TECHNISCHE UNIVERSITÄT MÜNCHEN

TUM School of Engineering and Design

Reliability Analysis and Design of Flight Control Systems Using Surrogate Models

Dalong Shi

Vollständiger Abdruck der von der TUM School of Engineering and Design der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

genehmigten Dissertation.

Vorsitzender:	Prof. Dr.-Ing. Boris Lohmann
Prüfer der Dissertation:	1. Prof. Dr.-Ing. Florian Holzapfel
	2. Prof. Dr. Markus Zimmermann

Die Dissertation wurde am 22.06.2022 bei der Technischen Universität München eingereicht und durch die TUM School of Engineering and Design am 16.11.2022 angenommen.

Abstract

In recent years, the world has witnessed a significant paradigm shift of certification criteria from design-based to performance-based and probabilistic requirements in aviation. For events that can lead to severe safety issues, the acceptable failure probabilities are usually very small. This poses a serious challenge for designers to tune the control parameters in order to satisfy the rare-event probabilistic requirements.

This thesis first presents a probabilistic performance-oriented control design optimization approach, called *reliability-based control optimization* (RBCO). It performs control design by solving a chance-constrained optimization problem that satisfies probabilistic design requirements. To ensure the precise treatment of rare-event chance constraints, the reliability analysis is conducted within the optimization loop. As a consequence, the proposed RBCO framework can be used to search for control parameters that fulfill the probabilistic requirements with a formal guarantee.

Subset simulation (SuS) is an accurate and efficient method for reliability analysis, but it still requires a large number of true model evaluations to achieve sufficient accuracy. To reduce this demand, the second part of this thesis incorporates different types of surrogate modeling techniques into SuS. Global surrogates, namely, *polynomial chaos expansion* (PCE) and *response surface method* (RSM), are first combined with SuS. Adaptive PCE or RSM is applied to progressively refine the surrogate at each subset level. Aiming at balancing the local and global prediction performance, an experimental design strategy is developed for the surrogate refinement. However, global surrogates are more applicable to weakly nonlinear applications. To this end, a local surrogate called *moving least-squares* (MLS) is then implemented to handle highly nonlinear problems. An active learning strategy is proposed to efficiently enrich the training set. For high-dimensional problems, a dimensionality reduction method is introduced to filter out unnecessary expansion items. Then, a more flexible surrogate called *kriging* is used to assist the SuS method which detects both the global trend and the local variability. Adaptive trend detection and experimental design strategies are proposed to further enhance the kriging modeling efficiency.

Finally, this thesis compares the performance of the introduced surrogate-accelerated SuS techniques to the conventional SuS method, and gives a recommendation of approaches to tackle different problems. The case studies demonstrate the real-life applicability of the proposed methods.

Kurzfassung

In den letzten Jahren vollzog sich ein Paradigmenwechsel im Bereich der Luftfahrt von auslegungsbasierten Anforderungen hin zu leistungsbasierten und probabilistischen Anforderungen. Fehlerereignisse, welche zu ernsthaften Sicherheitsproblemen führen, haben in der Regel eine sehr geringe zulässige Ausfallwahrscheinlichkeit. Dies stellt die Entwickler von Luftfahrtsystemen vor die große Herausforderung die Reglerparameter so abzustimmen, dass diese die probabilistischen Anforderungen in Bezug auf die Eintrittswahrscheinlichkeit seltener Ereignisse erfüllen.

In dieser Arbeit wird zunächst ein probabilistischer, leistungsorientierter Ansatz zur Optimierung des Steuerungsentwurfs vorgestellt, der als *reliability-based control optimization* (RBCO) bezeichnet wird. Durch das Lösen eines Optimierungsproblem mit Zufallsbeschränkungen, ermöglicht RBCO den Entwurf von Flugsteuerungsystemen welche die probabilistischen Anforderungen erfüllen. Um die Einhaltung der Anforderungen an die zulässige Eintrittswahrscheinlichkeit für seltene Ereignisse zu gewährleisten, wird die Zuverlässigkeitsanalyse innerhalb der Optimierungsschleife durchgeführt. Der vorgeschlagene RBCO-Ansatz kann folglich verwendet werden, um nach Kontrollparametern zu suchen, die die probabilistischen Anforderungen mit einer formalen Garantie erfüllen.

Subset Simulation (SuS) ist eine genaue und effiziente Methode der Zuverlässigkeitsanalyse. Trotz ihrer Effizienz erfordert diese Methode immer noch eine große Anzahl echter Modellbewertungen, um eine ausreichende Genauigkeit zu erreichen. Um diesen Bedarf weiter zu reduzieren, werden im zweiten Teil dieser Arbeit verschiedene Arten von Ersatzmodell-Modellierungstechniken in die SuS-Methodik integriert. Globale Ersatzmodelle, nämlich *polynomial chaos expansion* (PCE) und *Response Surface Method* (RSM), werden zunächst mit SuS kombiniert. Adaptive PCE oder RSM werden angewandt, um das Ersatzmodell auf jeder Teilmengenebene schrittweise zu verfeinern. Mit dem Ziel, ein Gleichgewicht zwischen lokaler und globaler Vorhersageleistung herzustellen, wird eine experimentelle Planungsstrategie für die Verfeinerung der Ersatzmodelle entwickelt. Im Allgemeinen eignen sich globale Ersatzmodelle jedoch eher für schwach nichtlineare Anwendungen. Um stark nichtlineare Probleme zu behandeln, wird ein lokales Ersatzmodell namens *moving least-squares* (MLS) implementiert. Hierbei wird eine aktive Lernstrategie vorgeschlagen, um die Trainingsmenge effizient anzureichern. Für hochdimensionale Probleme wird eine Dimensionalitätsreduktions Methode angewendet, um weniger relevante Dimensionen herauszufiltern. Im Anschluss wird ein flexibleres Ersatzmodell namens *kriging* verwendet, um die SuS-Methode zu unterstützen. Kriging Ersatzmodelle berücksichtigen sowohl den globalen Trend, als auch lokale Variabilität. Mit dem Ziel, die Effizienz der Kriging-Modellierung weiter zu verbessern, werden adaptive Strategien für die Trenderkennung und die Trainingsdatenanreicherung implementiert.

Schließlich vergleicht diese Arbeit die Ergebnisse der vorgestellten Ersatzmodell-beschleunigten SuS-Techniken mit derer der konventionellen SuS-Methode. Darüber hinaus wird eine Empfehlung gegeben, wie die Ansätze verwendet werden können, um verschiedene anspruchsvolle Probleme zu bewältigen. Fallstudien verdeutlichen die Anwendbarkeit der vorgeschlagenen Methoden in der Praxis.

Acknowledgment

This thesis summarizes the results during my doctoral research studies at the Institute of Flight System Dynamics (FSD) of the Technical University of Munich (TUM). The thesis would not have been possible without the help of many people. Firstly, I would like to express my sincere gratitude to my supervisor Prof. Florian Holzapfel for the continuous support of my research, for his guidance, patience, and wise insights. I am very grateful to the committee members, Prof. Markus Zimmermann and Prof. Boris Lohmann, for their valuable feedback and commitment.

I greatly appreciate my colleagues, David Löbl, Nils Mumm, Florian Schwaiger, David Braun, Xiang Fang, Jun Shi, Xiaolong Wang, Shen Kang, Jiannan Zhang, Guolong Jiang, Haichao Hong, Yahao Cheng, Yuhang Yun, Mingkai Wang, Chinmaya Mishra, Patrick Piprek, Johannes Diepolder, Benedikt Grüter, Tuğba Akman, Rasmus Steffensen, Agnes Steinert, Hangxu Li, Zhidong Lu, Ruichen He, and Prof. Shuguang Zhang for their support and fruitful discussions with them that lay the foundation of the success of this work.

Finally, I would like to thank my parents and my wife for their unconditional love and support. I am extremely thankful to my wife Siyuan Liu, who has stood by me through all my triumphs and travails.

Garching, June 15, 2022 Dalong Shi

Contents

List of Figures

List of Tables

Acronyms

AIS	Adaptive Importance Sampling
AK-IS	Adaptive Kriging-based Importance Sampling
AK-MCS	Adaptive Kriging-based Monte Carlo Simulation
AK-SuS	Adaptive Kriging-based Subset Simulation
ani.	Anisotropic
ANN	Artificial Neural Network
BLUP	Best Linear Unbiased Predictor
c.o.v.	Coefficient of Variation
CDF	Cumulative Distribution Function
CFCL	Critical Flight Control Law
CI	Confidence Interval
CLT	Central Limit Theorem
CS-AWO	Certification Specifications for All Weather Operations
EASA	European Aviation Safety Agency
EFF	Expected Feasibility Function
EGRA	Efficient Global Reliability Analysis
emp.	Empirical
eVTOL	Electric Vertical Take-Off and Landing
FAA	Federal Aviation Administration
FM	Fourth-Moment
FORM	First-Order Reliability Method
FSD	Institute of Flight System Dynamics
GA	Genetic Algorithm
i.i.d.	Independent and Identically Distributed
IS	Importance Sampling
ISD	Importance Sampling Density
KPLS	Partial Least-Squares-Based Kriging
kriging-MCS	Kriging-Accelerated Monte Carlo Simulation
kriging-SuS	Kriging-Accelerated Subset Simulation
KRR	Kernel Ridge Regression
LAR	Least Angle Regression

LLN	Law of Large Numbers
LOO	Leave-One-Out
MCMC	Markov Chain Monte Carlo
MCS	Monte Carlo Simulation
MH	Metropolis-Hastings
MIS	Multiple Importance Sampling
MLE	Maximum Likelihood Estimation
MLS	Moving Least-Squares
MLS-MCS	Moving Least-Squares-Accelerated Monte Carlo Simulation
MLS-SuS	Moving Least-Squares-Accelerated Subset Simulation
MPC	Model Predictive Control
MPFP	Most Probable Failure Point
NN	Neural Network
OLS	Ordinary Least-Squares
PC-kriging	Polynomial-Chaos Kriging
PCE	Polynomial Chaos Expansion
PCE-MCS	Polynomial Chaos Expansion-Based Monte Carlo Simulation
PDF	Probability Density Function
PID	Proportional-Integral-Derivative
PLS	Partial Least-Squares
PS-SuS	Polynomial Surrogate-Based Subset Simulation
RBCO	Reliability-Based Control Optimization
RBDO	Reliability-Based Design Optimization
RBRDO	Reliability-Based Robust Design Optimization
RDO	Robust Design Optimization
RMS	Root Mean Square
RR	Ridge Regression
RSM	Response Surface Method
SORM	Second-Order Reliability Method
SuS	Subset Simulation
SVM	Support Vector Machine
SVR	Support Vector Regression
TUM	Technical University of Munich
UDM	Uniform Design Method
VC	Vapnik-Chervonenkis
WLS	Weighted Least-Squares

Symbols and Indices

Symbols

$\boldsymbol{A}$	Coefficient matrix
$\mathcal{B}$	Beta distribution
$B(\cdot,\cdot)$	Beta function
b	Subset threshold
$c(\cdot)$	Deterministic function
c_v	Coefficient of variation
$\mathcal{D}$	Design space
d	Expansion order
$\mathbb{E}[\cdot]$	Expectation operator
$\mathcal{F}$	Failure domain
$\mathcal{F}^0$	Limit-state surface
$\mathcal{F}^{\mathrm{c}}$	Safe domain
$F(\cdot)$	Cumulative distribution function
$f(\cdot)$	Probability density function
$\boldsymbol{f(\cdot)}$	State transition function
$f(\cdot\vert\cdot)$	Transition density
$\mathcal{G}$	Gamma distribution
$G(\cdot)$	Transformed limit-state function
$g(\cdot)$	Limit-state function or performance function
$\boldsymbol{h(\cdot)}$	Output function
$\boldsymbol{I}$	Identity matrix
$I(\cdot)$	Indicator function
$\boldsymbol{k}$	Control parameters
k_M	Gain margin
$k(\cdot,\cdot)$	Kernel function
$\boldsymbol{L}$	Lower triangular matrix from the Cholesky decomposition
$\mathcal{L}(\cdot;\cdot)$	Likelihood function
m	Number of subsets
$\boldsymbol{m}$	Multi-index

$m_{\hat{g}}(\cdot)$	Kriging predicted value
N	Number of samples
$\mathcal{N}$	Normal distribution
N_0	Number of initial training samples
N_{call}	Number of calls to the true model
N_F	Number of samples in the failure domain at the last level of SuS
N_s	Number of seeds
N_t	Cardinality of the training set $\mathcal{T}$
n	Number of uncertain parameters
n_t	Number of training samples in the support domain $\mathcal{S}$
n_z	Vertical load factor
$\mathcal{O}$	Big O notation
P_F	Failure probability
P_j	Conditional probability
$\mathbb{P}[\cdot]$	Probability operator
p	Number of expansion items
p_0	Conditional probability
p_a	Acceptance probability
p_d	Training sample discard ratio
$p(\cdot\|\cdot)$	Proposal density
Q	Number of multi-dimensional quadrature nodes
$\boldsymbol{Q}$	Indicator matrix
q	Number of 1-dimensional quadrature nodes
q	Pitch rate
$q(\cdot)$	Importance sampling density
$\boldsymbol{R}$	Correlation matrix
$\mathbb{R}$	Set of real numbers
$R(\cdot,\cdot)$	Autocorrelation function
S	Sensitivity index
$\mathcal{S}$	Support domain
$s_{\hat{g}}(\cdot)$	Standard deviation of the kriging prediction
$\boldsymbol{T}$	Transition matrix
$\mathcal{T}$	Training set
$T(\cdot)$	Isoprobabilistic transformation
t_r	Rise time of the step response
t_{total}	Execution time
t_{train}	Training time
$\mathcal{U}$	Uniform distribution
$U(\cdot)$	U-function
$\boldsymbol{u}$	Input vector

$\mathcal{V}$	Validation set
$\mathrm{Var}[\cdot]$	Variance operator
$\boldsymbol{W}$	Weight matrix
w	Weight
w_z	Vertical wind velocity
$w(\cdot)$	Weight function
$\boldsymbol{x}$	State vector
$\boldsymbol{y}$	Output vector
$z(\cdot)$	Gaussian process
α	Confidence coefficient
α	Angle of attack
$\boldsymbol{\alpha}$	Expansion coefficients
β	Acceptable failure probability
β	Reliability index
$\boldsymbol{\beta}$	Expansion coefficients for kernel ridge regression
$\Gamma(\cdot)$	Gamma function
γ	Squared norm of an orthogonal polynomial
δ_{ij}	Kronecker function
δ_j	Coefficient of variation of $\hat{P}_j$
$\delta(\cdot)$	Dirac function
ε	Error
η	Elevator deflection
$\boldsymbol{\eta}$	Hyperparameters
Θ	Random variable
Θ	Pitch angle
$\boldsymbol{\Theta}$	Random vector
θ	Realization of the random variable Θ
$\boldsymbol{\theta}$	Realization of the random vector $\boldsymbol{\Theta}$
$\boldsymbol{\vartheta}$	Principal component
κ	Kurtosis
$\boldsymbol{\kappa}(\cdot)$	Kernel function vector
λ	Ridge parameter
μ	Mean or expected value
$\mu(\cdot)$	Mean function or trend
Ξ	Standard random variable
$\boldsymbol{\Xi}$	Independent standard random vector
ξ	Realization of the random variable Ξ
$\boldsymbol{\xi}$	Realization of the random vector $\boldsymbol{\Xi}$
$\boldsymbol{\xi}^*$	Most probable failure point
$\boldsymbol{\pi}$	Stationary probability mass

$\pi(\cdot)$	Stationary distribution
ρ	Correlation coefficient
$\boldsymbol{\Sigma}$	Covariance matrix
σ	Standard deviation
$\sigma\%$	Overshoot of the step response
τ	Skewness
$\boldsymbol{\Phi}$	Sample matrix
$\Phi(\cdot)$	CDF of the standard normal distribution
ϕ_M	Phase margin
$\phi(\cdot)$	PDF of the standard normal distribution
$\phi(\cdot)$	Univariate basis function
$\boldsymbol{\Psi}$	Experimental matrix
$\boldsymbol{\psi}(\cdot)$	Multivariate basis functions
Ω	Support of the random variable Θ
$\boldsymbol{\Omega}$	Support of the random vector $\boldsymbol{\Theta}$
ω_y	Pitch rate

Indices

cmd	Commanded value
est	Estimated value
lb	Lower bound
max	Maximum value
min	Minimum value
r	Non-training sample
t	Training sample
ref	Reference value
req	Required value
ub	Upper bound
$\hat{\square}$	Estimated quantity

Chapter 1

Introduction

1.1 Background and Motivation

Safety has always been a vital concern for aviation. An aircraft is a typical safety-critical system, whose failure may lead to unaffordable consequences such as substantial property damage, severe injury, and even loss of life. Driven by the ever increasing safety considerations, stringent requirements are imposed on the development of aircraft by certification authorities. According to the convention on international civil aviation, "*every aircraft engaged in international navigation shall be provided with a certificate of airworthiness issued or rendered valid by the State in which it is registered*" [1, p. 14]. For example, the European Aviation Safety Agency (EASA) is responsible for the certification of aircraft in the EU and for some European non-EU countries, whereas the Federal Aviation Administration (FAA) is the counterpart in the U.S. The certificate testifies that the aircraft meets the safety requirements set by the corresponding authority.

In recent years, there has been a significant paradigm shift in certification criteria. The requirements are no longer formulated as deterministic limits, but acceptable probabilities for specified failure events. For instance, according to the certification specifications for all weather operations (CS-AWO) published by the EASA, the probability of exceeding certain safety limits must be below the thresholds during the phase of automatic landing [2, p. 15]. The detailed requirements are given in Table 1.1, wherein the acceptable probabilities are very small, typically of the order 10^{-5} to 10^{-8}. Another example is the minimum level of system reliability prescribed by the CS-25 certification specifications for large aircraft [3, pp. 2-F-47–2-F-50]. It highly depends on the severity of the outcome, as illustrated in Tables 1.2 and 1.3. If a failure condition only leads to minor effects, such as a slight reduction in safety margins or functional capabilities, a slight increase in crew workload, and some physical discomfort to passengers, the admissible failure probability is 10^{-3} per flight hour. By contrast, for a catastrophic failure condition which can cause a hull loss and multiple fatalities, the probability of occurrence must be less than 10^{-9} per flight hour.

Table 1.1: *Touchdown requirements for automatic landing [2, p. 15].*

Touchdown performance	Probability threshold	
	Average[1]	Limit
Longitudinal touchdown earlier than 60 m from the threshold.	10^{-6}	10^{-5}
Longitudinal touchdown beyond 823 m from the threshold.	10^{-6}	N/A
Longitudinal touchdown beyond 914 m from the threshold.	N/A	10^{-5}
Lateral touchdown more than 21 m from the runway centerline, assuming a 45 m runway.	10^{-6}	10^{-5}
Exceed *sink rate* for structural limit load.	10^{-6}	10^{-5}
Exceed *bank angle* such that wing tip touches ground before wheels.	10^{-8}	10^{-7}
Exceed *lateral velocity* or *slip angle* for structural load limit.	10^{-6}	10^{-5}

Table 1.2: *Relationship between the severity of failure effects and the classification of failure conditions [3, p. 2-F-49].*

Severity of failure effects	Effect on aeroplane	No effect on operational capabilities or safety	Slight reduction in functional capabilities or safety margins	Significant reduction in functional capabilities or safety margins	Large reduction in functional capabilities or safety margins	Normally with hull loss
	Effect on occupants excluding flight crew	Inconvenience	Physical discomfort	Physical distress, possibly including injuries	Serious or fatal injury to a small number of passengers or cabin crew	Multiple fatalities
	Effect on flight crew	No effect on flight crew	Slight increase in workload	Physical discomfort or a significant increase in workload	Physical distress or excessive workload impairs ability to perform tasks	Fatalities or incapacitation
Classification of failure conditions		No safety effect	Minor	Major	Hazardous	Catastrophic

[1]The ‘Average’ column: all variables vary according to their distributions; The ‘Limit’ column: one variable takes its most adverse value while others vary according to their distributions.

Table 1.3: *Relationship between the classification of failure conditions and acceptable probabilities [3, p. 2-F-50].*

Classification of failure conditions	No safety effect	Minor	Major	Hazardous	Catastrophic
Allowable qualitative probability	No probability requirement	Probable	Remote	Extremely remote	Extremely improbable
Allowable quantitative probability: average probability per flight hour on the order of:	No probability requirement	$< 10^{-3}$	$< 10^{-5}$	$< 10^{-7}$	$< 10^{-9}$

In the context of specifications with such small acceptable failure probabilities, it is of great importance to assess the reliability of a function and all subsystems and components contributing to its provision. Additionally, control parameters must be tuned to satisfy functional requirements while maintaining the fulfillment of rare probabilistic safety and certification requirements. This requires repeated evaluations of the closed-loop system reliability. These demands motivate us to develop accurate and efficient reliability analysis methods as well as reliability-guaranteed control design strategies.

1.2 State of the Art

This section first introduces the state of the art of reliability analysis methods. Afterwards, an overview of design strategies under uncertainty is given. In the end, how the current control design methods deal with uncertainties is summarized.

1.2.1 Reliability Analysis Methods

In the presence of uncertainties, a system may react outside of its nominal range, which means the system encounters a failure. Given the model of uncertainties, reliability analysis aims to quantitatively assess the probability of such failures [4, p. 1]. In this subsection, all the presented approaches deal with black-box problems. They rely on the evaluation of a computational model, without the knowledge of its inner structure. These methods can be classified into three categories, i.e., approximation, simulation, and surrogate-based approaches.

1.2.1.1 Approximation Approaches

Approximation methods seek to approximate the limit-state function locally at a reference point. The *first-order reliability method* (FORM) is an elementary approximation approach. It was introduced by Hasofer and Lind [5] in 1974 to overcome the computational inefficiency of *Monte Carlo simulation* (MCS) and successfully applied to structural problems. In this method, reliability is evaluated based on the first-order Taylor expansion of the limit-state function at the so-called *most probable failure point* (MPFP) (also known as design point) in the standard space [6, Ch. 7]. Specifically, the FORM consists of three steps, i.e., transform the random variables from the original space into the standard space, search for the MPFP, and linearize the limit-state function at the MPFP and compute the approximation of failure probability. Note that the FORM assumes a linear (or weakly nonlinear) limit-state function with a unique MPFP. It may lead to a large estimation bias if these assumptions are not satisfied.

To enhance the accuracy of the FORM, the *second-order reliability method* (SORM) [7] has been developed. This method is a second-order refinement of the FORM. After the MPFP is identified, the limit-state function is approximated by the second-order Taylor expansion. By this means, in comparison with the FORM, the SORM achieves better accuracy at the cost of computational expense.

