}}}{g(\boldsymbol{x}_{\mathcal{S}}) - g(\boldsymbol{x}_{\mathcal{F}})} \tag{5.53}$$

If the algorithm never finds the failure region inside the grids bounds then it is assumed that the probability of failure is 0, and if it never finds the safe region then it is assumed the probability of failure is 1.

5.3 Numerical examples

In this section, the performances of the NIS, NBS and NLS algorithms on the benchmarks are analysed. Before looking at the overall results on the benchmarks, the behaviour of the algorithms on specific examples is explored. First the meatball function is considered in depth, and then the TDOF function.

5.3.1 Meatball function

The first step any algorithm based on the NMF is NInS. The behaviour on NInS was studied in the previous chapter. However, the performance functions that were considered in previous chapter had geometries that were quite easy for NInS to deal with. In contrast, the geometry of the meatball function makes the BoAs of the associated reliability problem quite difficult to model with hill valley tests. Despite this difficulty, the NInS algorithm is able to populate all the MSRs of the failure density.

Figure 5.1 shows a run of the NInS algorithm on the meatball function. Figure 5.1a joins the algorithm midway through its run. At this point, four MCOs have been run and the four MSRs have been populated by four generators. With a function like the piecewise linear function, which has geometry that NInS finds easier to model, once both of the MSRs have been populated by a generator, the non-admissible region is almost invariably the entire input space. This is an ideal situation, since the algorithm will terminate naturally, since it will not be able to generate any seeds. Returning to the meatball function, now that all of the MSRs have been populated, ideally the non-admissible region should cover the entire input space. However, as can be seen from the figure, there are small low density pockets of the input space that are still in the admissible region. As a result, the seed generation procedure is able to produce a seed, and an MCO may be run. The MCO terminates in a region of the failure density that is not a MSR.

Figure 5.1b shows the generator chain that is produced using the fifth MCO. Since the fifth MCO did not terminate in a MSR, it will almost certainly travel towards one, which is what has happened in this case. Note that the first half of the generator chain is discarded in the burn in phase, hence why there is no visible path from the seed to the MSR. It is precisely these cases that necessitate the burn in phase. Representatives have been randomly selected from the generator chain. However, all of these representatives are already in the non-admissible region, and so no new generator is added on this iteration. Recall that the NInS algorithm monitors how many consecutive iterations have passed

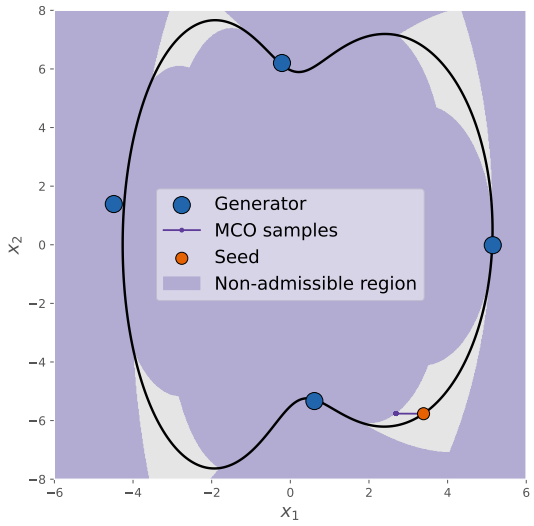
without the number of components in the joint hill valley graph increasing, and terminates the algorithm if this count becomes to high. This is the stopping condition that will typically eventually be triggered for performance functions like the meatball function.

The algorithm is now joined again at the seventh MCO run, shown in Figure 5.1c. Yet again, the seed generation routine is able to find the small pocket of admissible region so that a MCO may be started. Despite the seed being in roughly the same place as the last time, the seventh generator chain moves towards a different MSR, as shown in Figure 5.1d. This time, a new generator is found from the representatives. That is, a representative is chosen in the admissible region. However, this new generator is in the same connected component of the joint hill valley graph as the other generator in this MSR. This means that, again, the number of connected components of the joint hill valley graph does not increase. For this particular run of NInS, it is at the this iteration that it terminates.

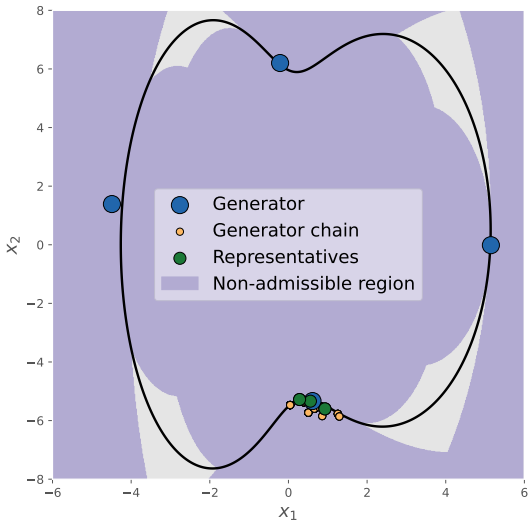
Figure 5.2 shows a run of NIS on the meatball function. The proposal NMF dataset on the first iteration is shown in Figure 5.2a. Note that despite the meatball function only having one high density MSR, all of the MSRs have roughly the same number of samples. This is because at this point in the algorithm, the proposal model has not yet been fit, and so the algorithm has no way of correcting the issues caused by the ergodicity problems of the chains. After this point the proposal model is fit, and then using that, the importance density is fit and then sampled from. The resulting importance samples are shown in Figure 5.2b. Notice that many more samples are generated in the high density MSR than the low density MSRs due to component weight correction procedure.

Figure 5.3 shows a run of NBS on the meatball function. This algorithm starts from the same place as NIS. That is, it runs NInS and then has a set of proposal failure samples that have created by the generator chains. However, the difference is that it now fits a classifier mixture distribution, rather than a vMFNM model. Figure 5.3a shows the boundaries of the classifiers in in the mixture. Note that essentially the red classifier boundaries are attempting to to model the black limit state surface. In the high density regions of the failure density, the model is quite good. That is, the red lines are a good approximation of the black line. It does not matter for the probability of failure estimation that in low density regions of the failure density that the classifiers are a poor approximation of the limit state surface. Figure 5.3b shows the importance samples generated from the classifier mixture distribution as well as the MCMC samples that are both required for bridge sampling. Note that the classifier mixture distribution is able to produce most of its samples in the high density MSR due to it adopting the corrected component weights from the proposal model.

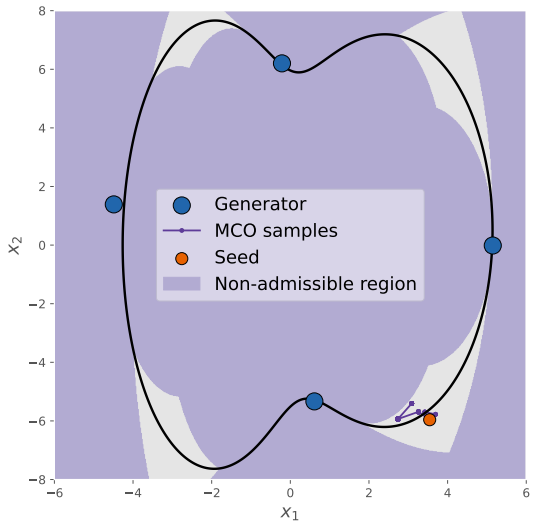
Figure 5.4 depicts a run of NLS on the meatball function. The proposal model is fit to the proposal failure samples, and then that is used to fit a vMFNM model to the failure samples. The four important directions are then derived from the vMFNM model and can be seen in Figure 5.4a. The line sampling procedure associated with the important direction that points to the high density MSR is shown in Figure 5.4b. The orthogonal samples are generated in the orthogonal complement of the important direction and then the solutions are found for the respective line search problems. Note that, although not shown here for clarity, this line sampling process is repeated for the other three important directions. Importantly, these other three directions will be assigned less samples due to their respective components in the proposal model having relatively small weights.