Based on the FORM and SORM, several variants have been proposed to improve the accuracy or efficiency. A first-order third-moment reliability method [8] was presented by introducing the skewness of the transformed reliability margin into the reliability index. Similarly, the moments up to the third-order are used to derive the second-order third-moment reliability index in [9]. More recently, based on the fourth-moment standardization function [10], the work in [11] proposed an explicit second-order fourth-moment reliability index. These methods extend the FORM or SORM to address more general reliability analysis problems. To maintain both high efficiency and accuracy, a SORM with first order efficiency was proposed in [12]. With the aim of improving the accuracy of reliability analysis, the work in [13] presented a direct SORM without parabolic approximation of the fitted quadratic surface.

Although the FORM and SORM are very efficient, they may not achieve accurate estimations especially for complicated and highly nonlinear limit-state functions [14]. These two methods only provide low-order approximations, thus introduce large estimation errors in case of strong nonlinearity. In addition, for functions with multiple design points, the FORM and SORM may result in biased failure probability estimates.

1.2.1.2 Simulation Approaches

For complex systems with highly nonlinear limit-state functions, researchers are likely to employ numerical simulation methods to estimate the failure probability without making any hypothesis on the complexity of the limit-state functions. The common idea of this type of methods is to generate samples in the output space and then find an estimate of the failure probability.

The standard MCS method is one of the most widely used ways to solve reliability analysis problems. Since it was originally developed in [15], it has been applied in many research fields, such as statistics, physics, computer science, finance, and engineering. The basic idea behind MCS is to draw samples according to the distribution of input random variables and compute the percentage of outputs falling in the failure region. The main strength of MCS is the strong robustness due to the fact that its accuracy does not depend on the geometry of the failure domain and the dimension of random variables [16]. However, on the other hand, MCS suffers from inefficiency in estimating small probabilities. To estimate a failure probability $P_F = 10^{-m}$ with a 10% *coefficient of variation* (c.o.v.), about 10^{m+2} samples are required [4, p. 11]. The inefficiency essentially stems from the fact that most of the generated samples are not in the failure domain. Nevertheless, all sampling-based methods for estimating rare events are based on MCS [16].

The efficiency of MCS can be enhanced by means of variance reduction techniques, which aim at increasing the estimation precision with given simulation effort [17, Ch. 5]. *Importance sampling* (IS) [18, 19] is one of the most popular variance reduction strategies. The fundamental concept of IS is to draw samples from the *importance sampling density* (ISD) which is different from the distribution of interest, so as to generate samples that lie more frequently in the failure domain. Its efficiency critically depends on the choice of the ISD. However, choosing a good ISD that results in a low-variance IS estimator is a challenging task, especially in high dimensions [16, 19, 20]. Thus, IS is often inefficient for high-dimensional problems. A variety of sophisticated strategies have been proposed to improve the performance of IS. *Multiple importance sampling* (MIS) methods (e.g. [21, 22]) employ a set of ISDs for the generation of samples. Because these schemes avoid entrusting a single ISD, they generally provide more robust results. Moreover, *adaptive importance sampling* (AIS) methods (e.g. [23–25]) enhance the efficiency by iteratively updating the parameters of ISD based on the past samples.

Subset simulation (SuS) [26, 27] is an advanced stochastic simulation method for reliability analysis which is based on *Markov chain Monte Carlo* (MCMC) [28, Ch. 6]. SuS estimates the rare failure probability by converting it into the product of a series of much larger conditional probabilities. In comparison with MCS, SuS gains far higher efficiency when achieving the same accuracy. Moreover, unlike IS, SuS does not suffer from the curse of dimensionality [16]. The accuracy and efficiency of SuS depend on the ability of the implemented MCMC algorithm. Several MCMC sampling algorithms have been developed motivated by the goal of improving the performance of SuS. Au and Beck [26] introduced the modified *Metropolis-Hastings* (MH) algorithm based on a component-wise sample generation to overcome the low acceptance ratio of the original MH sampler for high-dimensional problems. With the aim of addressing the same issue, the MH strategy with repeated sample generation was proposed by Santoso et al. [29]. To reduce the correlation between the states of the Markov chain in the component-wise MH algorithm [26], Zuev and Katafygiotis [30] integrated the MH approach with delayed rejection [31] into the component-wise MH algorithm. Additionally, Papaioannou et al. [32] proposed the Gaussian conditional sampling method, in which the candidate samples generated from the proposal *probability density function* (PDF) always differ from the current sample.

Due to the superiority of SuS, it has been successfully applied in various fields, such as fire risk analysis [33], structural reliability analysis [26, 34], wind [35] and nuclear engineering [36]. Recently, pioneered by the *Institute of Flight System Dynamics* (FSD) of the *Technical University of Munich* (TUM), SuS has been utilized in many aerospace applications. The probability of conflict between aircraft is estimated in [37, 38]. A safety-critical backup controller is assessed using SuS in [39]. The work in [40] implements SuS to evaluate the hover performance of an *electric vertical take-off and landing* (eVTOL) aircraft.

1.2.1.3 Surrogate-Based Approaches

Although SuS is tailored to estimate the rare failure probability, it still requires at least thousands of limit-state function evaluations to achieve sufficient estimation accuracy. Therefore, conducting SuS for computationally expensive limit-state functions can be a time-consuming task. In this context, many researchers resort to surrogate modeling techniques to further enhance the efficiency of SuS.

Surrogate Modeling Techniques

Before reviewing surrogate-based approaches, a brief introduction of surrogate models for reliability analysis and uncertainty propagation is given. Surrogate modeling in essence makes useful predictions based on assumptions and limited information [41]. Actually, in everyday life we attempt to save time and make predictions according to our assumptions and experience. This is similar as what a surrogate model does. To be specific, a surrogate model is a cheap-to-evaluate approximation model that intends to mimic the behavior of a costly-to-evaluate true model [42]. A common assumption for the surrogate modeling techniques discussed in this thesis is that the limit-state function is continuous.

Response surface method (RSM) [43] is an elementary and probably the most widely used surrogate modeling technique. It assumes that the shape of the function can be approximated by the chosen polynomial expansion, and exploits polynomial regression to build a global surrogate model. However, this assumption may be unfounded in many applications, which indicates that polynomial approximation may be not flexible enough to fit the true model (e.g. [44]). In addition, this method suffers from the curse of dimensionality, since the number of training samples required for the regression generally grows exponentially with the increase of dimension. Hence, RSM is unsuitable for highly nonlinear or high-dimensional problems. Despite this, a polynomial response surface may be an attractive choice for functions with weak nonlinearity, few dimensions, or where data is very cheap to obtain. In particular, the expansion coefficients reflect the effect of each term.

Moving least-squares (MLS) [45] consists in constructing lower-order local approximations and exhibits an attractive tradeoff between regression and interpolation. The *weighted least-squares* (WLS) approach is used in the region of interest, and this region "moves" with the point to be predicted. Since the calculation must be performed at every prediction, the MLS method requires more computational expense in comparison with RSM. On the other hand, the drawbacks of RSM are mitigated by this local approximation technique. The locality of the approximation is governed by the weight function. However, the parameters of the weight function are often hard to tune.

Polynomial chaos expansion (PCE) [46, 47] projects the function output onto a space spanned by polynomials that are orthogonal with regard to the input probability measure. This method can be interpreted as an extension of RSM. The main difference between them is that the inputs of RSM are deterministic variables, whereas in PCE, the inputs are random variables [48, Ch. 3]. A prominent property of the PCE approach is that it guarantees the convergence of the approximation, which means the approximation error reduces with the increase of the polynomial order. However, the convergence cannot be guaranteed by the RSM method. The strategies for computing the PCE coefficients can be divided into intrusive and non-intrusive ones according to whether they are coupled with the limit-state function. Stochastic Galerkin is a typical intrusive method, whereas popular non-intrusive techniques include pseudo projection, interpolation approach, and least-squares method [49, 50].

The surrogate models mentioned above are all extensions of rigid polynomial, and thus lacking of flexibility. This drawback can be avoided by kernel-based surrogate models such as kriging and support vector machines. *Kriging* [51], also known as *Gaussian process modeling*, is a statistical interpolation method based on Gaussian process governed by prior covariance. Depending on the type of trend which refers to the mean of a kriging model, a different naming is given to the surrogate model, i.e., simple kriging, ordinary kriging, and universal kriging. Since the kriging prediction is considered as a realization of a Gaussian random variable, the standard deviation of the prediction is derived. This built-in error measure is the major advantage of kriging over other surrogate models mentioned in this thesis. The generalization ability of kriging relies heavily on the proper choice of its hyperparameters. However, finding the optimal hyperparameters is a nontrivial task. This time-consuming parameter selection stage limits this technique to problems of low dimensionality (usually limited to around 20 [41]). Due to this expense, kriging is more applicable to limit-state functions that are particularly computationally intensive.

Support vector machines (SVMs) [52] are supervised learning techniques that aim at constructing an optimal hyperplane in the sense of the *Vapnik-Chervonenkis* (VC) dimension, which can be interpreted as the complexity of problems. SVMs were initially used in the context of binary classification and then extended to regression. Instead of solving nonlinear problems directly, SVMs seek to map the sample points into the feature space by means of kernels and then transform the nonlinear problems to linear ones. In this way, SVMs are able to efficiently handle nonlinear functions. Similar with kriging, the generalization ability of SVMs also depends on the suitable parameterization of the kernel, which is a challenging task. Moreover, compared with surrogate models based on rigid polynomial, SVMs necessitate longer training time.

Artificial neural networks (ANNs), usually simply called *neural networks* (NNs) [53], are another type of surrogate model inspired by the biological neural networks. A NN consists of a collection of connected nodes or neurons. A transfer or activation function is built in within each neuron and each connection is assigned a weight. The architecture of NNs is usually determined empirically. This is not trivial especially for complex applications. Given the selected architecture, the training of NNs consists in finding the optimal combination of biases and weights that minimizes a certain cost function. Various global and local algorithms are developed to accomplish that. Back-propagation is probably the most popular method among them.

Surrogate-Based Reliability Analysis Strategies

A common idea of surrogate-based approaches is to build a surrogate model and then replace the true model evaluations with the predictions of the surrogate model. The constructed surrogate model is usually used in conjunction with aforementioned approximation methods or simulation methods.

The quadratic polynomial response surface was first utilized as a surrogate in structural mechanics in [54]. Inspired by this seminal work, a variety of variants [55–58] have been proposed with different basis functions (with or without cross terms) and training points selection strategies. In all these works, the quadratic approximation is constructed around the design point. In [59, 60], weighted regression is applied to build the response surface, in which weights are allocated to the training samples according to their distances from the failure surface. The work in [61] employs a forward selection procedure to determine the most important regression terms with respect to statistical criteria. Recently, an adaptive RSM that incorporates several techniques including model selection, cross validation, and weighted regression is presented in [62].

To overcome the weaknesses of RSM, MLS is applied for the approximation of the region of interest in structural reliability analysis in [63, 64]. In [65], the MLS surrogate is not only employed for prediction, but also involved in the selection process of sample points. Besides the weight factor induced by MLS, the weights based on the distances from the MPFP are also considered in [66]. The design point is updated successively within an iterative MLS strategy in [67]. The work in [68] tunes the parameters of the weight function in the MLS approach to explore better surrogate performance.

The intrusive use of PCE in reliability analysis was originally investigated in [69, 70]. Later on, the application of PCE has blown up with the emergence of non-intrusive methods. To be specific, the regression method was developed and implemented in [71, 72]. In reliability analysis, RSM is used to fit the limit-state function in the vicinity of the design point, whereas PCE is usually performed in the entire space and thus may not be accurate enough in the tail of the output distribution. Focusing on this problem, a shifted and windowed PCE was proposed in [73] to enhance the accuracy of the estimation in the failure domain. The work in [74] derives a local error estimator and then an active learning scheme so as to efficiently estimate the failure probability. To reduce the number of PCE bases in high dimensions, the most significant expansion terms are adaptively detected in [75, 76]. With the same purpose, the work in [77] introduces *least angle regression* (LAR) to find the optimal basis functions for sparse PCE.

Kriging was first employed for reliability problems in [78]. Here, a fixed experimental design strategy called progressive lattice sampling is presented for the construction of the surrogate model. In [79], a sample enriching scheme based on the current design point is implemented for kriging. These early works do not take advantage of the variance information of kriging until the development of active learning approaches. Bichon et al. [80] introduced the *efficient global reliability analysis* (EGRA), where the surrogate accuracy is only enhanced in the vicinity of the limit-state and the indication on the vicinity is provided by the proposed *expected feasibility function* (EFF). The *adaptive kriging-based Monte Carlo simulation* (AK-MCS) approach was presented by Echard et al. [81], in which the U-function has been devised to represent the reliability index on the risk of misclassification. This work is a cornerstone of a variety of methods exploiting active learning kriging. For instance, replacing the MCS part of the approach with IS or SuS leads respectively to AK-IS [82] or AK-SuS [83]. The coupling of active learning kriging and SuS was further explored in [84–86]. An alternative extension of AK-MCS is *polynomial-chaos kriging* (PC-kriging) [87] which employs PCE to model the global trend and utilizes kriging to capture the local behavior.

SVMs have not been introduced to the field of reliability analysis until the early 21st century. The work in [88] considers reliability analysis as a classification problem and exploits SVM classifier to divide samples into two groups, i.e., safe and failure points. Similarly, the classifier is implemented for the assessment of failure probability for stochastic finite element models in [89]. Li et al. [90] has presented both SVM-based MCS and SVM-based FORM to assess structural safety. Hurtado [91] combined SVMs with IS and selected samples in the margin of SVMs as worth-testing points. Basudhar et al. [92] investigated the use of SVM classifier for complex limit-state functions with multiple failure domains or discontinuous responses. Bourinet et al. [93] adopted SVM classifier to solve the classification problem between subset levels in SuS. Alternatively, *support vector regression* (SVR), which leverages the principle of SVMs to tackle regression problems, are utilized for reliability analysis. In [94], the authors employed a joint use of SVR and adaptive Markov chain simulation for the assessment of reliability. Bourinet [95] developed a 3-phase adaptive SVR for the accurate estimation of rare-event failure probability. Later, the results are further explored for non-smooth limit-state functions in [96].

NN was initially applied to reliability analysis in [97], where a NN is trained for the prediction of the critical load factor and the failure probability is then estimated using MCS and IS. Hurtado and Alvarez [98] conducted a comprehensive comparison of different NN options for reliability assessment. Deng et al. [99] replaced the original performance function with the NN surrogate, whereupon MCS, FORM, and SORM are performed based on this surrogate model. Cheng [100] proposed two NN-based genetic algorithms (GAs) to reduce the computational effort of the traditional GA. In [101], the aforementioned NN-based GA is further developed by implementing the *uniform design method* (UDM). Papadopoulos et al. [102] integrated the robust NN surrogate into SuS to enhance the efficiency of this advanced simulation technique. More recently, Xiao et al. [103] introduced a novel adaptive sequential sampling strategy for NN. This method, in principle, also fits other surrogate models.

1.2.2 Design Under Uncertainty

Design optimization, which is widely used in engineering, consists in searching for the optimal design variables to achieve certain objective or/and satisfy the given constraint. According to the types of objective function and constraint function, the design approaches can be generally classified as in Table 1.4. The strategies that do not take uncertainties into account are deterministic methods. Due to the limited knowledge of system model, no one can ensure that the system will perform exactly as the nominal behavior. In this context, a deterministic design may lead to unsatisfactory results. A better solution is to account for uncertainties directly in the design optimization formulation.

Robust design optimization (RDO) and *reliability-based design optimization* (RBDO) are the most common schemes considering the impact of uncertainty. RDO [105] establishes a design framework that maximizes the performance while minimizing the sensitivity of

Table 1.4: *Classification of design approaches [104].*

		Type of objective function		
		None	Deterministic	Uncertain
Type of constraint function	None	— —	Unconstrained optimization	Robust design optimization (RDO)
	Deterministic	Admissible design	Constrained optimization	Constrained RDO
	Uncertain	Reliable design	Reliability-based design optimization (RBDO)	Reliability-based robust design optimization (RBRDO)

performance. This insensitive solution guarantees that the design is robust to the probable variations of uncertain parameters. The objective function is usually formulated in terms of the mean value and standard deviation of the system performance.

The objective of RBDO [106] is to optimize the cost function under the fulfillment of probabilistic (chance) constraints, thus seeking to find the best compromise between safety and cost. In comparison with RDO, RBDO shifts the focus from objective function to constraints and considers the hard limits from a probabilistic viewpoint. The methods used to solve the RBDO problem are generally classified into three categories: the two-level, the mono-level (also known as single loop), and the decoupled [106, 107]. The two-level approach is a direct and effective way to tackle this problem. It consists of two loops, of which the inner one estimates the failure probabilities using a reliability analysis method, and the outer one explores the design space using an appropriate optimization algorithm. However, such a reliability analysis procedure must be performed repeatedly, which requires high computational expense in the case of complex systems. The mono-level and decoupled approaches achieve higher efficiency by, respectively, reformulating and decoupling the original RBDO problem. However, they both can suffer from multiple failure domains and strong nonlinearities in the limit-state functions.

1.2.3 Control Design Methods Considering Uncertainties

In recent years, uncertainties have been frequently taken into account in various control paradigms. Robust control [108] that aims at reducing the sensitivity to uncertainties is the most widely used concept for the control design of uncertain systems. In conventional robust methods, uncertainties are usually modeled as bounded sets and controllers are designed against the worst case (e.g. [109–111]). However, hard bounds can hardly be quantified exactly in practice. Strict bounds or relaxed bounds would bring about either over safe or unsafe outcomes. Meanwhile, the worst-case scenario may only occur with a vanishingly small probability, thus diminishing the design space and sacrificing potential performance as well as resulting in conservative controllers. In a broad sense, many researchers (e.g. [112, 113]) optimize a weighted sum of the mean value and variance of the quantity of interest to enhance the robustness. The work presented in [114] introduces a type of random sampling method called scenario approach, to satisfy the chance constraints irrespective of the uncertainty distribution. This method is robust to all possible distributions, but may lead to conservative results when the distribution is known. In addition, a large number of samples are required in this approach.

Chance constraints are often considered in the context of optimal control. Mesbah and Streif [115] applied the Cantelli-Chebyshev inequality to convert chance constraints into relaxed deterministic expressions with regard to mean and variance. Zhao and Kumar [116] attempted to estimated chance constraints using the split-Bernstein approach, which is a form of conservative approximation. Caillau et al. [117] developed an optimal control framework, where the distribution of interest is approximated directly by the kernel density estimation with a small sample set. Paulson and Mesbah [118] presented a moment-based approximation strategy for joint chance constraints and the results show that it is less conservative than the commonly used Cantelli-Chebyshev inequality.

Model predictive control (MPC) iteratively solves optimal control problems. Robust MPC and stochastic MPC are two variants of MPC that account for the influence of uncertainties. In [119], the min-max theory was introduced to MPC to achieve a worst-case control design, thus enhancing the robustness. The work in [120] proposed a tube-based MPC scheme which retains the disturbed response within an invariant tube. These robust MPC strategies rely on deterministic uncertainty modeling in a bounded set. Besides, chance constraints are also usually incorporated into the MPC formulation. The work presented in [121] transforms probabilistic constraints into convex second-order cone constraints which are only related to the mean value and variance. The Cantelli-Chebyshev inequality are employed in [122, 123] to obtain computationally tractable but conservative surrogates for the chance constraints. In addition, the before-mentioned scenario approach was also implemented in stochastic MPC [124].

1.3 Objectives

As presented in Section 1.1, safety requirements have been specified in several certification specifications in terms of admissible rare failure probabilities. In this context, to verify the compliance with these specifications, it is of critical importance to evaluate the reliability of flight control systems. Among the approximation methods and simulation methods for reliability analysis discussed in Section 1.2.1, SuS is tailored to estimate rare events, and thus, appears to be the best choice for our problem.

Despite the superiority of SuS over its counterparts, it requires a large number of performance evaluations to achieve sufficient estimation accuracy. However, assessing the performance of flight control systems is known to be a costly task. This is mainly because the quantity of interest is often time-dependent, and it is necessary to perform a numerical simulation on an expensive-to-evaluate model to obtain a single sample for SuS. For instance, in [40], the performance function is formulated as the root mean square error between the true and estimated values of a time-domain response, and simulations are conducted on a high-fidelity eVTOL aircraft model. Therefore, conducting a considerable number of performance function evaluations is computationally intractable in practical applications. To improve the efficiency of SuS for flight control systems, this thesis resorts to integrating surrogate modeling techniques to reduce the number of calls to the computationally demanding simulation model.

Based on the results of a successful estimation of the rare failure probability, designers may seek to improve the performance of the system. As a typical example, one may tune control parameters to satisfy the rare probabilistic requirements or explore further performance while guaranteeing the fulfillment of the probabilistic requirements. However, the control design strategies reviewed in Section 1.2.3 are not suitable to cope with this kind of requirements. They either handle uncertainties in a conservative way (e.g., design against the worst case and relax chance constraints), or focus on the events near the mean values rather than the extreme ones, which leads to a shrunken design space or large errors in the context of very small probabilistic requirements. To the best knowledge of the author, existing control design methods in the literature cannot handle rare events accurately, and essentially tend to transfer uncertainties into simple and computationally tractable expressions. Motivated by this limitation, this thesis aims to develop a control optimization framework based on a precise treatment of rare failure probabilities.

The main objectives of this thesis are listed as follows:

1. Develop a control design optimization framework which satisfies the rare probabilistic requirements directly.

2. Implement surrogate modeling techniques to accelerate the estimation of statistical information and failure probability.

3. Combine SuS with surrogate models and propose corresponding experimental design strategies to achieve efficient but yet accurate estimations for the rare failure probability.

4. Integrate dimensionality reduction methods to alleviate the computational complexity issue caused by the curse of dimensionality of surrogate models.

1.4 Contributions

Following the objectives described in the previous section, the contributions of this thesis are summarized in the remainder of this section.

1. **Development of a reliability-based control design framework:** Inspired by the RBDO scheme discussed in Section 1.2.2, this thesis presents a new control design method, *reliability-based control optimization* (RBCO), which satisfies the chance constraints directly. The proposed framework consists of two loops: the inner one estimates failure probabilities based on different types of surrogate-accelerated reliability analysis approaches, and the outer one explores the design space using optimization techniques. RBCO is a verification-driven method that guarantees the direct fulfillment of probabilistic requirements. Compared with conventional control design methods, the novel approach proposed here is shown to be less conservative in the sense that it allows to explore further performance which is sacrificed by the conventional ones.

   The presented RBCO framework has been published in Journal of Guidance, Control, and Dynamics [125].

2. **Uncertainty propagation using *polynomial chaos expansion* (PCE) and *response surface method* (RSM)**: PCE and RSM are employed to build global surrogate models for the performance function of control systems. Based on the constructed surrogates, the probability of violating an inequality constraint-based performance metric is estimated at a low computational cost. Furthermore, *subset simulation* (SuS) is combined with PCE and RSM to efficiently estimate rare failure probabilities. A global surrogate is first built using PCE at the initial level of SuS, after which RSM is applied to progressively refine the local surrogate at the following subset levels. The novel concept of adaptive PCE and RSM are introduced to obtain surrogates minimizing the cross-validation error. In order to balance the global and local prediction behavior, an adaptive experimental design strategy is proposed to choose the most valuable training samples from the full training set for the surrogate refinement. Compared with SuS, this innovative surrogate-accelerated SuS method tremendously reduces the required number of true model evaluations while providing similar estimation quality. Additionally, an indicator is given to measure the performance of the constructed surrogate model. This actively allows the user to monitor the quality of the results and therefore provides the user confidence when applying this method.