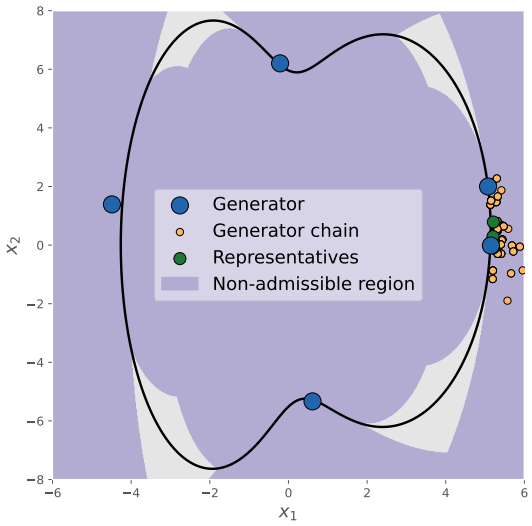
(a) Fifth MCO.



(b) Fifth generator chain.

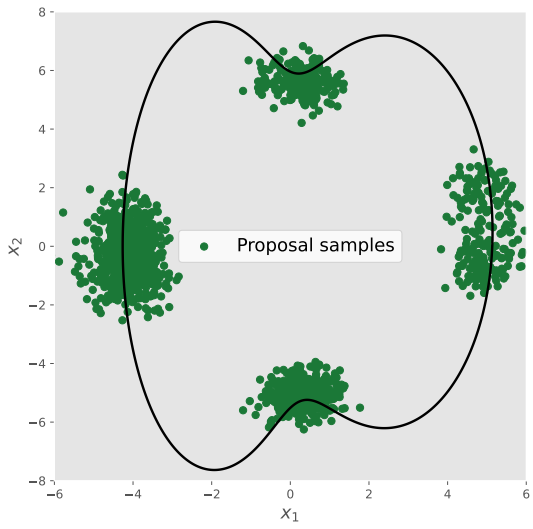


(c) Seventh MCO.

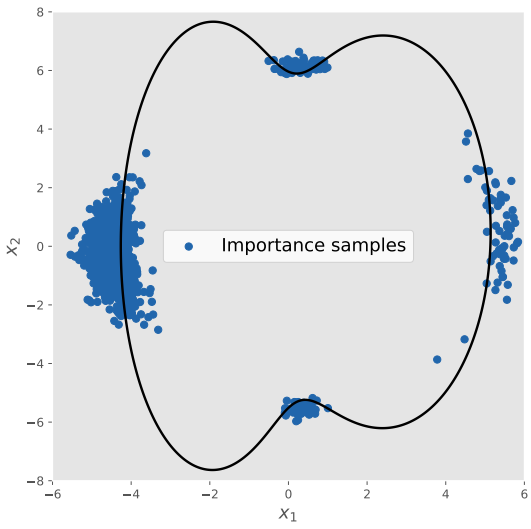


(d) Seventh generator chain.

Figure 5.1: A run of NInS on the meatball function.

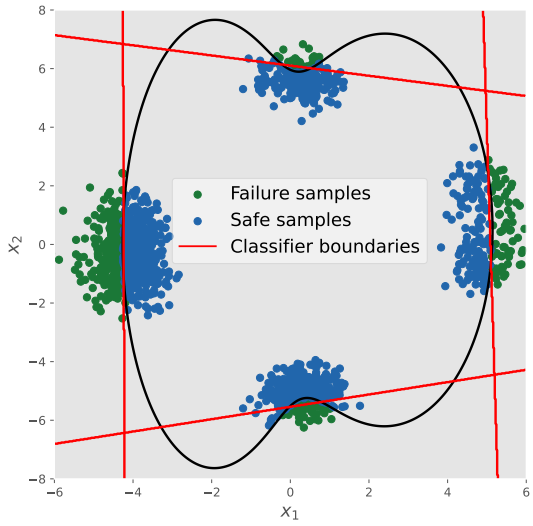


(a) Proposal failure samples.

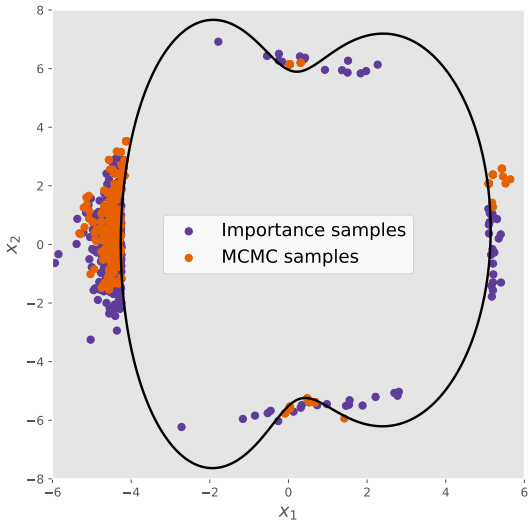


(b) Importance samples.

Figure 5.2: A run of NIS on the meatball function.

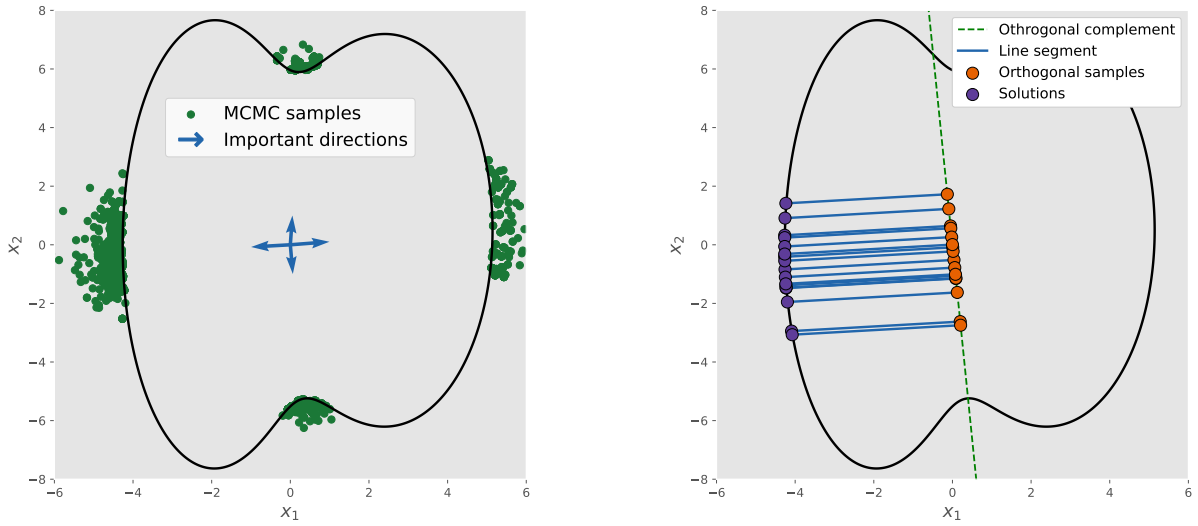


(a) Fitting classifiers of proposal failure samples.



(b) Bridge sampling with MCMC and Importance samples.

Figure 5.3: A run of NBS on the meatball function.



(a) Fitting the important directions.

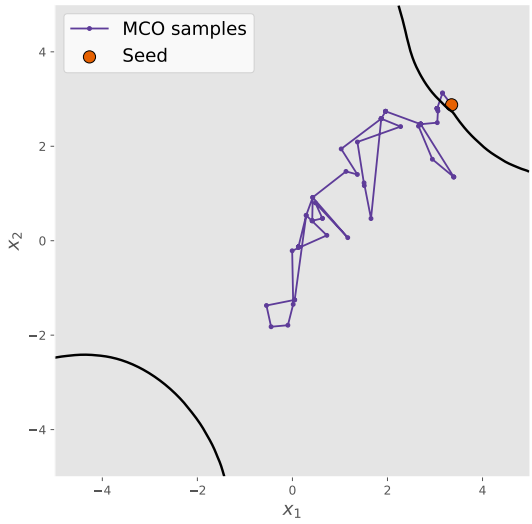
(b) Line sampling.