The results on the uncertainty propagation using PCE and RSM are based on publications [126, 127].

3. **Reliability analysis accelerated by *moving least-squares* (MLS)**: Although PCE and RSM are easy to implement, they may suffer severely from large estimation errors and the curse of dimensionality when encountering highly nonlinear and high-dimensional applications. To mitigate these limitations, MLS is exploited to construct a local low-order approximation for the performance function within the procedure of SuS. Aiming at reducing the classification error between subset levels, a new active learning strategy is proposed to add samples that are potentially misclassified by the intermediate failure boundary to the training set. By this means, in comparison with conventional SuS, the presented MLS-accelerated SuS gains much higher efficiency while still providing comparative estimation quality. To deal with high-dimensional problems, a novel dimensionality reduction strategy is further proposed for MLS-accelerated SuS. Here, sensitivity analysis is first conducted to rank the variables according to their importance with respect to the failure event. Afterwards, unnecessary expansion terms are filtered out based on the ranking, thus reducing the dimensionality of the feature space. With this dimensionality reduction strategy, the required training effort can be reduced dramatically without sacrificing the estimation accuracy.

   The contribution related to the MLS-accelerated SuS approach has been partially published in Journal of Guidance, Control, and Dynamics [125].

4. ***Kriging*-assisted reliability assessment:** In addition to the surrogate models based on rigid polynomial expressions, a kriging-based approach is developed to accelerate SuS. This thesis integrates the commonly implemented active learning kriging into SuS. However, the time-demanding hyperparameter optimization step limits kriging to low-dimensional applications. To address this issue, this thesis resorts to a partial least-squares-based kriging which reduces the number of hyperparameters and thereby the computational expense. Aiming at further speeding up the training procedure, an experimental design strategy is developed to choose influential training samples from the training set for the kriging model refinement. In addition, adaptive PCE is employed to detect the best polynomial trend for kriging. It is shown that the newly presented strategy requires the fewest number of calls to the true function compared to its counterparts proposed in this thesis.

5. **Comparisons of different surrogate models and applications to flight control systems:** This thesis compares the performance of the introduced surrogate-accelerated reliability analysis methods through several analytical illustrative examples. Given the comparison results, this thesis proposes a detailed recommendation

of methods to tackle different types of problems. These approaches are then applied to solve reliability assessment and RBCO problems for real-life flight control systems with an application-relevant level of complexity.

1.5 Outline of the Thesis

Before introducing the novel methods developed in this thesis, Chapter 2 recalls the basics of uncertainty propagation, including uncertainty representations, the failure probability estimation problem, and several simulation methods for solving this problem. Chapter 3 presents a reliability-based control design optimization framework which can be used to search for control parameters that satisfy probabilistic requirements with a formal guarantee. In Chapter 4, two polynomial-based global surrogates, namely, PCE and RSM, are employed to mimic the true model behaviors and achieve the uncertainty propagation task. A local surrogate model called MLS is integrated into simulation methods to accelerate the failure probability estimation procedure in Chapter 5. In Chapter 6, kriging surrogate modeling technique, which considers both the global trend and the local variability, is utilized to accelerate the simulation-based reliability analysis approaches. In addition, kriging and kriging-accelerated SuS are compared with the counterparts using PCE/RSM and MLS. In Chapter 7, the proposed methods are implemented to flight control systems for evaluating the failure probability and tune the control parameters. Finally, Chapter 8 concludes this thesis and provides an outlook to future work. In this thesis, all simulation results are obtained in *Matlab R2020a*.

Chapter 2

Fundamentals of Uncertainty Propagation

Uncertainty propagation is a class of problems that propagate uncertain inputs through a given computational model and then characterize the statistical features of system outputs, such as moments, failure probabilities, and distributions. This concept is illustrated in Figure 2.1.

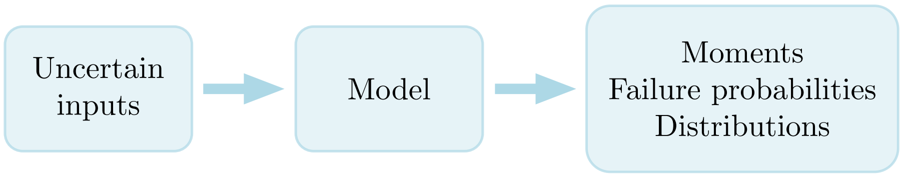


Figure 2.1: *Uncertainty propagation framework.*

This chapter presents the classification of uncertainties, the representation of uncertainties, the failure probability estimation problem, and simulation methods for solving this problem.

2.1 Uncertainties

2.1.1 Types of Uncertainties

Uncertainty arises from the limited knowledge of a state which makes it impossible to exactly describe the state. It can be generally classified into two categories [128, Ch. 2]:

- *Aleatory uncertainty*, also known as *stochastic uncertainty*, is the inherent randomness of phenomena. Every time we run the same experiment, the outcomes can differ from each other. For example, if an airplane takes off from the same runway for several times, the trajectories may not be identical due to different temperature and air pressure. If two same airplanes take off under the same condition, the outcomes may not be exactly the same because of the differences in manufacturing process. Herein, aleatory uncertainty lies in both the environment and the manufacturing process.

- *Epistemic uncertainty*, also known as *systematic uncertainty*, stems from a lack of knowledge about how a system should behave. For instance, when building an aircraft model, engineers may suffer from insufficient understandings or simplify complicated dynamics (e.g., the short period model). The resulting model uncertainties are typical epistemic uncertainties. This kind of uncertainty can be reduced by collecting more information, whereas aleatory uncertainty cannot.

2.1.2 Modeling of Uncertainties

Uncertainties have been represented using a variety of theories, such as probability theory, interval analysis, and evidence theory. This subsection presents the first two types of representations.

2.1.2.1 Probability Theory

Uncertainties are usually captured by *random variables*, which map possible outcomes to real numbers. Random variables can be continuous and discrete. Only continuous random variables are discussed in this thesis.

A continuous random variable Θ is described by its *probability density function* (PDF) $f_\Theta(\theta)$, where θ is a realization of Θ. It reflects the relative likelihood that the random variable equals the given outcome. For the sake of brevity, the subscript of $f_\Theta(\theta)$ will be omitted whenever it is clear from the context. The PDF satisfies the following properties:

$$\begin{aligned} f(\theta) &\geq 0, \\ \int_{-\infty}^{\infty} f(\theta)\mathrm{d}\theta &= 1. \end{aligned} \tag{2.1}$$

The *cumulative distribution function* (CDF) is defined by the probability that the random variable Θ is smaller than or equal to the realization θ:

$$F(\theta) = \mathbb{P}[\Theta \leq \theta] = \int_{-\infty}^{\theta} f(\vartheta)\mathrm{d}\vartheta. \tag{2.2}$$

The CDF is a non-decreasing function with limits:

$$\begin{aligned} \lim_{\theta\to-\infty} F(\theta) &= 0, \\ \lim_{\theta\to+\infty} F(\theta) &= 1. \end{aligned} \tag{2.3}$$

Statistical moments are a set of parameters that provide the distribution information of a random variable in a concise manner. The most typical types of moments are given as follows.

(1) The *mean*, also known as the *expected value* or *mathematical expectation*, is the first moment and defined as

$$\mu = \mathbb{E}_f[\Theta] = \int_{-\infty}^{\infty} \theta f(\theta) \mathrm{d}\theta, \tag{2.4}$$

where $\mathbb{E}_f$ is the mathematical expectation operator with regard to the PDF $f(\theta)$.

(2) The *variance*, which is the second central moment, measures how closely the outcomes spread around the mean:

$$\sigma^2 = \mathbb{E}_f[(\Theta - \mu)^2] = \int_{-\infty}^{\infty} (\theta - \mu)^2 f(\theta) \mathrm{d}\theta, \tag{2.5}$$

where σ represents the *standard deviation*. A small variance means that the random variable is tightly distributed, whereas a large variance indicates that the values spread widely around the mean.

(3) The standardized third central moment is called the *skewness*. It describes the asymmetry of a distribution and is defined as

$$\tau = \frac{1}{\sigma^3} \mathbb{E}_f[(\Theta - \mu)^3] = \frac{1}{\sigma^3} \int_{-\infty}^{\infty} (\theta - \mu)^3 f(\theta) \mathrm{d}\theta. \tag{2.6}$$

The skewness of a symmetric distribution is 0. A negative skewness indicates that the asymmetric distribution is left-skewed, i.e., the left tail is longer than the right one. If a distribution is skewed to the right, its skewness is positive.

(4) The standardized fourth central moment is known as the *kurtosis*. This parameter shows the heaviness of the tails of a distribution. It is given by

$$\kappa = \frac{1}{\sigma^4} \mathbb{E}_f[(\Theta - \mu)^4] = \frac{1}{\sigma^4} \int_{-\infty}^{\infty} (\theta - \mu)^4 f(\theta) \mathrm{d}\theta. \tag{2.7}$$

The kurtosis of a normal distribution is 3. If a distribution has light tails, the kurtosis is small. On the contrary, heavy-tailed distributions have large kurtosis.

Commonly used distributions are given in Appendix A.

A *random vector* $\boldsymbol{\Theta} = [\Theta_1, \Theta_2, \ldots, \Theta_n]^\mathsf{T}$ is a collection of random variables. It is also known as a *multivariate random variable*. The PDF and CDF of $\boldsymbol{\Theta}$ can be written as

$$\begin{aligned} f(\boldsymbol{\theta}) &= f(\theta_1, \theta_2, \ldots, \theta_n), \\ F(\boldsymbol{\theta}) &= F(\theta_1, \theta_2, \ldots, \theta_n) \\ &= \mathbb{P}[\Theta_1 \le \theta_1, \Theta_2 \le \theta_2, \ldots, \Theta_n \le \theta_n], \end{aligned} \tag{2.8}$$

where $\boldsymbol{\theta} = [\theta_1, \theta_2, \ldots, \theta_n]^\mathsf{T}$ is a realization of $\boldsymbol{\Theta}$.

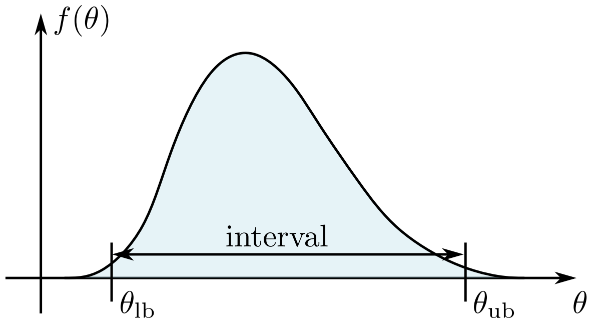


Figure 2.2: *Uncertainty representations.*

2.1.2.2 Interval Analysis

In many situations, uncertainties are modeled as bounded intervals. A uncertain parameter θ can take any value between the lower bound θ_{lb} and the upper bound θ_{ub} irrespective of the relative likelihood. Figure 2.2 shows examples of uncertainty representations using the bounded interval and the PDF. The bounded interval is simple and easy to explain. It can be an effective choice when the probability density is unavailable. However, it is nontrivial to choose the interval bounds. A tight interval may lead to unsafe approximations whereas a wide one can cause conservative results. Also, the uncertainty propagation within the framework of interval analysis addresses only the bounds on risks without any statement about how likely the risks are. These drawbacks can be avoided by modeling uncertainties using the PDF. Therefore, this thesis exploits the PDF to represent uncertainties and assumes that the joint PDF of uncertain parameters is available.

2.1.3 Isoprobabilistic Transformation

Uncertain inputs in surrogate modeling techniques and reliability analysis methods are often required to be mutually independent or/and standard random variables. For instance, PCE takes advantage of independent input components to construct orthonormal polynomials by tensor product. FORM and SORM estimate failure probabilities in the standard normal space. Therefore, it is necessary to convert any given random vector $\boldsymbol{\Theta}$ into an *independent standard random vector* $\boldsymbol{\Xi}$ (its realization is denoted by $\boldsymbol{\xi}$) through an *isoprobabilistic transformation*:

$$\boldsymbol{\Xi} = T(\boldsymbol{\Theta}). \tag{2.9}$$

2.1.3.1 Independent Variables

If the variables $\Theta_i, i = 1, \ldots, n$, are independent, the following transformation can be performed for each component:

$$\xi_i = \Phi^{-1}(F(\theta_i)), \tag{2.10}$$

where $\Phi(\cdot)$ is the CDF of the standard normal distribution. In the special case of independent normal variables, the isoprobabilistic transformation is simply the standardization:

$$\xi_i = \frac{\theta_i - \mu_{\Theta_i}}{\sigma_{\Theta_i}}, \tag{2.11}$$

where μ_{Θ_i} and σ_{Θ_i} are the mean and standard deviation of Θ_i.

2.1.3.2 Rosenblatt Transformation

Assume that the variables Θ_i are mutually dependent and the joint CDF is known, the Rosenblatt transformation can be applied. Based on the fact that

$$F(\theta_1, \theta_2, \ldots, \theta_n) = F(\theta_1)F(\theta_2|\theta_1)\cdots F(\theta_n|\theta_1, \theta_2, \ldots, \theta_{n-1}), \tag{2.12}$$

where $F(\theta_i|\theta_1, \ldots, \theta_{i-1})$ is the CDF of Θ_i conditioned by $\Theta_1 = \theta_1, \ldots, \Theta_{i-1} = \theta_{i-1}$, the *Rosenblatt transformation* [129] is defined as

$$\begin{aligned} \xi_1 &= \Phi^{-1}(F(\theta_1)), \\ \xi_2 &= \Phi^{-1}(F(\theta_2|\theta_1)), \\ &\vdots \\ \xi_n &= \Phi^{-1}(F(\theta_n|\theta_1, \theta_2, \ldots, \theta_{n-1})). \end{aligned} \tag{2.13}$$

The Rosenblatt transformation is accurate, but requires a full knowledge of the joint CDF which may not be available in practical applications.

2.1.3.3 Nataf Transformation

The *Nataf transformation* [130] exploits the marginal CDF $F(\theta_i)$ and the correlation matrix $\boldsymbol{R}_{\boldsymbol{\Theta}} = [\rho_{ij}]_{n\times n}$ to map a random vector from the physical space to the standard normal space. First, the random vector $\boldsymbol{\Theta}$ is transformed to $\boldsymbol{\Theta}'$ with zero mean, unit variance, and correlation matrix $\boldsymbol{R}_{\boldsymbol{\Theta}'} = [\rho'_{ij}]_{n\times n}$ through the component-wise transformation

$$\theta'_i = \Phi^{-1}(F(\theta_i)). \tag{2.14}$$

The relationship between ρ_{ij} and ρ'_{ij} is given by

$$\rho_{ij} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \left(\frac{\theta_i - \mu_{\Theta_i}}{\sigma_{\Theta_i}}\right)\left(\frac{\theta_j - \mu_{\Theta_j}}{\sigma_{\Theta_j}}\right) \phi_2(\theta'_i, \theta'_j, \rho'_{ij})\mathrm{d}\theta'_i\mathrm{d}\theta'_j, \tag{2.15}$$

where $\phi_2(\cdot)$ represents the PDF of the bivariate standard normal distribution. The computation of ρ'_{ij} via Equation (2.15) is rather complex, but empirical formulas for this are derived in [130]. The second step is to transfer $\boldsymbol{\Theta}'$ to a mutually independent standard normal random vector $\boldsymbol{\Xi}$ using the orthogonal transformation

$$\boldsymbol{\Xi} = \boldsymbol{L}^{-1}\boldsymbol{\Theta}', \tag{2.16}$$

where $\boldsymbol{L}$ is the lower triangular matrix from the Cholesky decomposition of $\boldsymbol{R}_{\boldsymbol{\Theta}'}$:

$$\boldsymbol{R}_{\boldsymbol{\Theta}'} = \boldsymbol{L}\boldsymbol{L}^{\mathsf{T}}. \tag{2.17}$$

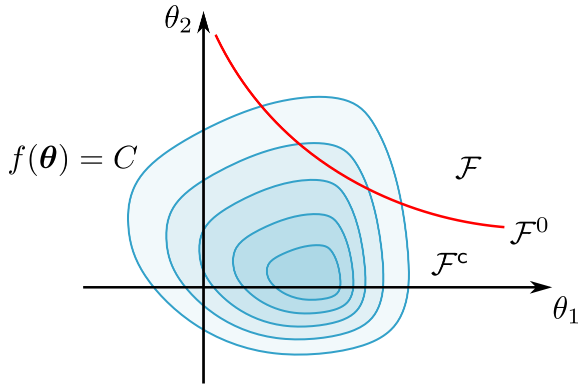


Figure 2.3: *Joint PDF $f(\boldsymbol{\theta})$, limit-state surface $\mathcal{F}^0$, failure domain $\mathcal{F}$, and safe domain $\mathcal{F}^{\mathrm{c}}$.*

2.2 Failure Probability Estimation

2.2.1 Definition of the Failure Domain

This thesis assumes that the quantity of interest of a system is described by a deterministic scalar function $g(\boldsymbol{\theta})$, which is called the *limit-state function* or *performance function*. The hypersurface $\mathcal{F}^0 = \{\boldsymbol{\theta} \in \boldsymbol{\Omega} : g(\boldsymbol{\theta}) = 0\}$ divides $\boldsymbol{\Omega} \subseteq \mathbb{R}^n$, where $\boldsymbol{\Omega}$ is the *support* of the random vector $\boldsymbol{\Theta}$, into a *failure domain* $\mathcal{F} = \{\boldsymbol{\theta} \in \boldsymbol{\Omega} : g(\boldsymbol{\theta}) \leq 0\}$ and a *safe domain* $\mathcal{F}^{\mathrm{c}} = \{\boldsymbol{\theta} \in \boldsymbol{\Omega} : g(\boldsymbol{\theta}) > 0\}$. This hypersurface is known as the *limit-state surface*. These concepts are visualized in Figure 2.3, where C is a constant and the contour represents the joint PDF $f(\boldsymbol{\theta})$. The *failure probability* can be expressed as an integral over the uncertain input space:

$$P_F = \mathbb{P}[g(\boldsymbol{\theta}) \leq 0] = \int_{\mathcal{F}} f(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta}. \tag{2.18}$$

By the isoprobabilistic transformation, the limit-state function becomes

$$g(\boldsymbol{\theta}) = g(T^{-1}(\boldsymbol{\xi})) = G(\boldsymbol{\xi}), \tag{2.19}$$

and Equation (2.18) boils down to

$$P_F = \mathbb{P}[G(\boldsymbol{\xi}) \leq 0]. \tag{2.20}$$

2.2.2 Evaluation of the Failure Probability

Approximation and simulation methods are two basic categories of strategies evaluating the failure probability. The most widely used approximation methods are the FORM [6, Ch. 7] and SORM [7]. Both approaches exploit the Taylor expansion of the limit-state surface at the MPFP $\boldsymbol{\xi}^*$ in the standard normal space to estimate the failure probability, as illustrated in Figure 2.4. Here, $\phi(\cdot)$ denotes the PDF of the standard normal distribution,

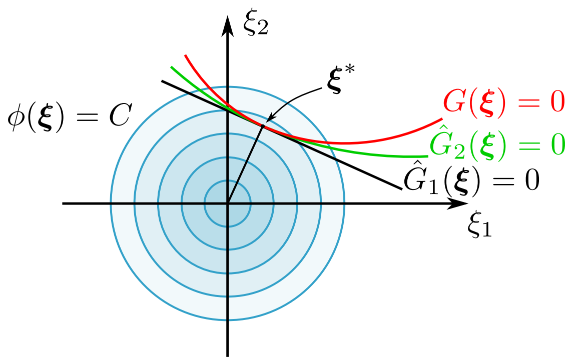

ξ_2
$\boldsymbol{\xi}^*$
$\phi(\boldsymbol{\xi}) = C$
$G(\boldsymbol{\xi}) = 0$
$\hat{G}_2(\boldsymbol{\xi}) = 0$
$\hat{G}_1(\boldsymbol{\xi}) = 0$
ξ_1

Figure 2.4: *FORM and SORM approximation.*

$\hat{G}_1(\boldsymbol{\xi}) = 0$ and $\hat{G}_2(\boldsymbol{\xi}) = 0$ denote the approximation of $G(\boldsymbol{\xi}) = 0$ using the FORM and SORM, respectively. Hence, the corresponding failure probability estimates are

$$\begin{aligned} \hat{P}_F^{\text{FORM}} &= \mathbb{P}[\hat{G}_1(\boldsymbol{\xi}) \leq 0] = \int_{\hat{G}_1(\boldsymbol{\xi})\leq 0} \phi(\boldsymbol{\xi})\mathrm{d}\boldsymbol{\xi}, \\ \hat{P}_F^{\text{SORM}} &= \mathbb{P}[\hat{G}_2(\boldsymbol{\xi}) \leq 0] = \int_{\hat{G}_2(\boldsymbol{\xi})\leq 0} \phi(\boldsymbol{\xi})\mathrm{d}\boldsymbol{\xi}. \end{aligned} \tag{2.21}$$

Simulation methods estimate the integral in Equation (2.18) by generating samples in the output space and identifying the percentage of samples falling in the failure domain. The most popular simulation methods are presented in the following sections.

2.3 Monte Carlo Simulation (MCS)

2.3.1 Standard Algorithm

Monte Carlo simulation (MCS) regards the integral in Equation (2.18) as an expectation:

$$P_F = \int_{\mathcal{F}} f(\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta} = \int_{\boldsymbol{\Omega}} I_{\mathcal{F}}(\boldsymbol{\theta}) f(\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta} = \mathbb{E}_f[I_{\mathcal{F}}(\boldsymbol{\Theta})], \tag{2.22}$$

where $I_{\mathcal{F}}(\boldsymbol{\theta})$ is the indicator function of the failure event $\mathcal{F}$:

$$I_{\mathcal{F}}(\boldsymbol{\theta}) = \begin{cases} 1, & \text{if } \boldsymbol{\theta} \in \mathcal{F}, \\ 0, & \text{if } \boldsymbol{\theta} \notin \mathcal{F}. \end{cases} \tag{2.23}$$

The MCS estimator can be expressed as the sample mean of the indicator function:

$$\hat{P}_F = \frac{1}{N} \sum_{i=1}^{N} I_{\mathcal{F}}(\boldsymbol{\theta}^{(i)}), \tag{2.24}$$

where $\{\boldsymbol{\theta}^{(i)} : i = 1, \ldots, N\}$ are N *independent and identically distributed* (i.i.d.) samples generated from $f(\boldsymbol{\theta})$. The conventional MCS method is summarized in Algorithm 1.

Algorithm 1 Conventional MCS algorithm

```
Input: The PDF of uncertain parameters $f(\boldsymbol{\theta})$; Limit-state function $g(\boldsymbol{\theta})$; The number of samples $N$.
Output: The estimate of failure probability $\hat{P}_F$.
1: Generate $N$ i.i.d. samples $\{\boldsymbol{\theta}^{(i)} : i = 1, \ldots, N\}$ according to $f(\boldsymbol{\theta})$;
2: Calculate the corresponding limit-state values $\{g(\boldsymbol{\theta}^{(i)}) : i = 1, \ldots, N\}$ and evaluate the indicator values $\{I_{\mathcal{F}}(\boldsymbol{\theta}^{(i)}) : i = 1, \ldots, N\}$;
3: Estimate the failure probability as in Equation (2.24);
4: return $\hat{P}_F$.
```

According to the strong *law of large numbers* (LLN) [131, Ch. 8], given a set of i.i.d. samples $\{\boldsymbol{\theta}^{(i)} : i = 1, \ldots, N\}$ drawn from $f(\boldsymbol{\theta})$ and any function $h(\cdot)$ with finite mean $\mathbb{E}_f[h(\boldsymbol{\Theta})]$, the sample average of $\{h(\boldsymbol{\theta}^{(i)}) : i = 1, \ldots, N\}$ converges to the mean $\mathbb{E}_f[h(\boldsymbol{\Theta})]$ almost surely as $N \to \infty$:

$$\mathbb{P}\left[\lim_{N\to\infty} \frac{1}{N}\sum_{i=1}^{N} h(\boldsymbol{\theta}^{(i)}) = \mathbb{E}_f[h(\boldsymbol{\Theta})]\right] = 1. \tag{2.25}$$

Let $h(\cdot) = I_{\mathcal{F}}(\cdot)$, one gets

$$\lim_{N\to\infty} \frac{1}{N}\sum_{i=1}^{N} I_{\mathcal{F}}(\boldsymbol{\theta}^{(i)}) = \mathbb{E}_f[I_{\mathcal{F}}(\boldsymbol{\Theta})] = P_F, \tag{2.26}$$

which means that the MCS estimate converges to the true failure probability.

2.3.2 Statistics and Accuracy of the Estimator

2.3.2.1 Mean and Variance

The mean of the MCS estimator is

$$\mathbb{E}\left[\hat{P}_F\right] = \mathbb{E}_f\left[\frac{1}{N}\sum_{i=1}^{N} I_{\mathcal{F}}(\boldsymbol{\theta}^{(i)})\right] = \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}_f\left[I_{\mathcal{F}}(\boldsymbol{\theta}^{(i)})\right] = \mathbb{E}_f\left[I_{\mathcal{F}}(\boldsymbol{\Theta})\right] = P_F, \tag{2.27}$$

demonstrating that the estimator is *unbiased*. Its variance is given by

$$\mathrm{Var}\left[\hat{P}_F\right] = \mathrm{Var}_f\left[\frac{1}{N}\sum_{i=1}^{N} I_{\mathcal{F}}(\boldsymbol{\theta}^{(i)})\right] = \frac{1}{N^2}\sum_{i=1}^{N}\mathrm{Var}_f\left[I_{\mathcal{F}}(\boldsymbol{\theta}^{(i)})\right] = \frac{1}{N}\mathrm{Var}_f\left[I_{\mathcal{F}}(\boldsymbol{\Theta})\right]. \tag{2.28}$$

In essence, $I_{\mathcal{F}}(\boldsymbol{\Theta})$ is a Bernoulli-distributed random variable [131, Ch. 4], i.e.,

$$\begin{aligned} \mathbb{P}\left[I_{\mathcal{F}}(\boldsymbol{\Theta}) = 1\right] &= P_F, \\ \mathbb{P}\left[I_{\mathcal{F}}(\boldsymbol{\Theta}) = 0\right] &= 1 - P_F, \end{aligned} \tag{2.29}$$

with mean and variance:

$$\begin{aligned} \mathbb{E}_f\left[I_{\mathcal{F}}(\boldsymbol{\Theta})\right] &= P_F, \\ \mathrm{Var}_f\left[I_{\mathcal{F}}(\boldsymbol{\Theta})\right] &= P_F(1 - P_F). \end{aligned} \tag{2.30}$$

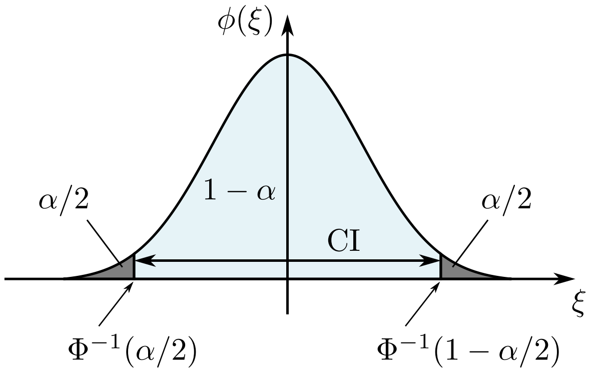


Figure 2.5: *Confidence interval for the standard normal distribution.*

Inserting Equation (2.30) into (2.28) yields

$$\mathrm{Var}\left[\hat{P}_F\right] = \frac{P_F(1-P_F)}{N} \approx \frac{\hat{P}_F(1-\hat{P}_F)}{N}. \tag{2.31}$$

2.3.2.2 Confidence Interval and Coefficient of Variation

Based on the *central limit theorem* (CLT) [131, Ch. 8], the average of $I_{\mathcal{F}}(\boldsymbol{\theta}^{(i)})$ tends to obey the following normal distribution as $N \to \infty$:

$$\frac{1}{N}\sum_{i=1}^{N} I_{\mathcal{F}}(\boldsymbol{\theta}^{(i)}) \sim \mathcal{N}\left(\mathbb{E}_f\left[I_{\mathcal{F}}(\boldsymbol{\Theta})\right], \frac{1}{N}\mathrm{Var}_f\left[I_{\mathcal{F}}(\boldsymbol{\Theta})\right]\right). \tag{2.32}$$

This indicates that, when N is large enough, the MCS estimate is normally distributed:

$$\hat{P}_F \sim \mathcal{N}\left(\mathbb{E}\left[\hat{P}_F\right], \mathrm{Var}\left[\hat{P}_F\right]\right). \tag{2.33}$$

A *confidence interval* (CI) of the estimate is a range of possible values $\left[\hat{P}_{F,\mathrm{lb}}, \hat{P}_{F,\mathrm{ub}}\right]$ with a *confidence level* $1-\alpha$, where $\alpha \in [0,1]$. The probability of an estimate falling in this interval is $1-\alpha$, whereas the probability of lying outside is α. For the standard normal distribution, the CI is usually given by

$$\left[\Phi^{-1}(\alpha/2), \Phi^{-1}(1-\alpha/2)\right], \tag{2.34}$$

as illustrated in Figure 2.5. The CI of the normally distributed estimate in Equation (2.33) is thus given by

$$\begin{aligned}\left[\hat{P}_{F,\mathrm{lb}}, \hat{P}_{F,\mathrm{ub}}\right] &= \left[\mathbb{E}\left[\hat{P}_F\right] + \Phi^{-1}(\alpha/2)\sqrt{\mathrm{Var}\left[\hat{P}_F\right]}, \mathbb{E}\left[\hat{P}_F\right] + \Phi^{-1}(1-\alpha/2)\sqrt{\mathrm{Var}\left[\hat{P}_F\right]}\right] \\ &\approx \left[\hat{P}_F + \Phi^{-1}(\alpha/2)\sqrt{\frac{\hat{P}_F(1-\hat{P}_F)}{N}}, \hat{P}_F + \Phi^{-1}(1-\alpha/2)\sqrt{\frac{\hat{P}_F(1-\hat{P}_F)}{N}}\right].\end{aligned} \tag{2.35}$$

The *coefficient of variation* (c.o.v.) is a commonly used parameter to evaluate the accuracy of the estimator. It is defined as the relative standard deviation with regard to the mean:

$$c_v = \frac{\sigma}{\mu}. \tag{2.36}$$

The c.o.v. of the MCS estimator is give by

$$c_v = \frac{\sqrt{\mathrm{Var}\left[\hat{P}_F\right]}}{\mathbb{E}\left[\hat{P}_F\right]} = \sqrt{\frac{1-P_F}{NP_F}} \approx \sqrt{\frac{1-\hat{P}_F}{N\hat{P}_F}}. \tag{2.37}$$

The relative confidence interval can be derived by dividing Equation (2.35) by the true probability and substituting Equation (2.37) into it, which results in:

$$\left[\hat{P}_{F,\mathrm{lb}}/P_F, \hat{P}_{F,\mathrm{ub}}/P_F\right] = \left[1+\Phi^{-1}(\alpha/2)c_v, 1+\Phi^{-1}(1-\alpha/2)c_v\right]. \tag{2.38}$$

This indicates that the relative confidence interval only depends on the c.o.v. given a confidence level.

2.3.3 Efficiency of the Estimator

To achieve a desired accuracy c_v, according to Equation (2.37), the required number of samples is derived as:

$$N_{\mathrm{req}} = \frac{1-P_F}{c_v^2 P_F}. \tag{2.39}$$

For rare events, it can be approximated by $N_{\mathrm{req}} \approx 1/(c_v^2 P_F)$. For instance, estimating a failure probability $P_F = 10^{-m}$ with a 10% c.o.v. requires about 10^{m+2} samples. As a consequence, MCS is inefficient at the assessment of rare failure probabilities. This results from the fact that the MCS technique generates samples from the original PDF $f(\boldsymbol{\theta})$ but most of these points do not lie in the failure domain. Also, achieving a prescribed accuracy level needs sufficient number of samples in the failure domain.

2.4 Importance Sampling (IS)

2.4.1 Standard Algorithm

Importance sampling (IS) [18] is a variance reduction approach that seeks to improve the efficiency of MCS by shifting the PDF for sampling towards the failure domain. The failure probability in Equation (2.22) can be rewritten as

$$\begin{aligned} P_F &= \int_{\boldsymbol{\Omega}} I_{\mathcal{F}}(\boldsymbol{\theta}) f(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta} = \int_{\boldsymbol{\Omega}} I_{\mathcal{F}}(\boldsymbol{\theta}) \frac{f(\boldsymbol{\theta})}{q(\boldsymbol{\theta})} q(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta} \\ &= \int_{\boldsymbol{\Omega}} I_{\mathcal{F}}(\boldsymbol{\theta}) w(\boldsymbol{\theta}) q(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta} = \mathbb{E}_q\left[I_{\mathcal{F}}(\boldsymbol{\Theta}) w(\boldsymbol{\Theta})\right], \end{aligned} \tag{2.40}$$

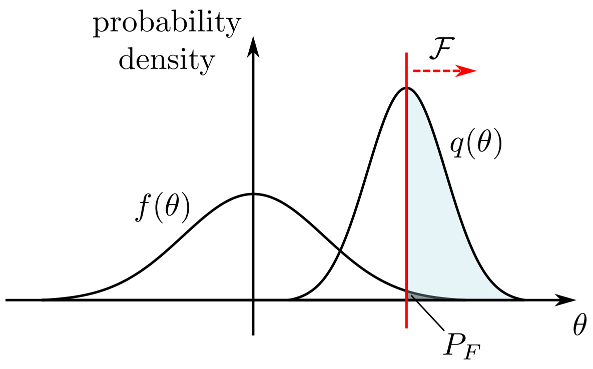


Figure 2.6: *General concept of IS.*

Algorithm 2 Basic IS algorithm [18]

```
Input: The PDF of uncertain parameters $f(\boldsymbol{\theta})$; The importance sampling density $q(\boldsymbol{\theta})$;
    Limit-state function $g(\boldsymbol{\theta})$; The number of samples $N$.
Output: The estimate of failure probability $\hat{P}_F$.
1: Generate $N$ i.i.d. samples $\{\boldsymbol{\theta}^{(i)} : i = 1, \dots, N\}$ according to $q(\boldsymbol{\theta})$;
2: Calculate the corresponding limit-state values $\{g(\boldsymbol{\theta}^{(i)}) : i = 1, \dots, N\}$, evaluate the
   indicator values $\{I_{\mathcal{F}}(\boldsymbol{\theta}^{(i)}) : i = 1, \dots, N\}$, and compute the importance weights
   $\{w(\boldsymbol{\theta}^{(i)}) : i = 1, \dots, N\}$;
3: Estimate the failure probability as in Equation (2.41);
4: return $\hat{P}_F$.
```

where $q(\boldsymbol{\theta})$ is the *importance sampling density* (ISD), and $w(\boldsymbol{\theta}) = f(\boldsymbol{\theta})/q(\boldsymbol{\theta})$ is the *importance weight function*. Instead of drawing samples from the original PDF $f(\boldsymbol{\theta})$, the IS method generates samples from the ISD $q(\boldsymbol{\theta})$ which is closer to the failure domain. This concept is illustrated in Figure 2.6. The samples drawn from $q(\boldsymbol{\theta})$ are more likely to lie in the failure domain. Therefore, fewer samples are required by IS to generate the same number of failure sampling points. By this means, compared with MCS, IS gains higher efficiency for failure probability evaluation.

The IS estimator is given by

$$\hat{P}_F = \frac{1}{N}\sum_{i=1}^{N} I_{\mathcal{F}}(\boldsymbol{\theta}^{(i)}) w(\boldsymbol{\theta}^{(i)}) = \frac{1}{N}\sum_{i=1}^{N} I_{\mathcal{F}}(\boldsymbol{\theta}^{(i)}) \frac{f(\boldsymbol{\theta}^{(i)})}{q(\boldsymbol{\theta}^{(i)})}, \tag{2.41}$$

where $\{\boldsymbol{\theta}^{(i)} : i = 1, \dots, N\}$ are N i.i.d. samples generated from $q(\boldsymbol{\theta})$. According to the strong *law of large numbers* (LLN) (see Equation (2.25)), the IS estimator converges almost surely to the true failure probability:

$$\lim_{N\to\infty} \frac{1}{N}\sum_{i=1}^{N} I_{\mathcal{F}}(\boldsymbol{\theta}^{(i)}) w(\boldsymbol{\theta}^{(i)}) = \mathbb{E}_q\left[I_{\mathcal{F}}(\boldsymbol{\Theta}) w(\boldsymbol{\Theta})\right] = P_F. \tag{2.42}$$

The basic IS algorithm is given in Algorithm 2, where the ISD $q(\boldsymbol{\theta})$ is prescribed. The selection of the ISD will be discussed in Section 2.4.3.

2.4.2 Statistics of the Estimator

The mean of the IS estimator is

$$\begin{aligned}\mathbb{E}\left[\hat{P}_F\right] &= \mathbb{E}_q\left[\frac{1}{N}\sum_{i=1}^{N} I_{\mathcal{F}}(\boldsymbol{\theta}^{(i)})w(\boldsymbol{\theta}^{(i)})\right] = \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}_q\left[I_{\mathcal{F}}(\boldsymbol{\theta}^{(i)})w(\boldsymbol{\theta}^{(i)})\right] \\ &= \mathbb{E}_q\left[I_{\mathcal{F}}(\boldsymbol{\Theta})w(\boldsymbol{\Theta})\right] = P_F,\end{aligned} \tag{2.43}$$

which proves that the IS estimator is *unbiased.* The variance of the estimation is

$$\begin{aligned}\mathrm{Var}\left[\hat{P}_F\right] &= \mathrm{Var}_q\left[\frac{1}{N}\sum_{i=1}^{N} I_{\mathcal{F}}(\boldsymbol{\theta}^{(i)})w(\boldsymbol{\theta}^{(i)})\right] = \frac{1}{N^2}\sum_{i=1}^{N}\mathrm{Var}_q\left[I_{\mathcal{F}}(\boldsymbol{\theta}^{(i)})w(\boldsymbol{\theta}^{(i)})\right] \\ &= \frac{1}{N}\mathrm{Var}_q\left[I_{\mathcal{F}}(\boldsymbol{\Theta})w(\boldsymbol{\Theta})\right] = \frac{1}{N}\mathrm{Var}_q\left[I_{\mathcal{F}}(\boldsymbol{\Theta})\frac{f(\boldsymbol{\Theta})}{q(\boldsymbol{\Theta})}\right].\end{aligned} \tag{2.44}$$

Due to the following property of variance for any function $h(\cdot)$:

$$\mathrm{Var}\left[h(\boldsymbol{\Theta})\right] = \mathbb{E}\left[h(\boldsymbol{\Theta})^2\right] - \mathbb{E}\left[h(\boldsymbol{\Theta})\right]^2, \tag{2.45}$$

Equation (2.44) becomes

$$\mathrm{Var}\left[\hat{P}_F\right] = \frac{1}{N}\left(\mathbb{E}_q\left[\left(I_{\mathcal{F}}(\boldsymbol{\Theta})\frac{f(\boldsymbol{\Theta})}{q(\boldsymbol{\Theta})}\right)^2\right] - P_F^2\right). \tag{2.46}$$

According to Equations (2.36), (2.43), and (2.46), the required number of samples given a prescribed accuracy c_v is computed by

$$N_{\text{req}} = \frac{\mathbb{E}_q\left[\left(I_{\mathcal{F}}(\boldsymbol{\Theta})\frac{f(\boldsymbol{\Theta})}{q(\boldsymbol{\Theta})}\right)^2\right] - P_F^2}{c_v^2 P_F^2}. \tag{2.47}$$

These results show that the accuracy and efficiency of the IS estimator highly depends on the selection of the ISD $q(\boldsymbol{\theta})$. If $q(\boldsymbol{\theta}) = f(\boldsymbol{\theta})$, Equation (2.46) reduces to Equation (2.31) and the IS estimator is equal to the MCS estimator. The key of IS is to choose a suitable ISD that results in a lower variance than MCS. This is the reason why the IS approach is a type of variance reduction technique.

2.4.3 Selection of the Importance Sampling Density

By minimizing the variance of IS, the optimal ISD is obtained:

$$q^*(\boldsymbol{\theta}) = f(\boldsymbol{\theta}|\mathcal{F}) = \frac{I_{\mathcal{F}}(\boldsymbol{\theta})f(\boldsymbol{\theta})}{P_F}. \tag{2.48}$$

The optimal ISD is shown in Figure 2.7. It is straightforward to show that this ISD leads to a zero estimation variance. A single sample generated from $q^*(\boldsymbol{\theta})$ is sufficient to accurately assess the failure probability. However, it is impossible to use the optimal ISD, because the failure probability P_F and the indicator function $I_{\mathcal{F}}(\boldsymbol{\theta})$ are unknown before sampling.

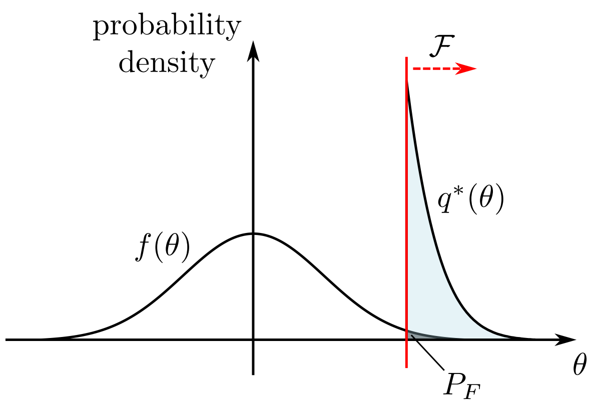


Figure 2.7: *Optimal importance sampling density.*

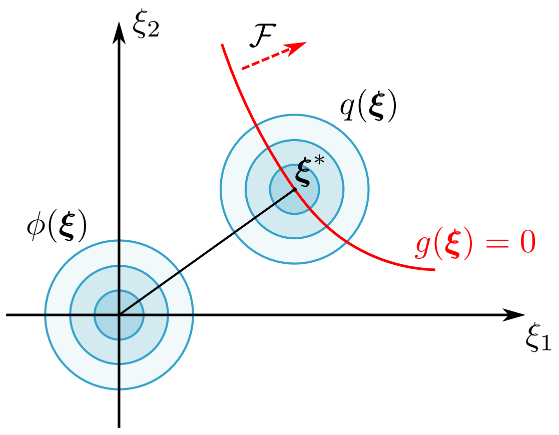


Figure 2.8: *MPFP-based importance sampling density.*

Various techniques have been developed to construct the ISD. A popular strategy is shifting the mean of the original parameter distribution to the MPFP [18]. In the standard normal space, the original parameter distribution is $\phi(\boldsymbol{\xi})$ and the ISD is given by

$$q(\boldsymbol{\xi}) = \phi(\boldsymbol{\xi} - \boldsymbol{\xi}^*), \tag{2.49}$$

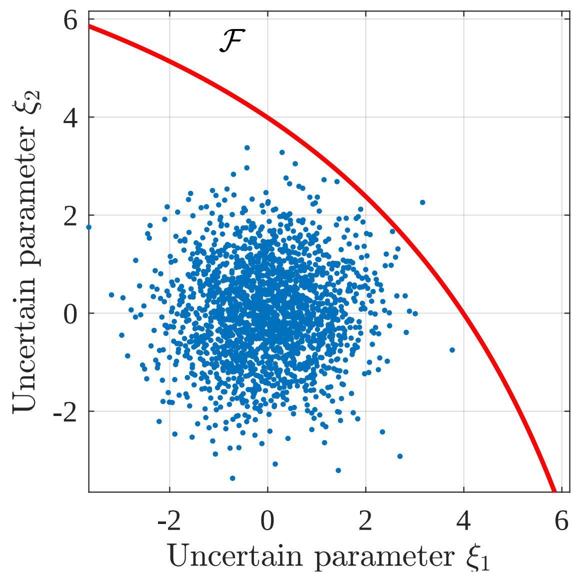
as illustrated in Figure 2.8. Drawing samples from $q(\boldsymbol{\xi})$ which is closer to the failure domain speeds up the convergence of the estimation. However, this method requires the knowledge of the MPFP. Finding the MPFP is an optimization process, which is a nontrivial task especially for high-dimensional problems.