Figure 5.4: A run of NLS on the meatball function.

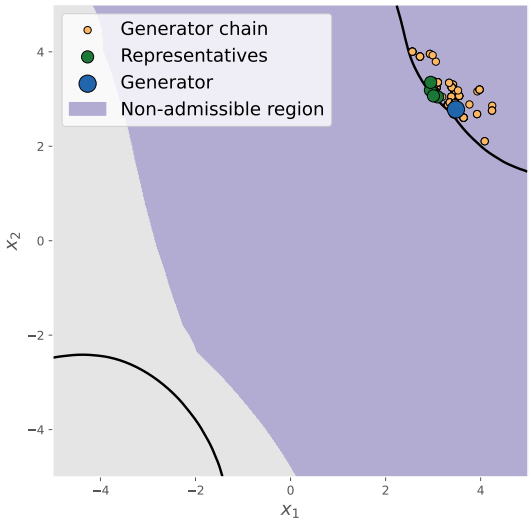
5.3.2 TDOF function

The NInS algorithm running on the TDOF performance function is shown in Figure 5.5. The TDOF performance function function has many local maxima that are not in the failure region. The path of the first MCO shown in Figure 5.5a illustrates how the MCO gets briefly trapped in the neighbourhood of one of these local maxima, but then manages to escape and reach the failure region. This is possible since it is not a large distance from the local maxima to points with higher performance. However, ultimately the distance in this context is measured with respect to the the proposal scale of the MCMC algorithm that is used. If a smaller proposal was used, it would be perfectly possible for the MCO to become trapped in local maxima. In this case, the NInS algorithm would still be able to populate the MSRs of the failure density, though it would likely to take a lot more iterations and would ultimately be more computationally expensive. Despite this nuance, the rest of the algorithm run is straight forward. Figure 5.5b shows the location of the first generator and Figures 5.5c and 5.5d shows the location of the second generator. The algorithm terminates here since it is not able to generate another seed. Note that in the upper left of Figure 5.5d, there is a very small region of admissible space. However, it is so small that the seed generation routine is not able to find it.

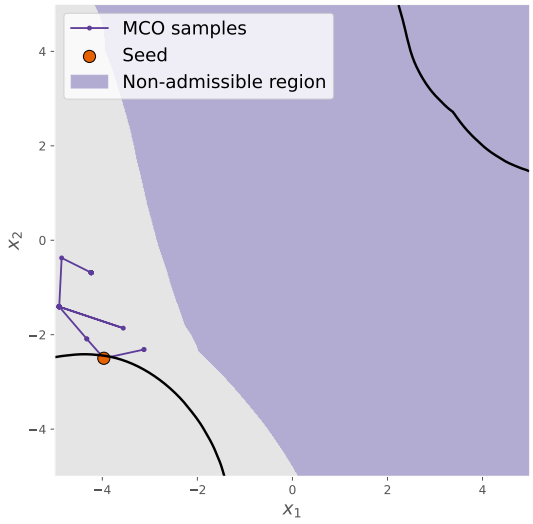
A run of NIS on the TDOF function is shown in Figure 5.6. This reliability problem has two high density MSRs with the lower right MSRs having a slightly higher probability. This means that the component weight correction routine becomes less vital. Figure 5.6a shows the proposal failure samples and Figure 5.6b shows the importance samples generated from the vMFNM model. A run of NBS on the TDOF function is depicted in Figure 5.7. Note that the classifiers of the classifier mixture distribution, shown in Figure 5.7a, do incorrectly classify small but high density parts of failure region as being in the safe region. This is precisely why bridge sampling is used with the classifier mixture rather than importance sampling, since this discrepancy violates the assumptions of importance sampling. If importance sampling were used here, the IS estimator would likely underestimate the true probability of failure. Figure 5.7b shows the samples used for a bridge sampling estimate. Note that the MCMC samples are able to populate the



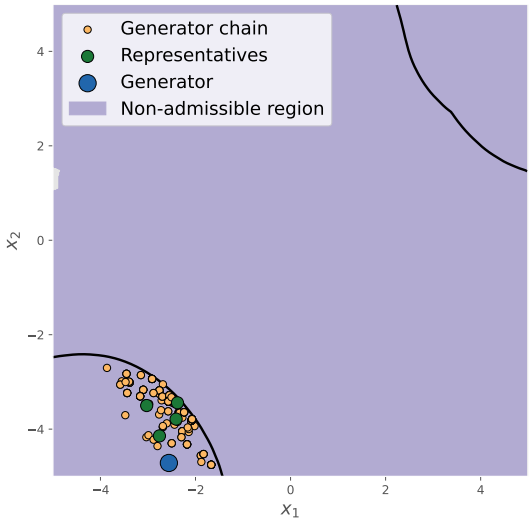
(a) First MCO.



(b) Locating the first generator.

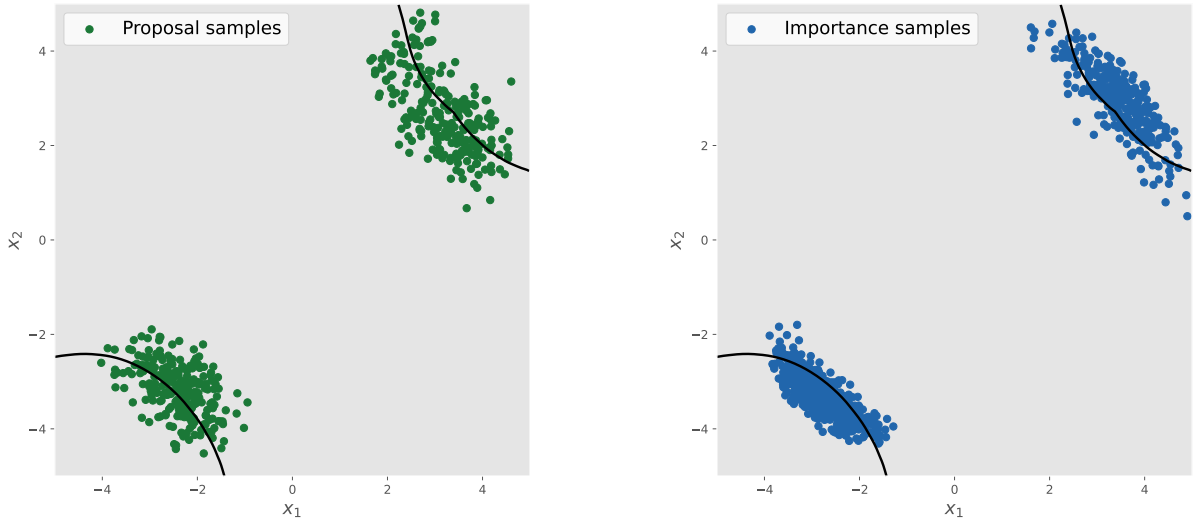


(c) Second MCO.



(d) Locating the second generator.

Figure 5.5: A run of NInS on the TDOF function.



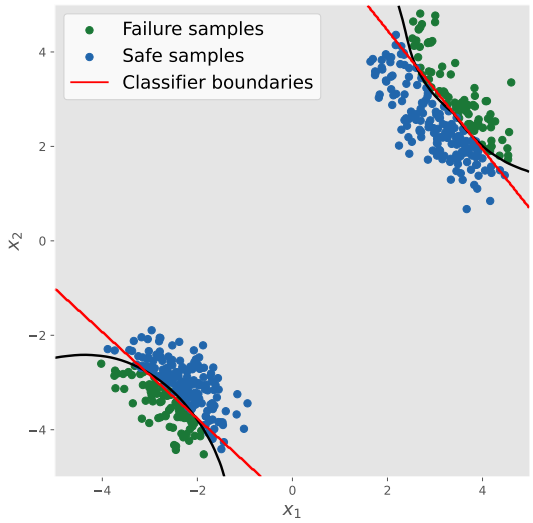
(a) Proposal failure samples.