2.4.4 Illustrative Example

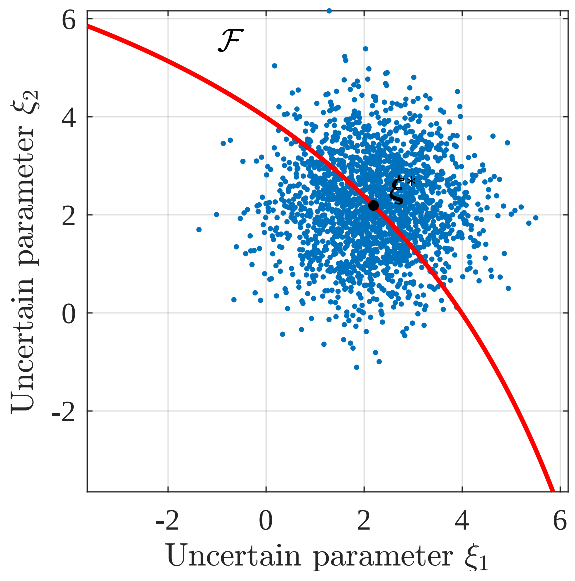
Consider the following 2-dimensional reliability problem with a smooth limit-state function given by

$$g(\boldsymbol{\xi}) = a - \left(e^{0.1\xi_1} + e^{0.1\xi_2}\right), \tag{2.50}$$

where a is a constant and $\xi_i, i = 1, 2$, are independent standard normal random variables, i.e., $\xi_i \sim \mathcal{N}(0, 1)$. The failure domain is defined as $\mathcal{F} = \{\boldsymbol{\xi} \in \mathbb{R}^2 : g(\boldsymbol{\xi}) \leq 0\}$. Figure 2.9 shows the sampling strategies of MCS and IS. Here, $a = 2.49$ and $N = 2000$ samples are



a) *MCS samples*



b) *IS samples*

Figure 2.9: *Samples for MCS and IS.*

generated by both approaches. As shown in Figure 2.9a, MCS draws samples from the original PDF $\phi(\boldsymbol{\xi})$ and only a few samples fall in the failure domain. In Figure 2.9b, the MPFP $\boldsymbol{\xi}^*$ is first calculated by solving the following constrained optimization problem:

$$\boldsymbol{\xi}^* = \arg\min \frac{1}{2}\boldsymbol{\xi}^\mathsf{T}\boldsymbol{\xi} \quad \text{s.t.} \;\; g(\boldsymbol{\xi}) = 0. \tag{2.51}$$

After that, IS generates samples from the ISD in Equation (2.49). A large proportion of the generated samples lie in the failure domain. By shifting the PDF for sampling towards the failure domain, the estimation accuracy is improved significantly.

The efficiency of MCS and IS is compared in Figure 2.10. In this graph, the required number of samples for different accuracy levels $c_v = [0.05, 0.1, 0.2]^\mathsf{T}$ and probabilities of different orders of magnitude are computed according to Equations (2.39) and (2.47). These probabilities of different orders of magnitude and the corresponding a are listed in Table 2.1. As illustrated in this figure, for both methods, more samples are necessary to achieve a higher accuracy level or to estimate a smaller failure probability. It should be emphasized that IS is considerably more efficient than MCS, especially for rare events. The high efficiency of IS, in this example, is based on the accessibility of the MPFP. For systems with multiple failure domains or complex nonlinearity where the MPFP is difficult to find, the accuracy and efficiency of IS may be degraded.

Table 2.1: *Parameters and the corresponding failure probabilities.*

a	2.81	2.71	2.61	2.49	2.36	2.20
P_F	1.01×10^{-6}	1.05×10^{-5}	9.81×10^{-5}	1.11×10^{-3}	1.07×10^{-2}	9.49×10^{-2}

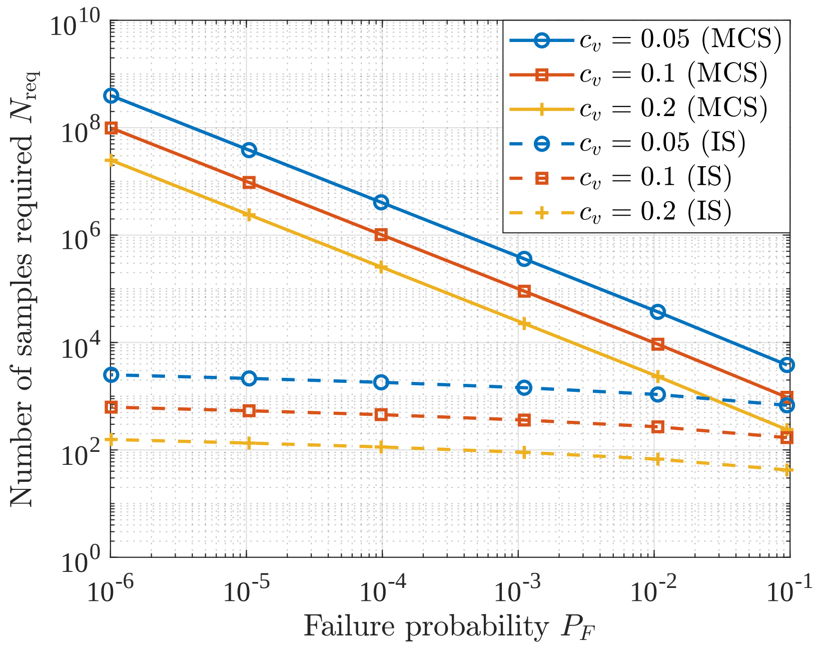


Figure 2.10: *Number of samples required by MCS and IS for given c.o.v.*

2.5 Subset Simulation (SuS)

2.5.1 Standard Algorithm

The *subset simulation* (SuS) method [26, 27] is an adaptive *Markov chain Monte Carlo* (MCMC) procedure to efficiently estimate small failure probabilities. The basic idea behind SuS is to introduce a series of nested *intermediate failure domains* or *subsets* $\mathcal{F}_j$, $j = 1, 2, \ldots, m$, with $\mathcal{F}_1 \supset \mathcal{F}_2 \supset \cdots \supset \mathcal{F}_m = \mathcal{F}$, and then transcribe the small failure probability into a product of larger conditional probabilities:

$$\begin{aligned} P_F = \mathbb{P}[\mathcal{F}_m] &= \mathbb{P}[\mathcal{F}_m|\mathcal{F}_{m-1}]\,\mathbb{P}[\mathcal{F}_{m-1}] = \cdots \\ &= \mathbb{P}[\mathcal{F}_m|\mathcal{F}_{m-1}]\cdots\mathbb{P}[\mathcal{F}_2|\mathcal{F}_1]\,\mathbb{P}[\mathcal{F}_1] = \prod_{j=1}^{m}\mathbb{P}[\mathcal{F}_j|\mathcal{F}_{j-1}], \end{aligned} \tag{2.52}$$

where $\mathcal{F}_0 = \boldsymbol{\Omega}$ is the support of the random variable $\boldsymbol{\Theta}$ and $\mathbb{P}[\mathcal{F}_1|\mathcal{F}_0] = \mathbb{P}[\mathcal{F}_1]$. The intermediate failure domains $\mathcal{F}_j$, $j = 1, 2, \ldots, m$, are defined as $\mathcal{F}_j = \{\boldsymbol{\theta} \in \boldsymbol{\Omega} : g(\boldsymbol{\theta}) \leq b_j\}$, where b_j, $j = 1, 2, \ldots, m$, are *intermediate thresholds* and $b_1 > b_2 > \cdots > b_m = 0$. In practice, b_j, $j = 1, \ldots, m-1$, are chosen to be the p_0-percentile of the function responses at each subset level (j-th level), such that the estimates of $\mathbb{P}[\mathcal{F}_j|\mathcal{F}_{j-1}]$, $j = 1, \ldots, m-1$, correspond to the preset *conditional probability* p_0, whereas that of $\mathbb{P}[\mathcal{F}_m|\mathcal{F}_{m-1}]$ is greater than or equal to p_0. The conditional probability p_0 should be large enough to enable the efficient estimation of $\mathbb{P}[\mathcal{F}_j|\mathcal{F}_{j-1}]$, $j = 1, \ldots, m$, by simulation. It is demonstrated in [132] that $p_0 \in [0.1, 0.3]$ leads to similar efficiency as the optimal choice of p_0. In this thesis, p_0 is chosen to be 0.1.

Algorithm 3 Basic SuS algorithm [26, 27]

```
Input: The PDF of uncertain parameters $f(\boldsymbol{\theta})$; Limit-state function $g(\boldsymbol{\theta})$; Conditional
    probability $p_0$; The number of samples per subset level $N$.
Output: The estimate of failure probability $\hat{P}_F$.
 1: Draw $N$ i.i.d. samples $\{\boldsymbol{\theta}_0^{(i)} : i = 1, \ldots, N\}$ in accordance with $f(\boldsymbol{\theta})$;
 2: Calculate the corresponding limit-state values $\{g(\boldsymbol{\theta}_0^{(i)}) : i = 1, \ldots, N\}$;
 3: Find $b_1$ as the $p_0$-percentile of the responses $\{g(\boldsymbol{\theta}_0^{(i)}) : i = 1, \ldots, N\}$ and set $\mathcal{F}_1 =$
    $\{\boldsymbol{\theta} \in \boldsymbol{\Omega} : g(\boldsymbol{\theta}) \leq b_1\}$;
 4: Set $j = 1$;
 5: while $b_j > 0$ do
 6:     Consider samples $\boldsymbol{\theta}_{j-1}^{(i)} \in \mathcal{F}_j$ as seeds $\{\boldsymbol{\theta}_{j-1,s}^{(i)} : i = 1, \ldots, N_s\}$, where $N_s = p_0 N$ is
        an integer;
 7:     Employ an MCMC sampling approach to generate $N$ samples $\{\boldsymbol{\theta}_j^{(i)} : i = 1, \ldots, N\}$
        from the seeds;
 8:     Obtain the corresponding limit-state values $\{g(\boldsymbol{\theta}_j^{(i)}) : i = 1, \ldots, N\}$;
 9:     Find $b_{j+1}$ as the $p_0$-percentile of $\{g(\boldsymbol{\theta}_j^{(i)}) : i = 1, \ldots, N\}$ and set $\mathcal{F}_{j+1} = \{\boldsymbol{\theta} \in \boldsymbol{\Omega} :$
        $g(\boldsymbol{\theta}) \leq b_{j+1}\}$;
10:     Set $j = j + 1$;
11: end while
12: Set the total number of subsets $m = j$, set $b_m = 0$, denote the number of samples
    $\boldsymbol{\theta}_{m-1}^{(i)} \in \mathcal{F}$ by $N_F$, and estimate the failure probability as in Eq. (2.54);
13: return $\hat{P}_F$.
```

The probability $\mathbb{P}[\mathcal{F}_1|\mathcal{F}_0]$ is approximated by MCS through drawing i.i.d. samples from $f(\boldsymbol{\theta})$. For estimating $\mathbb{P}[\mathcal{F}_j|\mathcal{F}_{j-1}]$, $j = 2, \ldots, m$, MCMC is implemented to generate samples from the conditional joint PDF $f(\boldsymbol{\theta}|\mathcal{F}_{j-1})$ using samples that fall in $\mathcal{F}_{j-1}$ as *seeds*, where

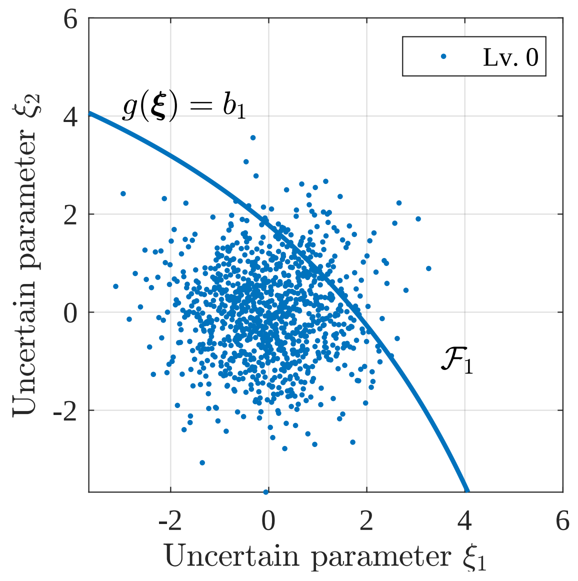
$$f(\boldsymbol{\theta}|\mathcal{F}_{j-1}) = \frac{f(\boldsymbol{\theta}) I_{\mathcal{F}_{j-1}}(\boldsymbol{\theta})}{\mathbb{P}[\mathcal{F}_{j-1}]}, \quad j = 2, \ldots, m. \tag{2.53}$$

MCMC algorithms will be discussed in detail in Section 2.5.2. In the end, the estimate of failure probability is given by

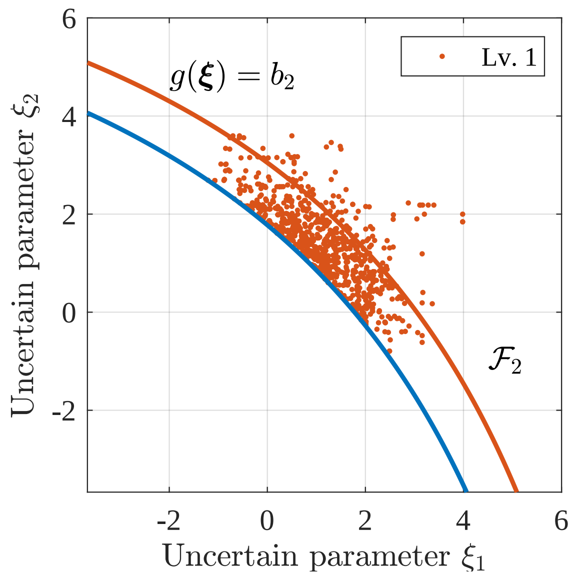
$$\hat{P}_F = p_0^{m-1} \frac{N_F}{N}, \tag{2.54}$$

where N is the number of samples at each subset level ($j = 0, 1, \ldots, m-1$) and N_F is the number of samples lying in the failure domain at the last level of SuS. The general SuS method is summarized in Algorithm 3.

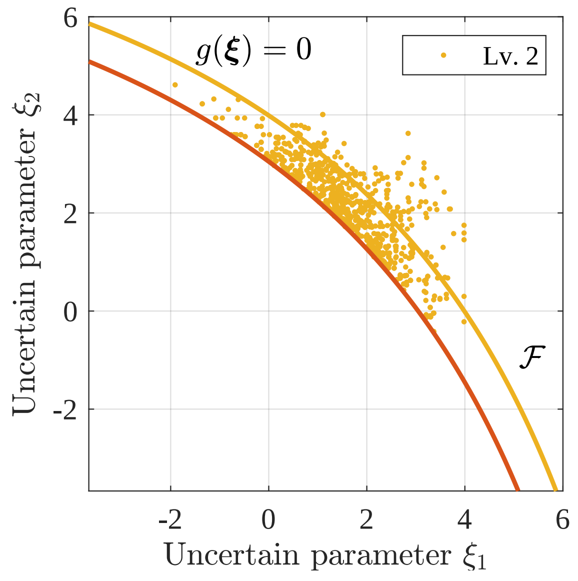
An example illustrating the process of SuS is given in Figure 2.11. Herein, the reliability problem in Section 2.4.4 with $a = 2.49$ is solved by SuS with $p_0 = 0.1$ and $N = 1000$. The uncertain parameters follow the independent standard normal distribution, which means $\boldsymbol{\theta} = \boldsymbol{\xi}$ in this example. At the initial level, MCS samples $\boldsymbol{\xi}_0^{(i)}$ are drawn from



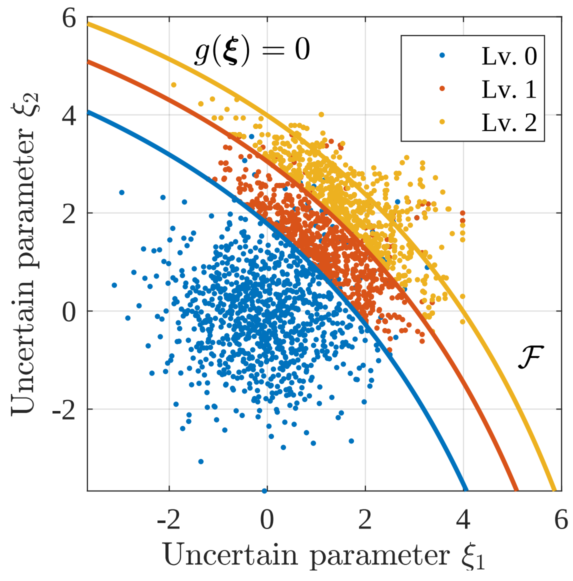
a) *Level 0*



b) *Level 1*



c) *Level 2*



d) *All levels*

Figure 2.11: *SuS process.*

$\phi(\boldsymbol{\xi})$, as shown in Figure 2.11a. The *intermediate boundary* $\{\boldsymbol{\xi} \in \boldsymbol{\Omega} : g(\boldsymbol{\xi}) = b_1\}$ and the intermediate failure domain $\mathcal{F}_1$ (the region above the intermediate boundary) are determined. Meanwhile, 10% of these samples lying in $\mathcal{F}_1$ are chosen as the seeds $\boldsymbol{\xi}_{0,s}^{(i)}$. Next, MCMC sampling is employed to generate samples $\boldsymbol{\xi}_1^{(i)}$ that locate in $\mathcal{F}_1$, as illustrated in Figure 2.11b. Then, the intermediate boundary $\{\boldsymbol{\xi} \in \boldsymbol{\Omega} : g(\boldsymbol{\xi}) = b_2\}$ and the intermediate failure domain $\mathcal{F}_2$ are decided while the seeds $\boldsymbol{\xi}_{1,s}^{(i)}$ are selected. The same steps (MCMC sampling and finding seeds) are repeated until enough samples ($\geq p_0 N$) generated in a certain level fall in the failure domain $\mathcal{F}$. In this example, $b_3 = 0$ and $\mathcal{F}_3 = \mathcal{F}$, as depicted in Figure 2.11c. Figure 2.11d shows all the subset levels. SuS detects the failure domain gradually by choosing seeds and generating samples in the intermediate failure domain.

2.5.2 Markov Chain Monte Carlo

Markov chain Monte Carlo (MCMC) sampling [28, Ch. 6] enables the efficient generation of samples in accordance with an arbitrary target distribution. The samples are generated as the states of Markov chains whose stationary distribution is the target distribution. This subsection introduces the concept of Markov chain, the principle of MCMC, and several popular MCMC algorithms.

2.5.2.1 Markov Chain

A *Markov chain* is a stochastic model that represents a sequence of random variables $\Theta^{(i)}, i = 1, 2, \dots,$ with the following Markov property:

$$\begin{aligned} &\mathbb{P}\left[\Theta^{(i+1)} = \theta^{(i+1)} \middle| \Theta^{(1)} = \theta^{(1)}, \Theta^{(2)} = \theta^{(2)}, \dots, \Theta^{(i)} = \theta^{(i)}\right] \\ =&\mathbb{P}\left[\Theta^{(i+1)} = \theta^{(i+1)} \middle| \Theta^{(i)} = \theta^{(i)}\right] \end{aligned} \tag{2.55}$$

for all i and $\theta^{(i)}$. This property means that the future state depends only on the current state and not on the previous states. If the transition probability does not change over time, i.e.,

$$\mathbb{P}\left[\Theta^{(i+1)} = \theta_{\text{new}} \middle| \Theta^{(i)} = \theta_{\text{old}}\right] = \mathbb{P}\left[\Theta^{(2)} = \theta_{\text{new}} \middle| \Theta^{(1)} = \theta_{\text{old}}\right] \tag{2.56}$$

for all i, θ_{old}, and θ_{new}, the Markov chain is *time-homogeneous*. In the following context, only the time-homogeneous Markov chain is discussed.

For a continuous state space, the conditional density of $\Theta^{(i+1)} = \theta_{\text{new}}$ given $\Theta^{(i)} = \theta_{\text{old}}$ is described by the *transition density* $f(\theta_{\text{new}}|\theta_{\text{old}})$. For a discrete state space, transition probabilities are collected by the *transition matrix* $\boldsymbol{T} \in [0,1]^{n\times n}$, where n is the size of the state space. The elements T_{kl} describes the transition probability from the l-th state to the k-th state:

$$T_{kl} = \mathbb{P}\left[\Theta^{(i+1)} = \theta_k \middle| \Theta^{(i)} = \theta_l\right] \tag{2.57}$$

An example is given in Figure 2.12 with states $S = \{\theta_1, \theta_2, \theta_3\}$ and transition matrix

$$\boldsymbol{T} = \begin{bmatrix} 0.5 & 0.2 & 0.2 \\ 0.1 & 0.6 & 0.3 \\ 0.4 & 0.2 & 0.5 \end{bmatrix}. \tag{2.58}$$

A Markov chain is *irreducible* or *ergodic* if any state in the state space is reachable from any other states by finite transitions with positive probability. A state is called *periodic* if it is visited at regular intervals, otherwise it is *aperiodic*. If the states of a Markov chain are aperiodic, this Markov chain is called aperiodic. An irreducible and aperiodic Markov chain has a *stationary distribution* $\pi(\theta)$ irrespective of its initial condition $\pi^{(1)}(\theta)$:

$$\pi^{(i)}(\theta) \to \pi(\theta) \text{ as } i \to \infty, \tag{2.59}$$

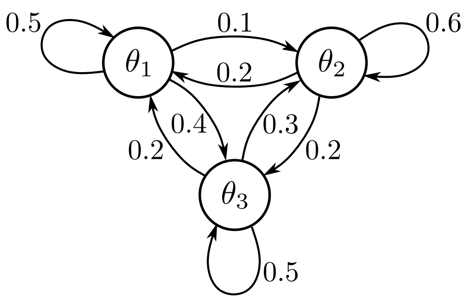


Figure 2.12: *Example of a discrete-state Markov chain with three states.*

where $\pi^{(i)}(\theta)$ is the distribution of $\Theta^{(i)}$. For discrete-state Markov chains, Equation (2.59) can be rewritten as

$$\boldsymbol{\pi}^{(i)} \rightarrow \boldsymbol{\pi} \text{ as } i \rightarrow \infty, \tag{2.60}$$

where $\boldsymbol{\pi}^{(i)} = [\pi_1^{(i)}, \ldots, \pi_n^{(i)}]^\mathsf{T}$ represents the probability mass of $\Theta^{(i)}$, each element describes the probability of the corresponding state: $\pi_k^{(i)} = \mathbb{P}[\Theta^{(i)} = \theta_k]$, $k = 1, \ldots, n$, and

$$\boldsymbol{\pi}^{(i+1)} = \boldsymbol{T}\boldsymbol{\pi}^{(i)}. \tag{2.61}$$

The *stationary probability mass* is denoted by $\boldsymbol{\pi}$. In the example depicted in Figure 2.12, the probability mass vector converges to $\boldsymbol{\pi} = [0.2857, 0.3469, 0.3673]^\mathsf{T}$ given any initial probability mass $\boldsymbol{\pi}^{(1)}$. The transient period until the chain reaches its stationary distribution is called *burn-in* phase. After the burn-in phase $i = 1, \ldots, r$, the distribution of $\Theta^{(i)}$ equals the stationary distribution

$$\pi^{(i)}(\theta) = \pi(\theta) \text{ or } \boldsymbol{\pi}^{(i)} = \boldsymbol{\pi} \tag{2.62}$$

for any $i \geq r$. This property holds for a single chain as well, i.e., the distribution of $\{\theta^{(i+1)}, \ldots, \theta^{(i+k)}\}$ is equal to $\pi(\theta)$ or $\boldsymbol{\pi}$ for any k and $i \geq r$. If the initial distribution of a Markov chain is equivalent to the stationary distribution, then the Markov chain is *stationary*, which means the previous property holds for any k and i.

A Markov chain satisfies the *detailed balance condition* if there exists a function $\pi(\theta)$ satisfying

$$\pi(\theta_{\text{old}}) f(\theta_{\text{new}}|\theta_{\text{old}}) = \pi(\theta_{\text{new}}) f(\theta_{\text{old}}|\theta_{\text{new}}) \tag{2.63}$$

for every θ_{old} and θ_{new}. If this condition is fulfilled, $\pi(\theta)$ is the stationary distribution of the Markov chain. This is proved as follows:

$$\begin{aligned} \int \pi(\theta_{\text{old}}) f(\theta_{\text{new}}|\theta_{\text{old}}) \mathrm{d}\theta_{\text{old}} &= \int \pi(\theta_{\text{new}}) f(\theta_{\text{old}}|\theta_{\text{new}}) \mathrm{d}\theta_{\text{old}} \\ &= \pi(\theta_{\text{new}}) \int f(\theta_{\text{old}}|\theta_{\text{new}}) \mathrm{d}\theta_{\text{old}} = \pi(\theta_{\text{new}}), \end{aligned} \tag{2.64}$$

which means that the distribution of the next state is equal to the current distribution $\pi(\theta)$. The detailed balance condition indicates that the probability transferred from θ_{old} to θ_{new} is the same as the probability transferred from θ_{new} back to θ_{old}. This condition is

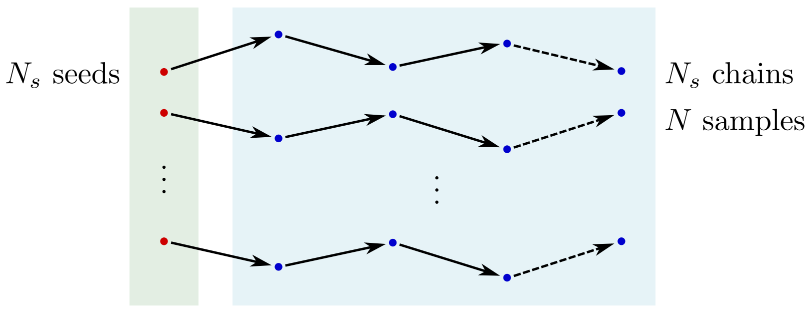


Figure 2.13: *MCMC sampling in SuS.*

sufficient but not necessary for $\pi(\theta)$ to be a stationary distribution. The detailed balance condition for discrete-state Markov chains requires that there exists a probability mass vector $\boldsymbol{\pi} = [\pi_1, \ldots, \pi_n]^\mathsf{T}$ satisfying

$$\pi_l T_{kl} = \pi_k T_{lk} \tag{2.65}$$

for any l and k. In the example shown in Figure 2.12, the stationary probability mass $\boldsymbol{\pi} = [0.2857, 0.3469, 0.3673]^\mathsf{T}$ does not satisfy Equation (2.65).

2.5.2.2 MCMC Sampling

MCMC sampling consists in generating states of a Markov chain whose stationary distribution is the target distribution. This is achieved by selecting a transition density that results in an irreducible and aperiodic Markov chain and satisfies the detailed balance condition with regard to the stationary distribution of the Markov chain. Note that the samples generated by MCMC sampling are not independent, which is different with using conventional MCS. Drawing independent samples is generally more efficient than generating dependent samples. However, it is challenging or impossible to directly draw independent samples from a complex distribution. In the following, how to apply MCMC sampling in SuS is first introduced and then several popular MCMC algorithms are presented.

MCMC Sampling in SuS

At each level of SuS, $N_s = p_0 N$ samples falling in the intermediate failure domain $\mathcal{F}_j$ are selected as seeds for MCMC sampling. The number of chains is equal to the number of seeds and N new samples are generated, as illustrated in Figure 2.13. As a result, each seed generates a Markov chain with $1/p_0$ samples. An alternative choice is to draw $N - N_s$ new samples and there are N samples in total including the seeds in the intermediate failure domain. Furthermore, the seeds at each level of SuS are already distributed as the target stationary distribution $f(\boldsymbol{\theta}|\mathcal{F}_j)$. As a consequence, the derived samples comply with the target distribution from the beginning of the MCMC sampling procedure. The following MCMC algorithms which generate samples from a single seed can be easily adapted to draw samples from multiple seeds.

Metropolis-Hastings Algorithm

The *Metropolis-Hastings* (MH) algorithm employs a transition PDF $f(\tilde{\boldsymbol{\theta}}|\boldsymbol{\theta}^{(i)})$ as follows to sample from the target PDF $\pi(\boldsymbol{\theta})$:

$$f(\tilde{\boldsymbol{\theta}}|\boldsymbol{\theta}^{(i)}) = p_a(\boldsymbol{\theta}^{(i)}, \tilde{\boldsymbol{\theta}})p(\tilde{\boldsymbol{\theta}}|\boldsymbol{\theta}^{(i)}) + (1 - P_a(\boldsymbol{\theta}^{(i)}))\delta(\tilde{\boldsymbol{\theta}} - \boldsymbol{\theta}^{(i)}), \tag{2.66}$$

where $p(\tilde{\boldsymbol{\theta}}|\boldsymbol{\theta}^{(i)})$ is the *proposal PDF*, δ is the *Dirac function*, $p_a(\boldsymbol{\theta}^{(i)}, \tilde{\boldsymbol{\theta}})$ is the *acceptance probability* of a candidate sample $\tilde{\boldsymbol{\theta}}$ defined as

$$p_a(\boldsymbol{\theta}^{(i)}, \tilde{\boldsymbol{\theta}}) = \min\left\{1, \frac{\pi(\tilde{\boldsymbol{\theta}})p(\boldsymbol{\theta}^{(i)}|\tilde{\boldsymbol{\theta}})}{\pi(\boldsymbol{\theta}^{(i)})p(\tilde{\boldsymbol{\theta}}|\boldsymbol{\theta}^{(i)})}\right\}, \tag{2.67}$$

and

$$P_a(\boldsymbol{\theta}^{(i)}) = \int p_a(\boldsymbol{\theta}^{(i)}, \tilde{\boldsymbol{\theta}})p(\tilde{\boldsymbol{\theta}}|\boldsymbol{\theta}^{(i)})\mathrm{d}\tilde{\boldsymbol{\theta}}. \tag{2.68}$$

The proposal distribution is typically chosen as Gaussian or uniform distribution. For a symmetric proposal density, $p(\boldsymbol{\theta}^{(i)}|\tilde{\boldsymbol{\theta}}) = p(\tilde{\boldsymbol{\theta}}|\boldsymbol{\theta}^{(i)})$, then the acceptance probability reduces to

$$p_a(\boldsymbol{\theta}^{(i)}, \tilde{\boldsymbol{\theta}}) = \min\left\{1, \frac{\pi(\tilde{\boldsymbol{\theta}})}{\pi(\boldsymbol{\theta}^{(i)})}\right\}. \tag{2.69}$$

It is trivial to show that the transition density in Equation (2.66) satisfies the detailed balance condition when $\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta}^{(i)}$. For the case $\tilde{\boldsymbol{\theta}} \neq \boldsymbol{\theta}^{(i)}$, one has

$$\begin{aligned}\pi(\boldsymbol{\theta}^{(i)})f(\tilde{\boldsymbol{\theta}}|\boldsymbol{\theta}^{(i)}) &= \pi(\boldsymbol{\theta}^{(i)})p_a(\boldsymbol{\theta}^{(i)}, \tilde{\boldsymbol{\theta}})p(\tilde{\boldsymbol{\theta}}|\boldsymbol{\theta}^{(i)}) \\ &= \pi(\boldsymbol{\theta}^{(i)})p(\tilde{\boldsymbol{\theta}}|\boldsymbol{\theta}^{(i)}) \min\left\{1, \frac{\pi(\tilde{\boldsymbol{\theta}})p(\boldsymbol{\theta}^{(i)}|\tilde{\boldsymbol{\theta}})}{\pi(\boldsymbol{\theta}^{(i)})p(\tilde{\boldsymbol{\theta}}|\boldsymbol{\theta}^{(i)})}\right\}.\end{aligned} \tag{2.70}$$

By leveraging the identity

$$b \cdot \min\left\{1, \frac{a}{b}\right\} \equiv a \cdot \min\left\{1, \frac{b}{a}\right\}, \quad a, b > 0, \tag{2.71}$$

Equation (2.70) becomes

$$\begin{aligned}\pi(\boldsymbol{\theta}^{(i)})f(\tilde{\boldsymbol{\theta}}|\boldsymbol{\theta}^{(i)}) &= \pi(\tilde{\boldsymbol{\theta}})p(\boldsymbol{\theta}^{(i)}|\tilde{\boldsymbol{\theta}}) \min\left\{1, \frac{\pi(\boldsymbol{\theta}^{(i)})p(\tilde{\boldsymbol{\theta}}|\boldsymbol{\theta}^{(i)})}{\pi(\tilde{\boldsymbol{\theta}})p(\boldsymbol{\theta}^{(i)}|\tilde{\boldsymbol{\theta}})}\right\} \\ &= \pi(\tilde{\boldsymbol{\theta}})f(\boldsymbol{\theta}^{(i)}|\tilde{\boldsymbol{\theta}}),\end{aligned} \tag{2.72}$$

indicating that the transition PDF in Equation (2.66) fulfills the detailed balance condition regardless of the choice of the proposal PDF.

The MH algorithm consists of two main steps. First, a candidate sample $\tilde{\boldsymbol{\theta}}$ is generated from the proposal PDF $p(\tilde{\boldsymbol{\theta}}|\boldsymbol{\theta}^{(i)})$. Next, $\tilde{\boldsymbol{\theta}}$ is accepted as the new sample $\boldsymbol{\theta}^{(i+1)}$ with probability $p_a(\boldsymbol{\theta}^{(i)}, \tilde{\boldsymbol{\theta}})$, or rejected with the chain remaining the same: $\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}^{(i)}$. The detailed algorithm is given in Algorithm 4.

Algorithm 4 Metropolis-Hastings algorithm [27, Ch. 4]

Input: Target distribution $\pi(\boldsymbol{\theta})$; Proposal PDF $p(\tilde{\boldsymbol{\theta}}|\boldsymbol{\theta}^{(i)})$; Initial sample $\boldsymbol{\theta}^{(0)}$; The number of samples N.

Output: Samples $\{\boldsymbol{\theta}^{(i)} : i = 1, \dots, N\}$ whose stationary distribution is $\pi(\boldsymbol{\theta})$.

```
1: for $i = 0, \dots, N-1$ do
2:     Generate a candidate sample $\tilde{\boldsymbol{\theta}}$ from $p(\tilde{\boldsymbol{\theta}}|\boldsymbol{\theta}^{(i)})$;
3:     Calculate the acceptance probability $p_a(\boldsymbol{\theta}^{(i)}, \tilde{\boldsymbol{\theta}})$ according to Equation (2.67);
4:     Generate a sample $u$ from $\mathcal{U}(0,1)$;
5:     if $u \le p_a(\boldsymbol{\theta}^{(i)}, \tilde{\boldsymbol{\theta}})$ then
6:         Accept $\tilde{\boldsymbol{\theta}}$ and set $\boldsymbol{\theta}^{(i+1)} = \tilde{\boldsymbol{\theta}}$;
7:     else
8:         Reject $\tilde{\boldsymbol{\theta}}$ and set $\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}^{(i)}$;
9:     end if
10: end for
11: return $\{\boldsymbol{\theta}^{(i)} : i = 1, \dots, N\}$.
```

In the context of SuS, the target distribution $\pi(\boldsymbol{\theta}) = f(\boldsymbol{\theta}|\mathcal{F}_j), j = 1, \dots, m-1$. Inserting Equation (2.53) into Equation (2.67) yields

$$\begin{aligned} p_a(\boldsymbol{\theta}^{(i)}, \tilde{\boldsymbol{\theta}}) &= \min\left\{1, \frac{f(\tilde{\boldsymbol{\theta}}|\mathcal{F}_j)p(\boldsymbol{\theta}^{(i)}|\tilde{\boldsymbol{\theta}})}{f(\boldsymbol{\theta}^{(i)}|\mathcal{F}_j)p(\tilde{\boldsymbol{\theta}}|\boldsymbol{\theta}^{(i)})}\right\} \\ &= \min\left\{1, \frac{f(\tilde{\boldsymbol{\theta}})p(\boldsymbol{\theta}^{(i)}|\tilde{\boldsymbol{\theta}})}{f(\boldsymbol{\theta}^{(i)})p(\tilde{\boldsymbol{\theta}}|\boldsymbol{\theta}^{(i)})} I_{\mathcal{F}_j}(\tilde{\boldsymbol{\theta}})\right\} \\ &= \min\left\{1, \frac{f(\tilde{\boldsymbol{\theta}})p(\boldsymbol{\theta}^{(i)}|\tilde{\boldsymbol{\theta}})}{f(\boldsymbol{\theta}^{(i)})p(\tilde{\boldsymbol{\theta}}|\boldsymbol{\theta}^{(i)})}\right\} I_{\mathcal{F}_j}(\tilde{\boldsymbol{\theta}}) \\ &= \tilde{p}_a(\boldsymbol{\theta}^{(i)}, \tilde{\boldsymbol{\theta}}) I_{\mathcal{F}_j}(\boldsymbol{\theta}), \end{aligned} \tag{2.73}$$

where

$$\tilde{p}_a(\boldsymbol{\theta}^{(i)}, \tilde{\boldsymbol{\theta}}) = \min\left\{1, \frac{f(\tilde{\boldsymbol{\theta}})p(\boldsymbol{\theta}^{(i)}|\tilde{\boldsymbol{\theta}})}{f(\boldsymbol{\theta}^{(i)})p(\tilde{\boldsymbol{\theta}}|\boldsymbol{\theta}^{(i)})}\right\}. \tag{2.74}$$

This suggests that the acceptance procedure is divided into two parts: accepting the candidate $\tilde{\boldsymbol{\theta}}$ with probability $\tilde{p}_a(\boldsymbol{\theta}^{(i)}, \tilde{\boldsymbol{\theta}})$ and accepting $\tilde{\boldsymbol{\theta}}$ if it falls in the intermediate failure domain $\mathcal{F}_j$. The MH algorithm for SuS is summarized in Algorithm 5. As is well known, the MH algorithm suffers from inefficiency in high dimensions since the acceptance probability $\tilde{p}_a(\boldsymbol{\theta}^{(i)}, \tilde{\boldsymbol{\theta}})$ decreases significantly with growing dimensions [20, 26].

Component-Wise Metropolis-Hastings Algorithm

To overcome the low acceptance ratio of the MH algorithm for high-dimensional problems, Au and Beck [26] proposed the component-wise MH algorithm in the context of SuS. Instead of sampling from an n-dimensional proposal PDF $p(\tilde{\boldsymbol{\theta}}|\boldsymbol{\theta}^{(i)})$ and accepting the candidate with probability $\tilde{p}_a(\boldsymbol{\theta}^{(i)}, \tilde{\boldsymbol{\theta}})$, the component-wise MH algorithm generates each

Algorithm 5 The MH algorithm for SuS [32]

```
Input: The PDF of uncertain parameters $f(\boldsymbol{\theta})$; Failure domain $\mathcal{F}_j$; Proposal PDF $p(\tilde{\boldsymbol{\theta}}|\boldsymbol{\theta}^{(i)})$;
    Initial sample $\boldsymbol{\theta}^{(0)}$; The number of samples $N$.
Output: Samples $\{\boldsymbol{\theta}^{(i)} : i = 1, \dots, N\}$ whose stationary distribution is $f(\boldsymbol{\theta}|\mathcal{F}_j)$.
 1: for $i = 0, \dots, N-1$ do
 2:     Generate a candidate sample $\tilde{\boldsymbol{\theta}}$ from $p(\tilde{\boldsymbol{\theta}}|\boldsymbol{\theta}^{(i)})$;
 3:     Calculate the acceptance probability $\tilde{p}_a(\boldsymbol{\theta}^{(i)}, \tilde{\boldsymbol{\theta}})$ according to Equation (2.74);
 4:     Generate a sample $u$ from $\mathcal{U}(0, 1)$;
 5:     if $u \leq \tilde{p}_a(\boldsymbol{\theta}^{(i)}, \tilde{\boldsymbol{\theta}})$ then
 6:         Accept $\tilde{\boldsymbol{\theta}}$ and set $\bar{\boldsymbol{\theta}} = \tilde{\boldsymbol{\theta}}$;
 7:     else
 8:         Reject $\tilde{\boldsymbol{\theta}}$ and set $\bar{\boldsymbol{\theta}} = \boldsymbol{\theta}^{(i)}$;
 9:     end if
10:     if $\bar{\boldsymbol{\theta}} \in \mathcal{F}_j$ then
11:         Accept $\bar{\boldsymbol{\theta}}$ and set $\boldsymbol{\theta}^{(i+1)} = \bar{\boldsymbol{\theta}}$;
12:     else
13:         Reject $\bar{\boldsymbol{\theta}}$ and set $\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}^{(i)}$;
14:     end if
15: end for
16: return $\{\boldsymbol{\theta}^{(i)} : i = 1, \dots, N\}$.
```

sample component $\tilde{\theta}_k, k = 1, \dots, n$, from a 1-dimensional proposal density $p_k(\tilde{\theta}_k|\theta_k^{(i)})$ and accepts $\tilde{\theta}_k$ with probability

$$\tilde{p}_{a,k}(\theta_k^{(i)}, \tilde{\theta}_k) = \min\left\{1, \frac{f_k(\tilde{\theta}_k) p_k(\theta_k^{(i)}|\tilde{\theta}_k)}{f_k(\theta_k^{(i)}) p_k(\tilde{\theta}_k|\theta_k^{(i)})}\right\}. \tag{2.75}$$

The component-wise transition density is defined as

$$f_k(\tilde{\theta}_k|\theta_k^{(i)}) = \tilde{p}_{a,k}(\theta_k^{(i)}, \tilde{\theta}_k) p_k(\tilde{\theta}_k|\theta_k^{(i)}) + (1 - P_{a,k}(\theta_k^{(i)}))\delta(\tilde{\theta}_k - \theta_k^{(i)}), \tag{2.76}$$

with

$$P_{a,k}(\theta_k^{(i)}) = \int \tilde{p}_{a,k}(\theta_k^{(i)}, \tilde{\theta}_k) p_k(\tilde{\theta}_k|\theta_k^{(i)}) \mathrm{d}\tilde{\theta}_k. \tag{2.77}$$

Additionally, the random variable space is assumed to be independent:

$$f(\boldsymbol{\theta}) = \prod_{k=1}^{n} f_k(\theta_k). \tag{2.78}$$

The component-wise MH algorithm is presented in Algorithm 6.

To show that the stationary distribution of the generated samples is $f(\boldsymbol{\theta}|\mathcal{F}_j)$, it is necessary to prove that the Markov chain satisfies the detailed balance condition:

$$f(\boldsymbol{\theta}^{(i)}|\mathcal{F}_j) f(\tilde{\boldsymbol{\theta}}|\boldsymbol{\theta}^{(i)}) = f(\tilde{\boldsymbol{\theta}}|\mathcal{F}_j) f(\boldsymbol{\theta}^{(i)}|\tilde{\boldsymbol{\theta}}). \tag{2.79}$$

Algorithm 6 The component-wise MH algorithm for SuS [26]

Input: The PDF of uncertain parameters $f_k(\theta_k), k = 1, \ldots, n$; Failure domain $\mathcal{F}_j$; Proposal PDF $p_k(\tilde{\theta}_k|\theta_k^{(i)}), k = 1, \ldots, n$; Initial sample $\boldsymbol{\theta}^{(0)} = [\theta_1^{(0)}, \ldots, \theta_n^{(0)}]^\mathsf{T}$; The number of samples N.

Output: Samples $\{\boldsymbol{\theta}^{(i)} : i = 1, \ldots, N\}$ whose stationary distribution is $f(\boldsymbol{\theta}|\mathcal{F}_j)$.

```
1:  for $i = 0, \ldots, N-1$ do
2:      for $k = 1, \ldots, n$ do
3:          Generate a sample component $\tilde{\theta}_k$ from $p_k(\tilde{\theta}_k|\theta_k^{(i)})$;
4:          Calculate the acceptance probability $\tilde{p}_{a,k}(\theta_k^{(i)}, \tilde{\theta}_k)$ according to Equation (2.75);
5:          Generate a sample $u$ from $\mathcal{U}(0,1)$;
6:          if $u \leq \tilde{p}_{a,k}(\theta_k^{(i)}, \tilde{\theta}_k)$ then
7:              Accept $\tilde{\theta}_k$ and set $\bar{\theta}_k = \tilde{\theta}_k$;
8:          else
9:              Reject $\tilde{\theta}_k$ and set $\bar{\theta}_k = \theta_k^{(i)}$;
10:         end if
11:     end for
12:     if $\bar{\boldsymbol{\theta}} = [\bar{\theta}_1, \ldots, \bar{\theta}_n]^\mathsf{T} \in \mathcal{F}_j$ then
13:         Accept $\bar{\boldsymbol{\theta}}$ and set $\boldsymbol{\theta}^{(i+1)} = \bar{\boldsymbol{\theta}}$;
14:     else
15:         Reject $\bar{\boldsymbol{\theta}}$ and set $\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}^{(i)}$;
16:     end if
17: end for
18: return $\{\boldsymbol{\theta}^{(i)} : i = 1, \ldots, N\}$.
```

It is trivial to show that this condition holds for $\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta}^{(i)}$ or any candidate does not lie in $\mathcal{F}_j$. Thus, the detailed balance condition remained to be demonstrated reduces to

$$f(\boldsymbol{\theta}^{(i)})f(\tilde{\boldsymbol{\theta}}|\boldsymbol{\theta}^{(i)}) = f(\tilde{\boldsymbol{\theta}})f(\boldsymbol{\theta}^{(i)}|\tilde{\boldsymbol{\theta}}), \quad \boldsymbol{\theta}^{(i)}, \tilde{\boldsymbol{\theta}} \in \mathcal{F}_j, \tilde{\boldsymbol{\theta}} \neq \boldsymbol{\theta}^{(i)}. \tag{2.80}$$

For each component, if $\tilde{\theta}_k \neq \theta_k^{(i)}$, one has

$$\begin{aligned} f_k(\theta_k^{(i)})f_k(\tilde{\theta}_k|\theta_k^{(i)}) &= f_k(\theta_k^{(i)})\tilde{p}_{a,k}(\theta_k^{(i)}, \tilde{\theta}_k)p_k(\tilde{\theta}_k|\theta_k^{(i)}) \\ &= f_k(\theta_k^{(i)})p_k(\tilde{\theta}_k|\theta_k^{(i)}) \min\left\{1, \frac{f_k(\tilde{\theta}_k)p_k(\theta_k^{(i)}|\tilde{\theta}_k)}{f_k(\theta_k^{(i)})p_k(\tilde{\theta}_k|\theta_k^{(i)})}\right\} \\ &= f_k(\tilde{\theta}_k)p_k(\theta_k^{(i)}|\tilde{\theta}_k) \min\left\{1, \frac{f_k(\theta_k^{(i)})p_k(\tilde{\theta}_k|\theta_k^{(i)})}{f_k(\tilde{\theta}_k)p_k(\theta_k^{(i)}|\tilde{\theta}_k)}\right\} \\ &= f_k(\tilde{\theta}_k)f_k(\theta_k^{(i)}|\tilde{\theta}_k), \end{aligned} \tag{2.81}$$

which indicates that the detailed balance condition is satisfied. Besides, it is straightforward to show the fulfillment of the condition if $\tilde{\theta}_k = \theta_k^{(i)}$. Due to the independence of random variables, the transition PDF between two states in $\mathcal{F}_j$ can be expressed as follows:

$$f(\tilde{\boldsymbol{\theta}}|\boldsymbol{\theta}^{(i)}) = \prod_{k=1}^{n} f_k(\tilde{\theta}_k|\theta_k^{(i)}). \tag{2.82}$$

By substituting Equations (2.78) and (2.82) into $f(\boldsymbol{\theta}^{(i)})f(\tilde{\boldsymbol{\theta}}|\boldsymbol{\theta}^{(i)})$, for $\boldsymbol{\theta}^{(i)}, \tilde{\boldsymbol{\theta}} \in \mathcal{F}_j$, one gets

$$\begin{aligned} f(\boldsymbol{\theta}^{(i)})f(\tilde{\boldsymbol{\theta}}|\boldsymbol{\theta}^{(i)}) &= \prod_{k=1}^{n} f_k(\theta_k^{(i)}) f_k(\tilde{\theta}_k|\theta_k^{(i)}) \\ &= \prod_{k=1}^{n} f_k(\tilde{\theta}_k) f_k(\theta_k^{(i)}|\tilde{\theta}_k) = f(\tilde{\boldsymbol{\theta}})f(\boldsymbol{\theta}^{(i)}|\tilde{\boldsymbol{\theta}}). \end{aligned} \tag{2.83}$$

Hence, Equation (2.80) is proved, and the chain $\{\boldsymbol{\theta}^{(i)}\}$ generated by the component-wise MH algorithm satisfies the detailed balance condition.