(b) Importance samples.

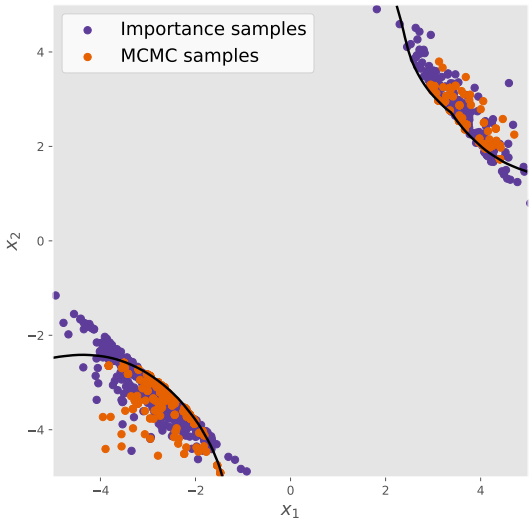
Figure 5.6: A run of NIS on the TDOF function.

parts of the failure region that the importance samples can not, and thus that information may be used by the BS estimator. Finally, Figure 5.8 shows a run of NLS on the TDOF function. The important directions are chosen using the proposal model and then failure model, as shown in Figure 5.8a. The line sampling procedure for one of the important directions is shown in Figure 5.8b. Due to the curvature of the limit state surface, the variance of the LS estimator will be larger compared to case where the limit state surface is linear.

The user defined parameters that were used for the numerical experiments on the benchmarks are given by Tables 5.1, 5.2 and 5.3 for NIS, NBS and NLS respectively. The results of the numerical experiments for NIS, NBS and NLS are given by Tables 5.4, 5.5 and 5.6 respectively. The NDP for nearly all of the numerical experiments for all three of the methods is 100%, with a few small exceptions. This verifies that, at least on these benchmarks, the NMF is a robust approach that can deal with multimodality well. It is important to note that the higher dimensional problems require more evaluations of the performance function than the lower dimensional problems. This is in contrast to NDSuS, which has a computational complexity that is largely independent of the dimension of the problem. However, the NMF methods in general do appear to be more efficient for lower dimensional problems. It should be noted that these methods tend to perform better when the failure model chosen has properties that are consistent with the failure region or failure density. For instance, the NBS implementation used here assumes that the failure region can be modelled with a mixture of hyperplanes. For the piecewise linear function, this assumption is almost perfectly correct, and the result is a very low NRMSE and relatively low computational cost. Compare this with the results of the meatball function, where the limit state surface is harder to model with straight lines.

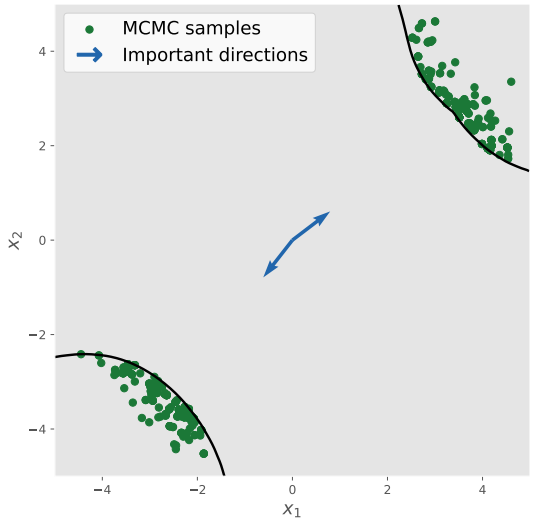


(a) Fitting classifiers of proposal failure samples.

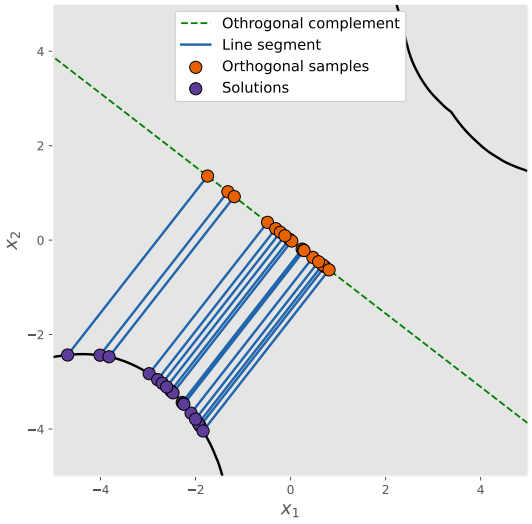


(b) Bridge sampling with MCMC and Importance samples.

Figure 5.7: A run of NBS on the TDOF function.



(a) Fitting the important directions.



(b) Line sampling.

Figure 5.8: A run of NLS on the TDOF function.

Parameter	Value
EM tol (L_{tol})	1^{-6}
EM iteration limit (L_{iter})	100
Budget multiplier (M_{budget})	40
Sample size (n_{size})	500
Max outer iteration (n_{outer})	5
Max inner iteration (n_{inner})	10
Smooth weight (w_{smooth})	0.05
Target CoV (δ_{target})	0.04

Table 5.1: NIS user-defined parameter values.

Parameter	Value
EM tolerance (L_{tol})	1^{-6}
EM iteration limit (L_{iter})	100
Budget multiplier (M_{budget})	40
Sample size (n_{size})	500
Max outer iteration (n_{outer})	5
Max inner iteration (n_{inner})	10
Smooth weight (w_{smooth})	0.05
Target CoV (δ_{target})	0.04
BS tolerance (B_{tol})	1^{-8}
BS iteration limit (B_{iter})	100
Regularisation (C_{reg})	1
Initial constant (Z_{init})	1

Table 5.2: NBS user-defined parameter values.

Parameter	Value
EM tolerance (L_{tol})	1^{-6}
EM iteration limit (L_{iter})	100
Budget multiplier (M_{budget})	20
Sample size (n_{size})	500
Max outer iteration (n_{outer})	5
Max inner iteration (n_{inner})	10
Smooth weight (w_{smooth})	0.05
Target CoV (δ_{target})	0.1

Table 5.3: NLS user-defined parameter values.

g	Dimension	Mean P_F	NRSME	NDP	Mean g evals
g_{pwl}	2	3.13×10^{-5}	0.05	100%	3.36×10^{3}
	100	3.17×10^{-5}	0.04	100%	2.56×10^{4}
g_{veh}	3	1.26×10^{-6}	0.05	100%	4.70×10^{3}
	99	1.30×10^{-6}	0.04	100%	2.85×10^{4}
g_{mix}	2	2.17×10^{-4}	0.04	100%	7.92×10^{3}
	100	2.18×10^{-4}	0.04	100%	3.79×10^{4}
g_{bs}	2	3.08×10^{-7}	0.05	100%	3.69×10^{3}
	100	3.15×10^{-7}	0.07	100%	3.28×10^{4}
g_{td}	2	2.30×10^{-5}	0.04	100%	5.34×10^{3}
	100	2.32×10^{-5}	0.05	100%	3.48×10^{4}
g_{mb}	2	1.10×10^{-5}	0.08	100%	6.44×10^{3}
	100	1.13×10^{-5}	0.12	100%	3.48×10^{4}
g_{loss}	32	4.31×10^{-3}	0.07	100%	1.58×10^{4}
	102	1.82×10^{-3}	0.04	100%	3.30×10^{4}

Table 5.4: NIS numerical experiment results.

g	Dimension	Mean P_F	NRSME	NDP	Mean g evals
g_{pwl}	2	3.20×10^{-5}	0.03	100%	2.93×10^{3}
	100	3.19×10^{-5}	0.03	100%	1.09×10^{4}
g_{veh}	3	1.30×10^{-6}	0.03	100%	3.43×10^{3}
	99	1.30×10^{-6}	0.03	100%	1.11×10^{4}
g_{mix}	2	2.19×10^{-4}	0.06	100%	9.61×10^{3}
	100	2.18×10^{-4}	0.08	100%	1.75×10^{4}
g_{bs}	2	3.09×10^{-7}	0.09	100%	7.91×10^{3}
	100	3.13×10^{-7}	0.11	100%	1.62×10^{4}
g_{td}	2	2.31×10^{-5}	0.06	100%	5.35×10^{3}
	100	2.21×10^{-5}	0.09	100%	1.31×10^{4}
g_{mb}	2	1.12×10^{-5}	0.12	99%	5.76×10^{3}
	100	1.12×10^{-5}	0.13	99%	1.67×10^{4}
g_{loss}	32	4.33×10^{-3}	0.13	100%	1.09×10^{4}
	102	1.83×10^{-3}	0.10	100%	1.19×10^{4}

Table 5.5: NBS numerical experiment results.

g	Dimension	Mean P_F	NRSME	NDP	Mean g evals
g_{pwl}	2	3.21×10^{-5}	0.02	100%	2.44×10^3
	100	3.14×10^{-5}	0.08	100%	7.30×10^3
g_{veh}	3	1.80×10^{-6}	0.44	100%	3.07×10^3
	99	1.60×10^{-6}	0.25	100%	7.73×10^3
g_{mix}	2	2.17×10^{-4}	0.04	100%	1.78×10^4
	100	2.17×10^{-4}	0.08	100%	2.63×10^4
g_{bs}	2	2.78×10^{-7}	0.13	100%	3.95×10^3
	100	2.82×10^{-7}	0.14	100%	1.58×10^4
g_{td}	2	2.35×10^{-5}	0.04	100%	4.95×10^3
	100	2.19×10^{-5}	0.12	100%	1.32×10^4
g_{mb}	2	1.09×10^{-5}	0.11	99%	5.07×10^3
	100	1.07×10^{-5}	0.11	100%	1.51×10^4
g_{loss}	32	4.17×10^{-3}	0.11	100%	9.45×10^3
	102	2.07×10^{-3}	0.16	100%	7.56×10^3

Table 5.6: NLS numerical experiment results.

6 Conclusion

This thesis set out with the aim of developing novel reliability methods that are capable of efficiently estimating the probability of failure for CRPs. Existing popular reliability methods, such as SuS, SIS and iCE are capable of solving high-dimensional black-box reliability problems. However, when reliability problems additionally exhibit certain types of multimodality, it can cause serious problems for such methods. The definition of a basin multimodal reliability problem was introduced in this work to help characterise different types of reliability problems, and also to help analyse the behaviour of reliability methods. The partitioning of the input space into the BoAs of a reliability problem proved a useful tool for visualising and predicting how a reliability method will interact with a particular reliability problem.

The definition of the BoA helped in curating a set of CRP benchmarks that could be used for comparing the performance of existing methods against any new methods that would be developed. The benchmarks were chosen so that there was a variety in the number of BoAs, in the number of failure and safe BoAs, as well as the number of BoA that contained a large portion of the failure probability. Importantly, the benchmarks contained a mixture of contrived and practical reliability problems. This because the no free lunch theorem must be considered when dealing with black-box problems. That is, caution must be taken against devising increasingly contrived reliability problems in order to violate the assumptions of some successful method, since this will always be possible to do. Also, an evaluation metric, the NDP, was introduced since commonly used reliability analysis metrics, such as the CoV, are not appropriate for measuring the performance of reliability methods when there is the potential for extremely poor estimates for the probability of failure. This set of benchmarks and evaluation metric could be used by other researchers and in future work to test the robustness of future novel reliability methods.

Defining the BoA also had a couple of other useful auxiliary benefits. Firstly, it made clear a potential strategy for dealing with CRPs. If there was some way to decompose a CRP into reliability problems with only one BoA, then existing reliability methods could be applied. Secondly, it made clear the link between niching methods and reliability analysis. Compare how similar the definitions of a NBC niche and a BoA are for instance. Many niching methods were considered during the course of this work for use in reliability analysis. Eventually, it was decided that hill valley tests would be the most appropriate, due to their effectiveness in high dimensions.

The idea behind NDSuS was to combine all the above ideas in order to make a robust reliability method that could deal with CRPs. In particular, the hill valley test was used to decompose a CRP into simpler reliability problems that could be solved with existing methods. This is an attractive strategy, since it combines well with existing literature and software implementations of reliability methods. The first step of this method introduced

a new initial sampling technique, called NInS, that is able to consistently populate the MSRs of a failure density of a CRP, even without access to any prior expert information or gradient information. It also provides a model of the failure region that is defined by a set of finite points called generators. This set of generators is particularly useful since it may be used to estimate the number of MSRs the failure density has. The NInS routine is used to decompose the reliability problem, which is then further decomposed by a smoothing routine. The NDSuS was shown to perform well on the benchmarks and importantly, did not need to increase the number of performance function evaluations for higher dimensional problems. This makes NDSuS suitable for very high dimensional problems.

The NSuS algorithm was also introduced, despite its deficiencies when compared to NDSuS. In particular, it relies on a classifier subroutine that requires more performance function evaluations in higher dimensions, and it does not deal well with performance functions like the meatball function or black swan function, where some of the BoAs are hard to reach. However, the fundamental idea underlying the NSuS algorithm, the hill valley graph partitioner, could potentially useful in some other context. The hill valley graph is able to convert a problem in high dimensional Euclidean space into a graph which is easier to analyse and visualise. The hill valley graph also builds a connection between community detection and niching which could be potentially exploited in both directions.

Finally, the NMF was introduced. This modular framework unifies techniques such as importance sampling, bridge sampling and line sampling under one scheme. One of novel ideas within this framework was to fit a model using the samples that were rejected during the MCMC algorithm sampling stage. This allowed for a component weight correction routine that could mitigate the problems caused by ergodicity issues. Another novel idea was to use bridge sampling to estimate the probability of failure directly, rather than estimating the ratio of normalising constants, which has been done before. It should be noted that this type of approach has been used for tasks in Bayesian inference. One of the advantages of using bridge sampling is that it enables the use of the classifier mixture, which itself is an attractive to model for the failure region, since it explicitly models the failure region. This approach opens up the possibility of using any binary classifier, which could be quite a powerful idea. All of the different NMF implementations were tested on the benchmarks achieved good results.

The ideas in this thesis could potentially be naturally applied in fields outside of reliability analysis. This is because reliability analysis itself has strong connections with other areas of research. For instance, reliability methods like SuS and SIS actually belong to a more general class of methods called sequential Monte Carlo samplers. These techniques are used for many tasks, but they are perhaps most commonly associated with Bayesian inference. It is not surprising then that many different formulations have been proposed for adapting SuS so that it can be used for Bayesian inference tasks [104–107]. It is also interesting to note when attempting rare event estimation from a sequential Monte Carlo perspective, precisely SuS is described in [108]. This connection implies niching techniques used in this thesis could be applied more generally to sequential Monte Carlo samplers or more specifically to tasks in Bayesian inference. The ideas in this these could also be potentially applied to black-box optimisation problems. Indeed, the NInS algorithm could likely be applied in this context with very few adaptions and the hill valley graph can be directly interpreted as a niching technique.

The work in this thesis has focused largely on the hill valley test. This choice was

limited by the fact that the CRP is high dimensional. There are however other niching techniques that can work in high dimensions that were not fully explored. For example, in SuS, the history of a sample could be considered and used to sort samples into niches. That is, it could be traced back to which seed it was generated from, and then which seed that seed was generated from and so on. Such an idea is similar to the niching technique of speciation. If the reliability problem is not high dimensional, or it is possible to transform the problem into a lower dimensional space, then many more niching techniques that rely on the Euclidean metric could be applied.

One way of summarising the work in this thesis is an attempt to mitigate the ergodicity issues encountered by MCMC algorithms when a target density has multiple MSRs. However, this is not the only type of pathological behaviour that a target density can exhibit. In some cases, a high density region of a target density can pinch into a region of high curvature. When these type of problems occur in Bayesian inference, the typical approach is to parametrise the target density [109]. A similar approach might be worth considering for reliability analysis.