There exists two phases concerning the acceptance of candidate samples: the previous phase accepts samples with probability $\tilde{p}_{a,k}(\theta_k^{(i)}, \tilde{\theta}_k)$, and the latter phase based on whether the candidates are located in $\mathcal{F}_j$. Since each sample component is rejected independently in the first phase with probability $\tilde{p}_{r,k} = 1 - \tilde{p}_{a,k}(\theta_k^{(i)}, \tilde{\theta}_k)$, the rejection probability in this phase is given by

$$\mathbb{P}[R_1] = \prod_{k=1}^{n} \tilde{p}_{r,k} \to 0 \text{ as } n \to \infty. \tag{2.84}$$

Therefore, the component-wise MH algorithm is suitable for coping with high-dimensional reliability analysis problems.

Gaussian Conditional Sampling

Gaussian conditional sampling, proposed by Papaioannou et al. [32], is another MCMC sampling approach that addresses the low acceptance ratio of the MH algorithm in high dimensions. This technique generates samples from a conditional PDF $\phi_n(\boldsymbol{\xi}|\mathcal{F}_j)$ in the standard normal space, where $\phi_n(\boldsymbol{\xi}) = \prod_{k=1}^{n} \phi(\xi_k)$ denotes the n-dimensional independent standard normal distribution. Any given random vector $\boldsymbol{\Theta}$ can be transformed into an independent standard normal random vector $\boldsymbol{\Xi}$ through the isoprobabilistic transformation (see Section 2.1.3). The Gaussian conditional sampling algorithm draws samples $\tilde{\boldsymbol{\xi}}$ from a proposal density which is an n-dimensional independent normal distribution with mean $\boldsymbol{R}\boldsymbol{\xi}^{(i)}$ and covariance $\boldsymbol{I} - \boldsymbol{R}\boldsymbol{R}^\intercal$, denoted by $\phi_n(\tilde{\boldsymbol{\xi}} - \boldsymbol{R}\boldsymbol{\xi}^{(i)}; \boldsymbol{I} - \boldsymbol{R}\boldsymbol{R}^\intercal)$, where $\boldsymbol{I}$ is the identity matrix, $\boldsymbol{R} = \mathrm{diag}\{\rho_1, \ldots, \rho_n\}$ is a correlation matrix, and $\rho_k, k = 1, \ldots, n$, are correlation coefficients. The proposal density is represented as

$$\begin{aligned} p(\tilde{\boldsymbol{\xi}}|\boldsymbol{\xi}^{(i)}) &= \phi_n(\tilde{\boldsymbol{\xi}} - \boldsymbol{R}\boldsymbol{\xi}^{(i)}; \boldsymbol{I} - \boldsymbol{R}\boldsymbol{R}^\intercal) \\ &= \prod_{k=1}^{n} \phi(\tilde{\xi}_k - \rho_k \xi_k^{(i)}; 1 - \rho_k^2) \\ &= \prod_{k=1}^{n} p_k(\tilde{\xi}_k|\xi_k^{(i)}), \end{aligned} \tag{2.85}$$

Algorithm 7 The Gaussian conditional sampling algorithm for SuS [32]

```
Input: Failure domain $\mathcal{F}_j$; Initial sample $\boldsymbol{\xi}^{(0)}$; Correlation coefficient $\rho_k, k = 1, \ldots, n$; The
    number of samples $N$.
Output: Samples $\{\boldsymbol{\xi}^{(i)} : i = 1, \ldots, N\}$ whose stationary distribution is $\phi_n(\boldsymbol{\xi}|\mathcal{F}_j)$.
 1: for $i = 0, \ldots, N-1$ do
 2:     for $k = 1, \ldots, n$ do
 3:         Generate a sample component $\tilde{\xi}_k$ from $p_k(\tilde{\xi}_k|\xi_k^{(i)})$ as in Equation (2.86);
 4:     end for
 5:     if $\tilde{\boldsymbol{\xi}} \in \mathcal{F}_j$ then
 6:         Accept $\tilde{\boldsymbol{\xi}}$ and set $\boldsymbol{\xi}^{(i+1)} = \tilde{\boldsymbol{\xi}}$;
 7:     else
 8:         Reject $\tilde{\boldsymbol{\xi}}$ and set $\boldsymbol{\xi}^{(i+1)} = \boldsymbol{\xi}^{(i)}$;
 9:     end if
10: end for
11: return $\{\boldsymbol{\xi}^{(i)} : i = 1, \ldots, N\}$.
```

with

$$p_k(\tilde{\xi}_k|\xi_k^{(i)}) = \phi(\tilde{\xi}_k - \rho_k \xi_k^{(i)}; 1 - \rho_k^2). \tag{2.86}$$

The transition PDF is given by

$$f(\tilde{\boldsymbol{\xi}}|\boldsymbol{\xi}^{(i)}) = \phi_n(\tilde{\boldsymbol{\xi}} - \boldsymbol{R}\boldsymbol{\xi}^{(i)}; \boldsymbol{I} - \boldsymbol{R}\boldsymbol{R}^\mathsf{T}) I_{\mathcal{F}_j}(\tilde{\boldsymbol{\xi}}) + (1 - P_a(\boldsymbol{\xi}^{(i)}))\delta(\tilde{\boldsymbol{\xi}} - \boldsymbol{\xi}^{(i)}), \tag{2.87}$$

in which

$$P_a(\boldsymbol{\xi}^{(i)}) = \int \phi_n(\tilde{\boldsymbol{\xi}} - \boldsymbol{R}\boldsymbol{\xi}^{(i)}; \boldsymbol{I} - \boldsymbol{R}\boldsymbol{R}^\mathsf{T}) I_{\mathcal{F}_j}(\tilde{\boldsymbol{\xi}}) \mathrm{d}\tilde{\boldsymbol{\xi}}. \tag{2.88}$$

The algorithm is summarized in Algorithm 7.

To show that the Markov chain generated by this algorithm satisfies the detailed balance condition, it suffices to demonstrate that

$$\phi_n(\boldsymbol{\xi}^{(i)}|\mathcal{F}_j) f(\tilde{\boldsymbol{\xi}}|\boldsymbol{\xi}^{(i)}) = \phi_n(\tilde{\boldsymbol{\xi}}|\mathcal{F}_j) f(\boldsymbol{\xi}^{(i)}|\tilde{\boldsymbol{\xi}}). \tag{2.89}$$

It is straightforward to show that this condition is fulfilled if $\tilde{\boldsymbol{\xi}} = \boldsymbol{\xi}^{(i)}$ or any candidate is not in $\mathcal{F}$. Hence, the detailed balance condition remained to be proved reduces to

$$\phi_n(\boldsymbol{\xi}^{(i)})\phi_n(\tilde{\boldsymbol{\xi}} - \boldsymbol{R}\boldsymbol{\xi}^{(i)}; \boldsymbol{I} - \boldsymbol{R}\boldsymbol{R}^\mathsf{T}) = \phi_n(\tilde{\boldsymbol{\xi}})\phi_n(\boldsymbol{\xi}^{(i)} - \boldsymbol{R}\tilde{\boldsymbol{\xi}}; \boldsymbol{I} - \boldsymbol{R}\boldsymbol{R}^\mathsf{T}), \quad \boldsymbol{\xi}^{(i)}, \tilde{\boldsymbol{\xi}} \in \mathcal{F}_j, \tilde{\boldsymbol{\xi}} \neq \boldsymbol{\xi}^{(i)}. \tag{2.90}$$

Consider a $2n$-dimensional normal random vector with zero mean as follows:

$$\begin{bmatrix} \Xi_1 \\ \Xi_2 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \boldsymbol{I} & \boldsymbol{R} \\ \boldsymbol{R}^\mathsf{T} & \boldsymbol{I} \end{bmatrix} \right), \tag{2.91}$$

where Ξ_1 and Ξ_2 are both n-dimensional normal random vectors. Then the conditional distribution of Ξ_2 given $\Xi_1 = \boldsymbol{\xi}_1$ is

$$\Xi_2|\Xi_1 = \boldsymbol{\xi}_1 \sim \mathcal{N}\left(\boldsymbol{R}\boldsymbol{\xi}_1, \boldsymbol{I} - \boldsymbol{R}\boldsymbol{R}^\mathsf{T}\right). \tag{2.92}$$

Consequently, the joint PDF of $[\Xi_1, \Xi_2]^\mathsf{T}$ can be expressed as

$$\phi_{2n}(\boldsymbol{\xi}; \boldsymbol{\Sigma}) = \phi_n(\boldsymbol{\xi}_1)\phi_n(\boldsymbol{\xi}_2 - \boldsymbol{R}\boldsymbol{\xi}_1; \boldsymbol{I} - \boldsymbol{R}\boldsymbol{R}^\mathsf{T}), \tag{2.93}$$

where $\boldsymbol{\xi} = [\boldsymbol{\xi}_1, \boldsymbol{\xi}_2]^\mathsf{T}$ and

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{R} \\ \boldsymbol{R}^\mathsf{T} & \boldsymbol{I} \end{bmatrix}. \tag{2.94}$$

Similarly, one has

$$\Xi_1 | \Xi_2 = \boldsymbol{\xi}_2 \sim \mathcal{N}\left(\boldsymbol{R}\boldsymbol{\xi}_2, \boldsymbol{I} - \boldsymbol{R}\boldsymbol{R}^\mathsf{T}\right), \tag{2.95}$$

$$\phi_{2n}(\boldsymbol{\xi}; \boldsymbol{\Sigma}) = \phi_n(\boldsymbol{\xi}_2)\phi_n(\boldsymbol{\xi}_1 - \boldsymbol{R}\boldsymbol{\xi}_2; \boldsymbol{I} - \boldsymbol{R}\boldsymbol{R}^\mathsf{T}). \tag{2.96}$$

Equations (2.93) and (2.96) indicate that

$$\phi_n(\boldsymbol{\xi}_1)\phi_n(\boldsymbol{\xi}_2 - \boldsymbol{R}\boldsymbol{\xi}_1; \boldsymbol{I} - \boldsymbol{R}\boldsymbol{R}^\mathsf{T}) = \phi_n(\boldsymbol{\xi}_2)\phi_n(\boldsymbol{\xi}_1 - \boldsymbol{R}\boldsymbol{\xi}_2; \boldsymbol{I} - \boldsymbol{R}\boldsymbol{R}^\mathsf{T}). \tag{2.97}$$

Setting $\boldsymbol{\xi}_1 = \boldsymbol{\xi}^{(i)}$ and $\boldsymbol{\xi}_2 = \tilde{\boldsymbol{\xi}}$ shows that Equation (2.90) is satisfied. This demonstrates that the stationary distribution of the Markov chain $\{\boldsymbol{\xi}^{(i)}\}$ generated by the Gaussian conditional sampling algorithm is $\phi_n(\boldsymbol{\xi}|\mathcal{F}_j)$.

Recall that there are two phases concerning the acceptance of candidate samples in Algorithm 5 and Algorithm 6. Also, the acceptance ratio in the first phase depends on the dimension of uncertain parameters. For Gaussian conditional sampling, the acceptance probability is

$$p_a(\boldsymbol{\xi}^{(i)}, \tilde{\boldsymbol{\xi}}) = I_{\mathcal{F}_j}(\tilde{\boldsymbol{\xi}}), \tag{2.98}$$

which suggests that the candidates are always accepted (acceptance ratio equals 1) in the first phase and no longer depends on the dimensions.

The acceptance rate of the Gaussian conditional sampling method relies on the choice of the correlation coefficient $\rho_k \in [0, 1]$. Using a small ρ_k results in a relatively low acceptance ratio $p_a(\boldsymbol{\xi}^{(i)}, \tilde{\boldsymbol{\xi}})$, thus bringing about a large correlation among the generated samples $\{\boldsymbol{\xi}^{(i)}\}$. However, choosing a larger ρ_k improves the acceptance ratio but leads to a high dependency among the states of a chain, which also causes a large correlation. Thus, it is a tradeoff between a high acceptance rate and a low correlation. Papaioannou et al. [32] have shown that the optimal acceptance rate $p_a^* \approx 0.44$ in terms of chain efficiency, which measures the convergence rate of the Markov chain.

The work in [32] presents an adaptive Gaussian conditional sampling algorithm which adjusts ρ_k in real time such that the acceptance rate is always around p_a^*. At each level of SuS, N_s seeds are determined and each seed generates a Markov chain with $1/p_0$ samples. Instead of creating N_s Markov chains in parallel, N_a chains are generated first, where N_a and N_s/N_a are both positive integers. After that, the average acceptance ratio of the N_a chains is evaluated. If the average acceptance probability is larger than p_a^*, smaller ρ_k that can decrease the acceptance rate are used. Conversely, larger ρ_k are adopted if the average acceptance probability is smaller than p_a^*. This adaptation is conducted when every N_a chains are generated. By this means, the acceptance rate of the N_s Markov chains remains around p_a^*. Specifically, the adaptation is achieved by choosing suitable standard deviations

$$\sigma_k = \sqrt{1-\rho_k^2}. \tag{2.99}$$

They are updated when every N_a chains have been generated:

$$\sigma_{l,k} = \min\{1, \lambda_l \sigma_{l-1,k}\}, \tag{2.100}$$

where the adaptation step $l = 1, \ldots, N_s/N_a$ and $\lambda_l \in (0,1)$ is a scaling factor. The starting values $\sigma_{0,k}$ can either be prescribed constants or the standard deviations of the components of the seeds. Then the correlation coefficients are computed by

$$\rho_k = \sqrt{1-\sigma_{l,k}^2}. \tag{2.101}$$

The average acceptance ratio is assessed by

$$\hat{p}_{a,l} = \frac{1}{N_a}\sum_{r=1}^{N_a} \hat{p}_a(\boldsymbol{\xi}_s^{(r+(l-1)N_a)}), \tag{2.102}$$

where $\hat{p}_a(\boldsymbol{\xi}_s^{(r+(l-1)N_a)})$ is the acceptance rate of the chain starting from the seed $\boldsymbol{\xi}_s^{(r+(l-1)N_a)}$. The scaling factor λ_l is updated by

$$\log \lambda_{l+1} = \log \lambda_l + \zeta_l(\hat{p}_{a,l} - p_a^*), \tag{2.103}$$

where ζ_l is a decreasing positive number and $\zeta_l = 1/\sqrt{l}$ here. The adaptive Gaussian conditional sampling algorithm is given in Algorithm 8.

2.5.3 Statistics of the Estimator

The failure probability estimate $\hat{P}_F$ in Equation (2.54) is asymptotically unbiased with bias of order $\mathcal{O}(1/N)$ [26, 133]. This results from the correlation between the estimates of conditional failure probabilities, and the correlation arises from the fact that the samples $\boldsymbol{\theta}_{j-1}^{(i)} \in \mathcal{F}_j$ are selected as seeds for the next subset level. Nonetheless, the bias is negligible in comparison with the coefficient of variation (c.o.v.) of the estimate.

The c.o.v. of the probability estimate $\hat{P}_F$ has been approximated to measure the estimation accuracy in [26, 27, 132]. Let $\hat{P}_j$ denote the estimate of the conditional failure probability $\mathbb{P}[\mathcal{F}_j|\mathcal{F}_{j-1}]$, $j = 1, \ldots, m$. The c.o.v. of $\hat{P}_j$ is estimated by

$$\delta_j \approx \sqrt{\frac{1-\hat{P}_j}{N\hat{P}_j}(1+\gamma_j)}, \tag{2.104}$$

Algorithm 8 The adaptive Gaussian conditional sampling algorithm for SuS [32]

```
Input: Failure domain $\mathcal{F}_j$; Initial samples $\{\boldsymbol{\xi}_s^{(i)} : i = 1, \ldots, N_s\}$; Starting values of the
    standard deviations $\boldsymbol{\sigma}_0 = [\sigma_{0,1}, \ldots, \sigma_{0,n}]^\top$ (optional); Adaptation interval $N_a$; Initial
    scaling factor $\lambda_1$; The number of samples $N$.
Output: Samples $\{\boldsymbol{\xi}^{(i)} : i = 1, \ldots, N\}$ whose stationary distribution is $\phi_n(\boldsymbol{\xi}|\mathcal{F}_j)$.
 1: for $i = 0, \ldots, N-1$ do
 2:     if $\boldsymbol{\sigma}_0$ is undefined then
 3:         for $k = 1, \ldots, n$ do
 4:             Compute the standard deviation $\hat{\sigma}_k$ of the $k$-th component of the seeds;
 5:             Set $\sigma_{0,k} = \hat{\sigma}_k$;
 6:         end for
 7:     end if
 8:     for $l = 1, \ldots, N_s/N_a$ do
 9:         for $k = 1, \ldots, n$ do
10:             Calculate $\sigma_{l,k}$ according to Equation (2.100);
11:             Compute the correlation coefficient $\rho_k$ based on Equation (2.101);
12:         end for
13:         for $r = (l-1)N_a + 1, \ldots, lN_a$ do
14:             Generate $1/p_0$ samples $\{\boldsymbol{\xi}^{(r-1)/p_0+t} : t = 1, \ldots, 1/p_0\}$ from $\boldsymbol{\xi}_s^{(r)}$ implementing
                the Gaussian conditional sampling algorithm in Algorithm 7;
15:         end for
16:         Assess the average acceptance ratio $\hat{p}_{a,l}$ of the $N_a$ chains as in Equation (2.102);
17:         Update the scaling parameter $\lambda_{l+1}$ in accordance with Equation (2.103);
18:     end for
19: end for
20: return $\{\boldsymbol{\xi}^{(i)} : i = 1, \ldots, N\}$.
```

where γ_j is a factor accounting for the correlation among the MCMC samples at level $j-1$ and

$$\begin{aligned} \gamma_1 &= 0, \\ \gamma_j &= 2 \sum_{k=1}^{N/N_s-1} \left(1 - \frac{kN_s}{N}\right) \rho_j(k), \quad j = 2, \ldots, m. \end{aligned} \tag{2.105}$$

Here, $\rho_j(k)$ are the k-lag auto-correlation coefficients which are averaged based on the N_s chains. They can be estimated by

$$\rho_j(k) \approx \frac{1}{\hat{P}_j(1-\hat{P}_j)} \left[\frac{1}{N - kN_s} \sum_{l=1}^{N_s} \sum_{r=1}^{N/N_s-k} I_{\mathcal{F}_j}\left(\boldsymbol{\theta}_{j-1}^{(l,r)}\right) I_{\mathcal{F}_j}\left(\boldsymbol{\theta}_{j-1}^{(l,r+k)}\right) - \hat{P}_j^2 \right], \tag{2.106}$$

wherein $\boldsymbol{\theta}_{j-1}^{(l,r)}$ denotes the r-th sample in the l-th Markov chain at level $j-1$.

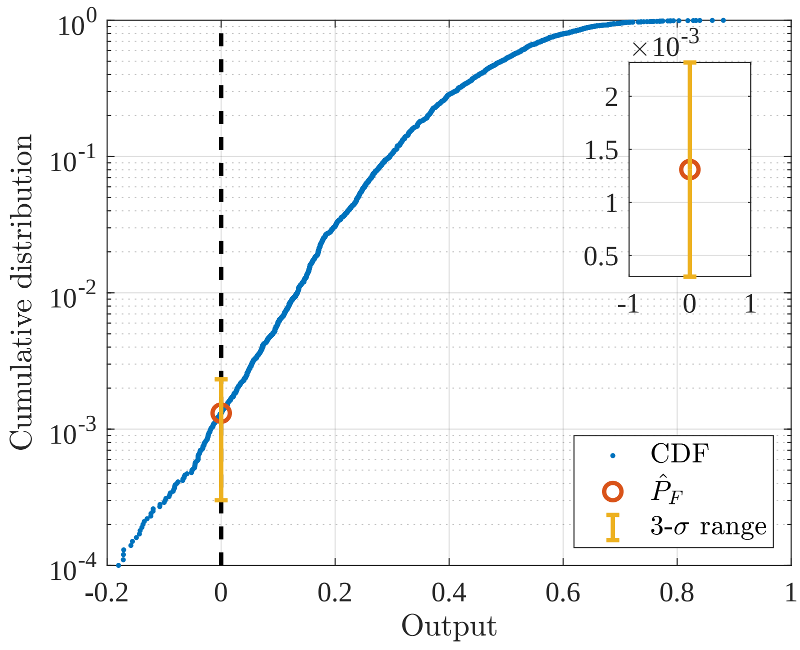


Figure 2.14: *Failure probability estimation results using SuS.*

The first-order estimate of the c.o.v. of $\hat{P}_F$ is given by

$$c_v \approx \sqrt{\sum_{j=1}^{m}\sum_{k=1}^{m} \delta_j \delta_k \rho_{jk}}, \tag{2.107}$$

where ρ_{jk} represents the correlation coefficient between $\hat{P}_j$ and $\hat{P}_k$. By assuming that all the estimates $\hat{P}_j$ and $\hat{P}_k$ are uncorrelated, i.e., $\rho_{jk} = 0$ for all $j \neq k$, or completely correlated, i.e., $\rho_{jk} = 1$ for all $j \neq k$, the lower and upper bounds of c_v are approximated, respectively, by

$$\begin{aligned} c_{v,\mathrm{lb}} &\approx \sqrt{\sum_{j=1}^{m} \delta_j^2}, \\ c_{v,\mathrm{ub}} &\approx \sqrt{\sum_{j=1}^{m}\sum_{k=1}^{m} \delta_j \delta_k}. \end{aligned} \tag{2.108}$$

2.5.4 Illustrative Example

The reliability problem in Section 2.4.4 is solved by SuS with the following parameters: $N = 1000$, $p_0 = 0.1$, and $a = 2.49$. The component-wise MH algorithm is applied in this example. All the SuS results, as well as the surrogate-accelerated SuS results in the following chapters, are obtained using the *Subset Simulation* toolbox [134]. Figure 2.11 presents the sample points at each subset level. These samples approach the failure domain level by level. Figure 2.14 shows the probability estimation results, where σ is the standard deviation of the estimation approximated by Equation (2.36) with $\mu \approx \hat{P}_F$ and $c_v \approx \hat{c}_{v,\mathrm{lb}}$. The 3-$\sigma$ range can be narrowed down by increasing the number of samples per level N.