It seems that ultimate goal when considering black-box reliability problems is to describe some type of meta-algorithm. That is, some algorithm that can determine which type of reliability method is most appropriate for the problem at hand and then to apply it. Phrased another way, there likely needs to be some type of procedure that can validate the assumptions that a reliability method uses. For example, blindly applying a method like SuS to a black-box reliability problem can go wrong as has been seen. In addition to this, often when SuS goes wrong there is no way of the user knowing. The NInS procedure could be viewed as an algorithm that is able to check the implicit assumptions of SuS. The one piece of information that is available regarding a black-box reliability problem is its dimension. This can be used by a meta-algorithm to decide which method is appropriate before any analysis has started at all. Indeed, a simple meta-algorithm using the methods of this paper could simply be to use NDSuS above a certain dimension, and to use NMF otherwise. In any case, it has been shown in this work that niching techniques, and in particular hill valley tests, are a powerful tool for dealing with multimodality in high dimensions and so should likely be considered in some capacity when designing such meta-algorithms.

Bibliography

[1] N. K. Boots and P. Shahabuddin, "Simulating ruin probabilities in insurance risk processes with subexponential claims," in *Proceeding of the 2001 Winter Simulation Conference (Cat. No.01CH37304)*, vol. 1, pp. 468–476 vol.1, Dec. 2001.

[2] M. El Masri, J. Morio, and F. Simatos, "Improvement of the cross-entropy method in high dimension for failure probability estimation through a one-dimensional projection without gradient estimation," *Reliability Engineering & System Safety*, vol. 216, p. 107991, Dec. 2021.

[3] A. Agarwal, S. De Marco, E. Gobet, and G. Liu, "Study of new rare event simulation schemes and their application to extreme scenario generation," *Mathematics and Computers in Simulation*, vol. 143, pp. 89–98, Jan. 2018.

[4] F. Ragone and F. Bouchet, "Rare Event Algorithm Study of Extreme Warm Summers and Heatwaves Over Europe," *Geophysical Research Letters*, vol. 48, no. 12, p. e2020GL091197, 2021.

[5] J. Wouters, R. K. H. Schiemann, and L. C. Shaffrey, "Rare Event Simulation of Extreme European Winter Rainfall in an Intermediate Complexity Climate Model," *Journal of Advances in Modeling Earth Systems*, vol. 15, no. 4, p. e2022MS003537, 2023.

[6] J. Morio, D. Jacquemart, M. Balesdent, and J. Marzat, "Optimisation of interacting particle systems for rare event estimation," *Computational Statistics & Data Analysis*, vol. 66, pp. 117–128, Oct. 2013.

[7] P. Heidelberger, "Fast simulation of rare events in queueing and reliability models," in *Performance Evaluation of Computer and Communication Systems* (L. Donatiello and R. Nelson, eds.), (Berlin, Heidelberg), pp. 165–202, Springer, 1993.

[8] B. Potrzeszcz-Sut, "Reliability analysis of shell truss structure by hybrid Monte Carlo method," *Journal of Theoretical and Applied Mechanics*, vol. 58, pp. 469–482, Apr. 2020.

[9] M. Rashki, M. Miri, and M. A. Moghaddam, "A simulation-based method for reliability based design optimization problems with highly nonlinear constraints," *Automation in Construction*, vol. 47, pp. 24–36, Nov. 2014.

[10] S. Lee, "Monte Carlo simulation using support vector machine and kernel density for failure probability estimation," *Reliability Engineering & System Safety*, vol. 209, p. 107481, May 2021.

[11] A. Sharma and C. S. Manohar, "Modified replica exchange-based MCMC algorithm for estimation of structural reliability based on particle splitting method," *Probabilistic Engineering Mechanics*, vol. 72, p. 103448, Apr. 2023.

[12] C. Ling, Z. Lu, K. Feng, and X. Zhang, "A coupled subset simulation and active learning kriging reliability analysis method for rare failure events," *Structural and Multidisciplinary Optimization*, vol. 60, pp. 2325–2341, Dec. 2019.

[13] F. Uribe, I. Papaioannou, Y. M. Marzouk, and D. Straub, "Cross-entropy-based importance sampling with failure-informed dimension reduction for rare event simulation," June 2020.

[14] A. Robens-Radermacher and J. F. Unger, "Efficient structural reliability analysis by using a PGD model in an adaptive importance sampling schema," *Advanced Modeling and Simulation in Engineering Sciences*, vol. 7, p. 29, June 2020.

[15] Z.-m. Jiang, D.-C. Feng, H. Zhou, and W.-F. Tao, "A recursive dimension-reduction method for high-dimensional reliability analysis with rare failure event," *Reliability Engineering & System Safety*, vol. 213, p. 107710, Sept. 2021.

[16] R. Melchers, "Search-based importance sampling," *Structural Safety*, vol. 9, pp. 117–128, Dec. 1990.

[17] M. Rashki, "SESC: A new subset simulation method for rare-events estimation," *Mechanical Systems and Signal Processing*, vol. 150, p. 107139, Mar. 2021.

[18] S.-K. Au and Y. Wang, *Engineering Risk Assessment with Subset Simulation*. Wiley, 1 ed., Apr. 2014.

[19] K. Breitung, "The geometry of limit state function graphs and subset simulation: Counterexamples," *Reliability Engineering & System Safety*, vol. 182, pp. 98–106, Feb. 2019.

[20] H. J. Kinnear and F. A. DiazDelaO, "Niching subset simulation," *Probabilistic Engineering Mechanics*, vol. 79, p. 103729, Jan. 2025.

[21] M. Rosenblatt, "Remarks on a Multivariate Transformation," *The Annals of Mathematical Statistics*, vol. 23, pp. 470–472, Sept. 1952.

[22] R. Lebrun and A. Dutfoy, "Do Rosenblatt and Nataf isoprobabilistic transformations really differ?," *Probabilistic Engineering Mechanics*, vol. 24, pp. 577–584, Oct. 2009.

[23] A. Nataf, "Determination des distributions dont les marges sont donnees.," *C. R. Acad. Sci. Paris*, no. 225, pp. 42–43, 1962.

[24] R. Lebrun and A. Dutfoy, "An innovating analysis of the Nataf transformation from the copula viewpoint," *Probabilistic Engineering Mechanics*, vol. 24, pp. 312–320, July 2009.

[25] R. Lebrun and A. Dutfoy, "A generalization of the Nataf transformation to distributions with elliptical copula," *Probabilistic Engineering Mechanics*, vol. 24, pp. 172–178, Apr. 2009.

[26] A. M. Hasofer and N. C. Lind, "Exact and Invariant Second-Moment Code Format," *Journal of the Engineering Mechanics Division*, vol. 100, pp. 111–121, Feb. 1974.

[27] R. Rackwitz and B. Flessler, "Structural reliability under combined random load sequences," *Computers & Structures*, vol. 9, pp. 489–494, Nov. 1978.

[28] K. Breitung, "40 years FORM: Some new aspects?," *Probabilistic Engineering Mechanics*, vol. 42, pp. 71–77, Oct. 2015.

[29] K. Breitung, *Asymptotic Approximations for Probability Integrals*. No. v.1592 in Lecture Notes in Mathematics Ser, Berlin, Heidelberg: Springer Berlin / Heidelberg, 1994.

[30] L. Katafygiotis and K. Zuev, "Geometric insight into the challenges of solving high-dimensional reliability problems," *Probabilistic Engineering Mechanics*, vol. 23, pp. 208–218, Apr. 2008.

[31] B. Fiessler, H.-J. Neumann, and R. Rackwitz, "Quadratic Limit States in Structural Reliability," *Journal of the Engineering Mechanics Division*, vol. 105, pp. 661–676, Aug. 1979.

[32] M. A. Valdebenito, H. J. Pradlwarter, and G. I. Schuëller, "The role of the design point for calculating failure probabilities in view of dimensionality and structural nonlinearities," *Structural Safety*, vol. 32, pp. 101–111, Mar. 2010.

[33] K. Breitung, "SORM, Design Points, Subset Simulation, and Markov Chain Monte Carlo," *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A: Civil Engineering*, vol. 7, Dec. 2021.

[34] K. Breitung, "The return of the design points," *Reliability Engineering & System Safety*, vol. 247, p. 110103, July 2024.

[35] M. Matsumoto and T. Nishimura, "Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator," *ACM Transactions on Modeling and Computer Simulation*, vol. 8, pp. 3–30, Jan. 1998.

[36] M. O'Neill, "PCG: A Family of Simple Fast Space-Efficient Statistically Good Algorithms for Random Number Generation," *Proceedings of the ACM on Programming Languages*, 2014.

[37] G. E. P. Box and M. E. Muller, "A Note on the Generation of Random Normal Deviates," *The Annals of Mathematical Statistics*, vol. 29, pp. 610–611, June 1958.

[38] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of State Calculations by Fast Computing Machines," *The Journal of Chemical Physics*, vol. 21, pp. 1087–1092, June 1953.

[39] W. K. Hastings, "Monte Carlo Sampling Methods Using Markov Chains and Their Applications," *Biometrika*, vol. 57, no. 1, pp. 97–109, 1970.

[40] A. Gelman, W. R. Gilks, and G. O. Roberts, "Weak convergence and optimal scaling of random walk Metropolis algorithms," *The Annals of Applied Probability*, vol. 7, pp. 110–120, Feb. 1997.

[41] S. P. Meyn and R. L. Tweedie, *Markov Chains and Stochastic Stability*. London: Springer, 1993.

[42] S. K. Au and J. L. Beck, "Estimation of small failure probabilities in high dimensions by subset simulation," *Probabilistic Engineering Mechanics*, vol. 16, pp. 263–277, Oct. 2001.

[43] P. S. Koutsourelakis, H. J. Pradlwarter, and G. I. Schuëller, "Reliability of structures in high dimensions, part I: Algorithms and applications," *Probabilistic Engineering Mechanics*, vol. 19, pp. 409–417, Oct. 2004.

[44] C. Song and R. Kawai, "Monte Carlo and variance reduction methods for structural reliability analysis: A comprehensive review," *Probabilistic Engineering Mechanics*, vol. 73, p. 103479, July 2023.

[45] S. Song, Z. Lu, and H. Qiao, "Subset simulation for structural reliability sensitivity analysis," *Reliability Engineering & System Safety*, vol. 94, pp. 658–665, Feb. 2009.

[46] A. Guillal, M. E. A. Ben Seghier, A. Nourddine, J. A. F. O. Correia, Z. Bt Mustaffa, and N.-T. Trung, "Probabilistic investigation on the reliability assessment of mid- and high-strength pipelines under corrosion and fracture conditions," *Engineering Failure Analysis*, vol. 118, p. 104891, Dec. 2020.

[47] S. S. Kar and L. B. Roy, "A Comparative Study on Reliability Analysis of Cohesive Soil Slope using Subset Simulation and Other Methods," *U.Porto Journal of Engineering*, vol. 8, pp. 135–155, Apr. 2022.

[48] R. Schneider, S. Thöns, and D. Straub, "Reliability analysis and updating of deteriorating systems with subset simulation," *Structural Safety*, vol. 64, pp. 20–36, Jan. 2017.

[49] Y. Houmadi, M. Y. Cherif Benmoussa, W. N. E. H. Cherifi, and D. D. Rahal, "Probabilistic analysis of consolidation problems using subset simulation," *Computers and Geotechnics*, vol. 124, p. 103612, Aug. 2020.

[50] I. Papaioannou, W. Betz, K. Zwirglmaier, and D. Straub, "MCMC algorithms for Subset Simulation," *Probabilistic Engineering Mechanics*, vol. 41, pp. 89–103, July 2015.

[51] S. L. Cotter, G. O. Roberts, A. M. Stuart, and D. White, "MCMC Methods for Functions: Modifying Old Algorithms to Make Them Faster," *Statistical Science*, vol. 28, pp. 424–446, Aug. 2013.

[52] I. Papaioannou, C. Papadimitriou, and D. Straub, "Sequential importance sampling for structural reliability analysis," *Structural Safety*, vol. 62, pp. 66–75, Sept. 2016.

[53] M. B. Mehni and M. B. Mehni, "Reliability analysis with cross-entropy based adaptive Markov chain importance sampling and control variates," *Reliability Engineering & System Safety*, vol. 231, p. 109014, Mar. 2023.

[54] Z. Wang and J. Song, "Cross-entropy-based adaptive importance sampling using von Mises-Fisher mixture for high dimensional reliability analysis," *Structural Safety*, vol. 59, pp. 42–52, Mar. 2016.

[55] N. Kurtz and J. Song, "Cross-entropy-based adaptive importance sampling using Gaussian mixture," *Structural Safety*, vol. 42, pp. 35–44, May 2013.

[56] S. Geyer, I. Papaioannou, and D. Straub, "Cross entropy-based importance sampling using Gaussian densities revisited," *Structural Safety*, vol. 76, pp. 15–27, Jan. 2019.

[57] I. Papaioannou, S. Geyer, and D. Straub, "Improved cross entropy-based importance sampling with a flexible mixture model," *Reliability Engineering & System Safety*, vol. 191, p. 106564, Nov. 2019.

[58] D. Wolpert and W. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, pp. 67–82, Apr. 1997.

[59] M. Rashki and M. G. R. Faes, "No-Free-Lunch Theorems for Reliability Analysis," *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A: Civil Engineering*, vol. 9, Sept. 2023.

[60] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. The MIT Press, Apr. 1992.

[61] A. H. Wright, "Genetic Algorithms for Real Parameter Optimization," in *Foundations of Genetic Algorithms* (G. J. E. Rawlins, ed.), vol. 1, pp. 205–218, Elsevier, Jan. 1991.

[62] I. Rechenberg, *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Problemata, 15, Stuttgart-Bad Cannstatt: Frommann-Holzboog, 1973.

[63] H.-P. Schwefel, *Evolutionsstrategie Und Numerische Optimierung*. PhD thesis, Jan. 1975.

[64] N. Hansen and A. Ostermeier, "Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation," in *Proceedings of IEEE International Conference on Evolutionary Computation*, pp. 312–317, May 1996.

[65] N. Hansen and A. Ostermeier, "Completely Derandomized Self-Adaptation in Evolution Strategies," *Evolutionary Computation*, vol. 9, pp. 159–195, June 2001.

[66] N. Hansen and S. Kern, "Evaluating the CMA Evolution Strategy on Multimodal Test Functions," in *Parallel Problem Solving from Nature - PPSN VIII* (X. Yao, E. K. Burke, J. A. Lozano, J. Smith, J. J. Merelo-Guervós, J. A. Bullinaria, J. E. Rowe, P. Tiňo, A. Kabán, and H.-P. Schwefel, eds.), (Berlin, Heidelberg), pp. 282–291, Springer, 2004.

[67] R. Storn and K. Price, "Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces,"

[68] R. Storn and K. Price, "Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces," *Journal of Global Optimization*, vol. 11, pp. 341–359, Dec. 1997.

[69] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, (Perth, WA, Australia), pp. 1942–1948, IEEE, 1995.

[70] H.-S. Li, "Subset simulation for unconstrained global optimization," *Applied Mathematical Modelling*, vol. 35, pp. 5108–5120, Oct. 2011.

[71] C. Elegbede, "Structural reliability assessment based on particles swarm optimization," *Structural Safety*, vol. 27, pp. 171–186, Apr. 2005.

[72] E. Zio and N. Pedroni, "An optimized Line Sampling method for the estimation of the failure probability of nuclear passive systems," *Reliability Engineering & System Safety*, vol. 95, pp. 1300–1313, Dec. 2010.

[73] H. Zhao, M. Zhao, and C. Zhu, "Reliability-based optimization of geotechnical engineering using the artificial bee colony algorithm," *KSCE Journal of Civil Engineering*, vol. 20, pp. 1728–1736, July 2016.

[74] C. Zhong, M. Wang, C. Dang, W. Ke, and S. Guo, "First-order reliability method based on Harris Hawks Optimization for high-dimensional reliability analysis," *Structural and Multidisciplinary Optimization*, vol. 62, pp. 1951–1968, Oct. 2020.

[75] J. Jafari-Asl, S. Ohadi, M. E. A. Ben Seghier, and N.-T. Trung, "Accurate Structural Reliability Analysis Using an Improved Line-Sampling-Method-Based Slime Mold Algorithm," *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A: Civil Engineering*, vol. 7, p. 04021015, June 2021.

[76] D. J. Cavicchio, *Adaptive Search Using Simulated Evolution*. Ph.d. thesis, University of Michigan, Ann Arbor, MI, USA, May 1970.

[77] D. Jong and K. Alan, "Analysis of the behavior of a class of genetic adaptive systems," 1975.

[78] D. E. Goldberg and J. Richardson, "Genetic algorithms with sharing for multimodal function optimization," in *Proceedings of the Second International Conference on Genetic Algorithms on Genetic Algorithms and Their Application*, (USA), pp. 41–49, L. Erlbaum Associates Inc., Oct. 1987.

[79] A. Petrowski, "A clearing procedure as a niching method for genetic algorithms," in *Proceedings of IEEE International Conference on Evolutionary Computation*, pp. 798–803, May 1996.

[80] J.-P. Li, M. E. Balazs, G. T. Parks, and P. J. Clarkson, "A Species Conserving Genetic Algorithm for Multimodal Function Optimization," *Evolutionary Computation*, vol. 10, pp. 207–234, Sept. 2002.

[81] M. Bessaou, A. Pétrowski, and P. Siarry, "Island Model Cooperating with Speciation for Multimodal Optimization," in *PPSN VI: Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*, pp. 437–446, Sept. 2000.

[82] Xiaodong Li, "Niching Without Niching Parameters: Particle Swarm Optimization Using a Ring Topology," *IEEE Transactions on Evolutionary Computation*, vol. 14, pp. 150–169, Feb. 2010.

[83] X. Yin and Noël. Germay, "A Fast Genetic Algorithm with Sharing Scheme Using Cluster Analysis Methods in Multimodal Function Optimization," in *Artificial Neural Nets and Genetic Algorithms* (R. F. Albrecht, C. R. Reeves, and N. C. Steele, eds.), (Vienna), pp. 450–457, Springer, 1993.

[84] M. Preuss, L. Schönemann, and M. Emmerich, "Counteracting genetic drift and disruptive recombination in (μ, $+\lambda$)-EA on multimodal fitness landscapes," in *GECCO 2005 - Genetic and Evolutionary Computation Conference*, pp. 865–872, June 2005.

[85] F. Klawonn, F. Höppner, and B. Jayaram, "What are Clusters in High Dimensions and are they Difficult to Find?," in *Clustering High–Dimensional Data* (F. Masulli, A. Petrosino, and S. Rovetta, eds.), (Berlin, Heidelberg), pp. 14–33, Springer, 2015.

[86] R. Ursem, "Multinational evolutionary algorithms," in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, (Washington, DC, USA), pp. 1633–1640, IEEE, 1999.

[87] C. G. Bucher, "Adaptive sampling — an iterative fast Monte Carlo procedure," *Structural Safety*, vol. 5, pp. 119–126, June 1988.

[88] A. Der Kiureghian and T. Dakessian, "Multiple design points in first and second-order reliability," *Structural Safety*, vol. 20, pp. 37–49, Jan. 1998.

[89] A. Tabandeh, G. Jia, and P. Gardoni, "A review and assessment of importance sampling methods for reliability analysis," *Structural Safety*, vol. 97, p. 102216, July 2022.

[90] G. Jia, A. Tabandeh, and P. Gardoni, "A density extrapolation approach to estimate failure probabilities," *Structural Safety*, vol. 93, p. 102128, Nov. 2021.

[91] H.-S. Li, Y.-Z. Ma, and Z. Cao, "A generalized Subset Simulation approach for estimating small failure probabilities of multiple stochastic responses," *Computers & Structures*, vol. 153, pp. 239–251, June 2015.

[92] W. Xia and Z. Liao, "Enhanced generalized subset simulation with multiple importance sampling for reliability estimation," *Computers & Structures*, vol. 313, p. 107741, June 2025.

[93] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, p. P10008, Oct. 2008.

[94] F. Parés, D. G. Gasulla, A. Vilalta, J. Moreno, E. Ayguadé, J. Labarta, U. Cortés, and T. Suzumura, "Fluid Communities: A Competitive, Scalable and Diverse Community Detection Algorithm," in *Complex Networks & Their Applications VI* (C. Cherifi, H. Cherifi, M. Karsai, and M. Musolesi, eds.), Studies in Computational Intelligence, (Cham), pp. 229–240, Springer International Publishing, 2018.

[95] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical Review E*, vol. 76, p. 036106, Sept. 2007.

[96] A. Abdi and M. Kaveh, "Performance comparison of three different estimators for the Nakagami m parameter using Monte Carlo simulation," *IEEE Communications Letters*, vol. 4, pp. 119–121, Apr. 2000.

[97] S. Sra, "A short note on parameter approximation for von Mises-Fisher distributions: And a fast implementation of I s (x)," *Computational Statistics*, vol. 27, pp. 177–190, Mar. 2012.

[98] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, (Portland, Oregon), pp. 226–231, AAAI Press, Aug. 1996.

[99] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, vol. 5.1, pp. 281–298, University of California Press, Jan. 1967.

[100] P. Beaurepaire, H. A. Jensen, G. I. Schuëller, and M. A. Valdebenito, "Reliability-based optimization using bridge importance sampling," *Probabilistic Engineering Mechanics*, vol. 34, pp. 48–57, Oct. 2013.

[101] A. M. Overstall and J. J. Forster, "Default Bayesian model determination methods for generalised linear mixed models," *Computational Statistics & Data Analysis*, vol. 54, pp. 3269–3288, Dec. 2010.

[102] X.-L. Meng and W. H. Wong, "Simulating Ratios of Normalizing Constants Via a Simple Identity: A Theoretical Exploration," *Statistica Sinica*, vol. 6, no. 4, pp. 831–860, 1996.

[103] S. Frühwirth-Schnatter, "Estimating marginal likelihoods for mixture and Markov switching models using bridge sampling techniques*," *The Econometrics Journal*, vol. 7, pp. 143–167, June 2004.

[104] M. Chiachio, J. L. Beck, J. Chiachio, and G. Rus, "Approximate Bayesian Computation by Subset Simulation," *SIAM Journal on Scientific Computing*, vol. 36, pp. A1339–A1358, Jan. 2014.

[105] D. Straub and I. Papaioannou, "Bayesian Updating with Structural Reliability Methods," *Journal of Engineering Mechanics*, vol. 141, p. 04014134, Mar. 2015.

[106] F. A. DiazDelaO, A. Garbuno-Inigo, S. K. Au, and I. Yoshida, "Bayesian updating and model class selection with Subset Simulation," *Computer Methods in Applied Mechanics and Engineering*, vol. 317, pp. 1102–1121, Apr. 2017.

[107] W. Betz, I. Papaioannou, and D. Straub, "Adaptive variant of the BUS approach to Bayesian updating: 9th International Conference on Structural Dynamics, EURODYN 2014," *Proceedings of the 9th International Conference on Structural Dynamics, EURODYN 2014*, pp. 3021–3028, 2014.

[108] F. Cérou, P. Del Moral, T. Furon, and A. Guyader, "Sequential Monte Carlo for rare event estimation," *Statistics and Computing*, vol. 22, pp. 795–808, May 2012.

[109] M. Betancourt, "A conceptual introduction to hamiltonian monte carlo," *arXiv preprint arXiv:1701.02434*, July 2018. Preprint available at arXiv.