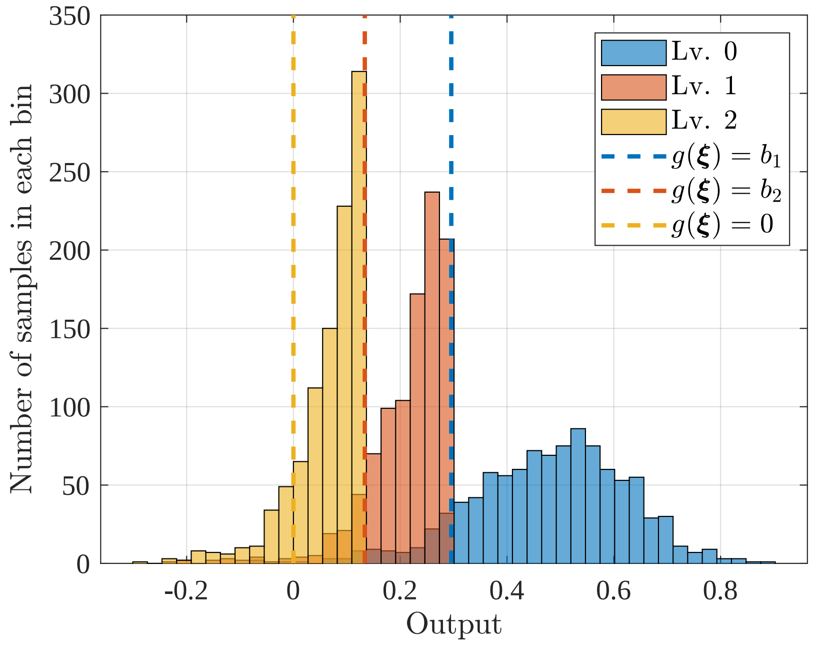


Figure 2.15: *Function outputs at each level of SuS.*

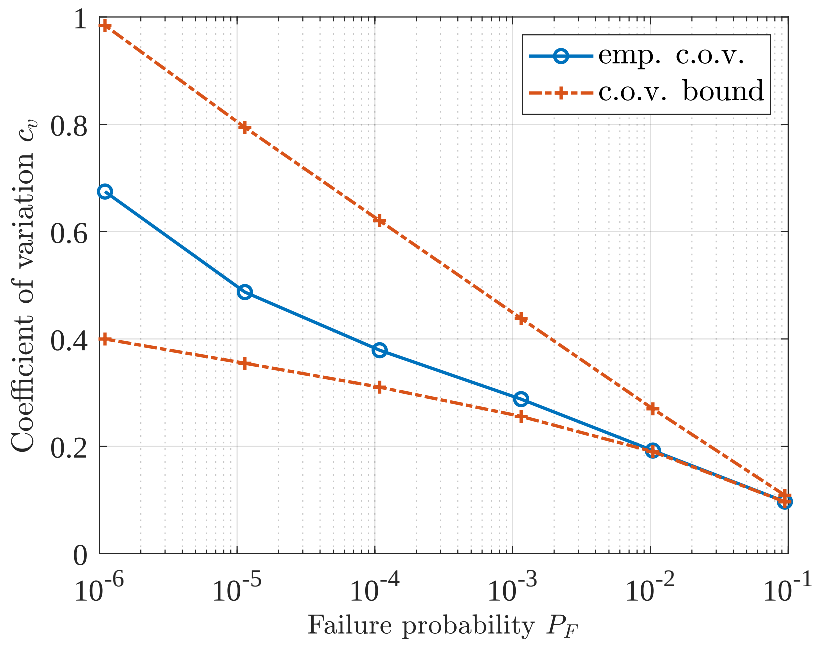


Figure 2.16: *C.o.v. estimates for SuS with $N = 1000$ and $p_0 = 0.1$.*

Figure 2.15 depicts the histogram of the function responses for different subset levels. The dashed lines represent the intermediate failure thresholds. By leveraging MCMC sampling techniques, the samples generated at level 1 and level 2 cannot exceed the corresponding thresholds. This enables the SuS method to reach the failure domain and explore the distribution tail efficiently.

The c.o.v. estimates for probabilities of different orders of magnitude are shown in Figure 2.16. The corresponding a are listed in Table 2.1. For each a, 100 independent

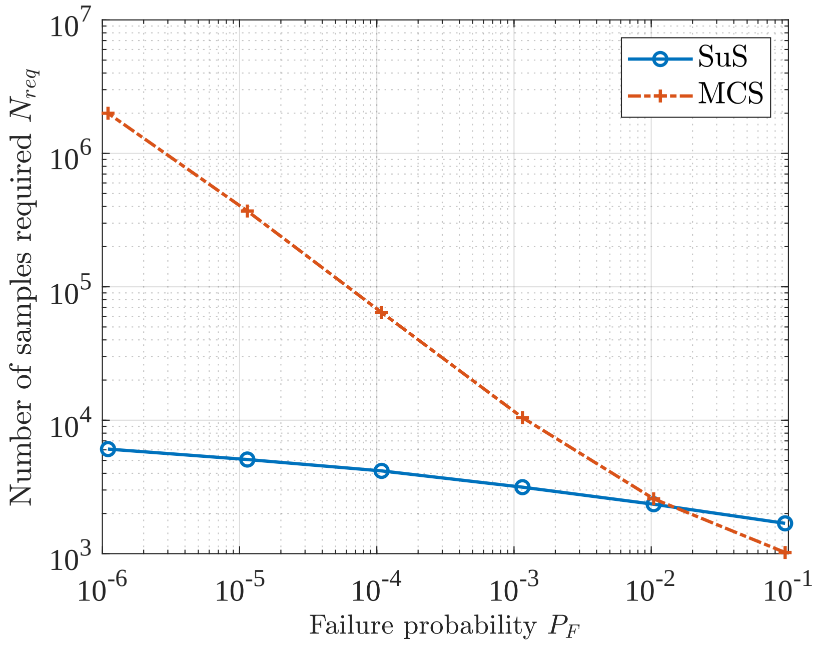


Figure 2.17: *Number of samples required by SuS and MCS*

applications of SuS are conducted. In each simulation, both upper and lower bounds are evaluated as in Section 2.5.3. The c.o.v. upper and lower bounds in this figure are the average values of the 100 results. The empirical c.o.v., which is denoted by "emp. c.o.v." in the figure legend, is the sample c.o.v. of the 100 estimates of failure probability. The empirical c.o.v. results lie between the upper and lower bounds, which demonstrates the validity of the c.o.v. estimation in Section 2.5.3. For higher probabilities, the empirical c.o.v. is closer to the lower bound, since there is less correlation between subset levels.

Figure 2.17 illustrates the efficiency of SuS via comparing the number of samples required by SuS and MCS. The required number of samples for SuS is the average number of model evaluations over the 100 runs. For MCS, to have a fair comparison, the number of samples required to achieve the same accuracy level as in SuS is calculated using Equation (2.39). The results indicate that SuS needs considerably fewer samples for estimating small probabilities ($\leq 10^{-2}$), thus gains much higher efficiency.

Chapter 3

Reliability-Based Control Optimization and Surrogate-Assisted Reliability Analysis

This chapter presents a novel control design framework called *reliability-based control optimization* (RBCO). It is a verification-driven approach that deals with probabilistic requirements in a less conservative way when comparing with conventional control design methods. In the following sections, system dynamics and the performance function are first introduced. After that, the RBCO framework incorporating reliability analysis and optimization is presented. In the end, surrogate modeling techniques are applied to accelerate the reliability assessment procedure.

3.1 System Dynamics and Performance Function

Consider a class of closed-loop dynamic systems subject to parametric uncertainties:

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\theta}, \boldsymbol{k}), \\ \boldsymbol{y} &= \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{\theta}), \end{aligned} \tag{3.1}$$

where $\boldsymbol{x} \in \mathbb{R}^{n_x}$ is the *state variable*, $\boldsymbol{u} \in \mathbb{R}^{n_u}$ is the *system input*, and $\boldsymbol{y} \in \mathbb{R}^{n_y}$ is the *controlled output*. The *uncertain parameters* are represented by $\boldsymbol{\theta} \in \mathbb{R}^{n}$ with PDF $f(\boldsymbol{\theta})$. The *control parameters*, denoted by $\boldsymbol{k} \in \mathbb{R}^{n_k}$, are assumed to be uncorrelated with $\boldsymbol{\theta}$. The *state transition function* $\boldsymbol{f}(\cdot)$ and *output function* $\boldsymbol{h}(\cdot)$ represent the closed-loop system dynamics and are potentially unknown.

The requirements for deterministic systems are usually formulated as inequalities in terms of target values for given metrics. The *performance function* $g(\boldsymbol{k})$ is defined as the margin to the threshold of the target value. If $g(\boldsymbol{k}) < 0$, then the system fails to satisfy

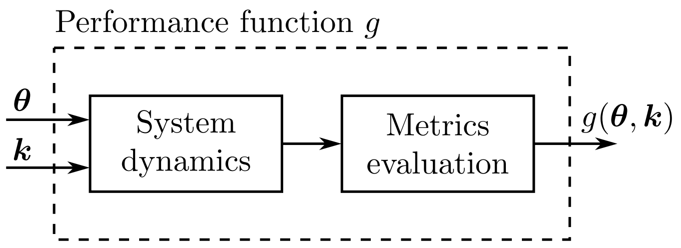


Figure 3.1: *Performance function.*

the requirement, otherwise it meets the requirement. For instance, if the requirement for a linear system is that the overshoot $\sigma\%$ must be smaller than 10%, the corresponding performance function can be defined as $g(\boldsymbol{k}) = 10\% - \sigma\%$. For nondeterministic systems with uncertainties, the requirements are often specified as inequalities where admissible probabilities are imposed on the occurrence of the *failure event* $g(\boldsymbol{\theta}, \boldsymbol{k}) < 0$. If the system requirement, for example, is that the probability of overshoot exceeding 10% must not be larger than 10^{-3}, then the probabilistic requirement is described as inequality: $\mathbb{P}[g(\boldsymbol{\theta}, \boldsymbol{k}) < 0] \leq 10^{-3}$, with performance function $g(\boldsymbol{\theta}, \boldsymbol{k}) = 10\% - \sigma\%$.

Figure 3.1 illustrates the concept of performance function. It is a static system that integrates the system dynamics and the evaluation of metrics. Herein, the metrics can be either time-domain or frequency-domain specifications, such as the minimum distance between aircraft [37], overshoot [39], the root mean square of the measurement error [40], and the stability margin [135, p. 26]. Due to the potential complexity of the system dynamics and the metrics evaluation process, the performance function is regarded as a black-box system.

3.2 Reliability-Based Control Optimization

Given the plant model, control structure, uncertainty model, and design requirements, *reliability-based control optimization* (RBCO) searches for control parameters within the region where the occurrence probabilities of failure events satisfy the design requirements. This can be characterized as a constrained optimization problem of the form:

$$\begin{aligned} \min_{\boldsymbol{k}} \quad & c_0(\boldsymbol{k}), \\ \text{s.t.} \quad & c_i(\boldsymbol{k}) \leq 0, \quad i = 1, \ldots, n_c, \\ & \mathbb{P}[g_i(\boldsymbol{\theta}, \boldsymbol{k}) < 0] \leq \beta_i, \quad i = 1, \ldots, n_g, \end{aligned} \tag{3.2}$$

where $c_0(\boldsymbol{k})$ is the *deterministic objective function*, $c_i(\boldsymbol{k}) \leq 0$, $i = 1, \ldots, n_c$, are *standard constraints*, and $\mathbb{P}[g_i(\boldsymbol{\theta}, \boldsymbol{k}) < 0] \leq \beta_i$, $i = 1, \ldots, n_g$, are *reliability constraints*, where β_i, $i = 1, \ldots, n_g$, are the minimum safety requirements in terms of *acceptable failure probabilities*. The objective function can also be the probability of a failure event $\mathbb{P}[g_0(\boldsymbol{\theta}, \boldsymbol{k}) < 0]$. The standard constraints impose a bound on the admissible design space, in which

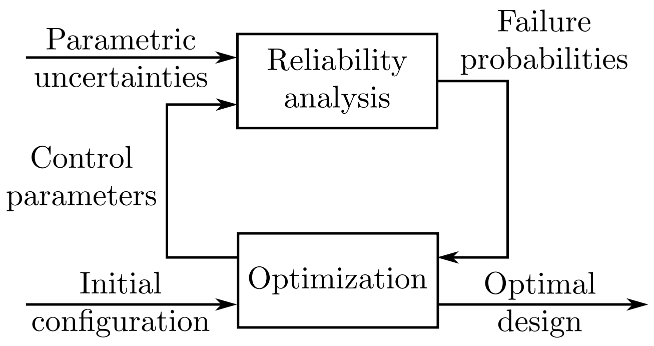


Figure 3.2: *Schema of the two-level approach to the RBCO problem.*

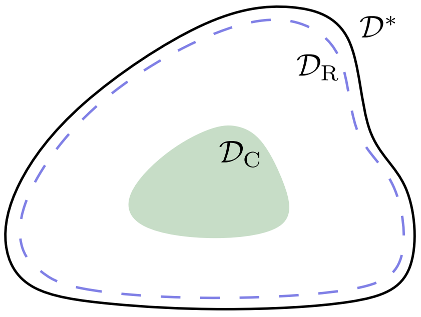


Figure 3.3: *Design spaces of different strategies.*

the *deterministic functions* $c_i(\boldsymbol{k})$ are usually analytical or can be simply evaluated in one simulation run. In contrast, the reliability constraints set thresholds on the failure probabilities for each failure mode, whereby it is necessary for the performance functions $g_i(\boldsymbol{\theta}, \boldsymbol{k})$ to be evaluated many times to estimate the failure probabilities.

In this thesis, a two-level approach is implemented to solve the RBCO problem in Equation (3.2). It consists of two loops, of which the inner one estimates failure probabilities and the outer one explores the design space, as shown in Figure 3.2. The reliability analysis is discussed in the next section, and global search algorithms are implemented to explore the entire design domain.

In conventional methods where controllers are designed in the presence of uncertainties, verification is required to verify compliance with probabilistic requirements. The two-level approach for RBCO searches for control parameters on the basis of the verification results, thus directly ensuring that the probabilistic requirements are fulfilled. Furthermore, these conventional control design methods usually treat chance constraints in a conservative manner, thereby reducing the design space and sacrificing potential performance. The *design space* $\mathcal{D}$ here is defined as all the control parameters $\boldsymbol{k}$ that satisfy the given chance constraints. Worse still, these methods may not even find a solution due to the conservative treatment of each chance constraint. In comparison with these conservative treatments, the proposed RBCO framework estimates the failure probability more precisely, which enables us to enlarge the design space and explore further performance. Figure 3.3 illustrates the design spaces of different strategies. Here, $\mathcal{D}^*$ denotes the *feasible design* *space*, $\mathcal{D}_{\mathrm{R}}$ denotes the *design space obtained by the RBCO framework*, and $\mathcal{D}_{\mathrm{C}}$ denotes the *design space achieved by the conventional control design methods*. Though RBCO is less conservative, there is still a gap between $\mathcal{D}_{\mathrm{R}}$ and $\mathcal{D}^*$ owing to the error of the reliability analysis. The gap can be narrowed down by increasing the estimation accuracy of the failure probability, but this demands higher computational cost. Therefore, it is a tradeoff between the conservatism and computational efficiency.

The concepts of design spaces in Figure 3.3 are exemplified as follows. Assume that the system performance $g(\boldsymbol{\theta}, \boldsymbol{k})$ is normally distributed with zero mean, and the control parameters are required to be tuned such that

$$\mathbb{P}\left[g(\boldsymbol{\theta}, \boldsymbol{k}) \geq 1\right] \leq 0.1. \tag{3.3}$$

The control design methods in [115, 122] approximate chance constraints in a conservative manner by converting them into relaxed expressions using the Cantelli-Chebyshev inequality:

$$\mathbb{P}\left[z \geq \mathbb{E}\left[z\right] + \lambda\right] \leq \frac{\mathrm{Var}\left[z\right]}{\mathrm{Var}\left[z\right] + \lambda^2}, \tag{3.4}$$

where z is a random variable with finite variance, and λ is a positive number. By applying this inequality and letting $z = g(\boldsymbol{\theta}, \boldsymbol{k})$ and $\lambda = 1$, the requirement in Equation (3.3) is converted to

$$\frac{\mathrm{Var}\left[g(\boldsymbol{\theta}, \boldsymbol{k})\right]}{\mathrm{Var}\left[g(\boldsymbol{\theta}, \boldsymbol{k})\right] + 1} \leq 0.1. \tag{3.5}$$

Therefore, the design space is given by

$$\mathcal{D}_{\mathrm{C}} = \left\{\boldsymbol{k} : \mathrm{Var}\left[g(\boldsymbol{\theta}, \boldsymbol{k})\right] \leq 0.111\right\}. \tag{3.6}$$

However, with the knowledge of the distribution of the system performance, Equation (3.3) can be rewritten as

$$\mathbb{P}\left[g(\boldsymbol{\theta}, \boldsymbol{k}) \geq 1\right] = 1 - \Phi\left(\frac{1}{\sqrt{\mathrm{Var}\left[g(\boldsymbol{\theta}, \boldsymbol{k})\right]}}\right) \leq 0.1. \tag{3.7}$$

The feasible design space is thus obtained as follows:

$$\mathcal{D}^* = \left\{\boldsymbol{k} : \mathrm{Var}\left[g(\boldsymbol{\theta}, \boldsymbol{k})\right] \leq 0.609\right\}. \tag{3.8}$$

For RBCO, assume that the probability is estimated by MCS or SuS with an accuracy $c_v = 0.1$. To achieve a high confidence level, we consider the 3-σ upper bound of the estimate

$$(1 + 3c_v)\hat{\mathbb{P}}\left[g(\boldsymbol{\theta}, \boldsymbol{k}) \geq 1\right] \leq 0.1. \tag{3.9}$$

This corresponds to a more strict requirement than that in Equation (3.7):

$$\hat{\mathbb{P}}\left[g(\boldsymbol{\theta}, \boldsymbol{k}) \geq 1\right] = 1 - \Phi\left(\frac{1}{\sqrt{\mathrm{Var}\left[g(\boldsymbol{\theta}, \boldsymbol{k})\right]}}\right) \leq 0.077, \tag{3.10}$$

thus leading to the design space

$$\mathcal{D}_{\mathrm{R}} = \{\boldsymbol{k} : \mathrm{Var}\left[g(\boldsymbol{\theta}, \boldsymbol{k})\right] \leq 0.492\}\,. \tag{3.11}$$

Obviously, the use of the Cantelli-Chebyshev inequality leads to a conservative design, and the design space $\mathcal{D}^* - \mathcal{D}_{\mathrm{C}} = \{\boldsymbol{k} : 0.111 < \mathrm{Var}\left[g(\boldsymbol{\theta}, \boldsymbol{k})\right] \leq 0.609\}$ is excluded by this strategy. In essence, the conservative result stems from the fact that the Cantelli-Chebyshev inequality is used to find a sufficient but not necessary condition for the original chance constraint without considering the distribution information. However, the conservative design can be significantly alleviated by the RBCO framework with a precise uncertainty propagation. If the acceptable failure probability in Equation (3.3) is as small as 10^{-3} and the accuracy of the probability estimation is $c_v = 0.3$, then the design spaces are

$$\begin{aligned} \mathcal{D}_{\mathrm{C}} &= \{\boldsymbol{k} : \mathrm{Var}\left[g(\boldsymbol{\theta}, \boldsymbol{k})\right] \leq 0.001\}\,, \\ \mathcal{D}^* &= \{\boldsymbol{k} : \mathrm{Var}\left[g(\boldsymbol{\theta}, \boldsymbol{k})\right] \leq 0.105\}\,, \\ \mathcal{D}_{\mathrm{R}} &= \{\boldsymbol{k} : \mathrm{Var}\left[g(\boldsymbol{\theta}, \boldsymbol{k})\right] \leq 0.093\}\,. \end{aligned} \tag{3.12}$$

The results show that the influence of the conservative design is more severe for rare-event chance constraints, whereas the benefit of the RBCO framework is more prominent.

3.3 Surrogate-Accelerated Reliability Analysis

Simulation methods, such as MCS (see Section 2.3) and SuS (see Section 2.5), are widely used to estimate the failure probability for complex systems without making any hypothesis on the complexity of the performance function. To obtain sufficiently accurate estimations, a large number of simulations are required. Each simulation takes a different combination of uncertain parameters as the inputs of the performance function. Unfortunately, the evaluation of the performance of a complex or high-fidelity model is usually a time-consuming task. As a consequence, a precise uncertainty propagation may require prohibitive computational cost, thus being infeasible in practice.

To enhance the efficiency of simulation methods, many researchers have attempted to reduce the number of calls to the computationally demanding simulation model (also called the *true model*) by combining surrogate modeling techniques with the simulation approaches. A *surrogate model* $\hat{g}(\boldsymbol{\theta})$ is a cheap-to-evaluate model that mimics the behavior of the expensive-to-evaluate true model $g(\boldsymbol{\theta})$, as shown in Figure 3.4. A common idea of surrogate-based simulation methods is to build a surrogate model and then to replace the true model evaluations with the predictions of the surrogate model.

The surrogate model is constructed as illustrated in Figure 3.5. First of all, initial training samples $\boldsymbol{\theta}_t^{(i)}$ are determined. It is preferable to generate training samples spread across the parameter space. This allows us to build a surrogate model that can make

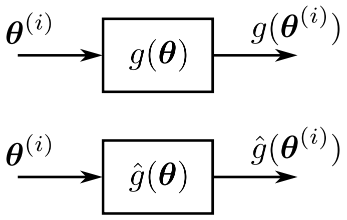

$\boldsymbol{\theta}^{(i)}$
$g(\boldsymbol{\theta})$
$g(\boldsymbol{\theta}^{(i)})$
$\boldsymbol{\theta}^{(i)}$
$\hat{g}(\boldsymbol{\theta})$
$\hat{g}(\boldsymbol{\theta}^{(i)})$

Figure 3.4: *True model g and surrogate model $\hat{g}$.*

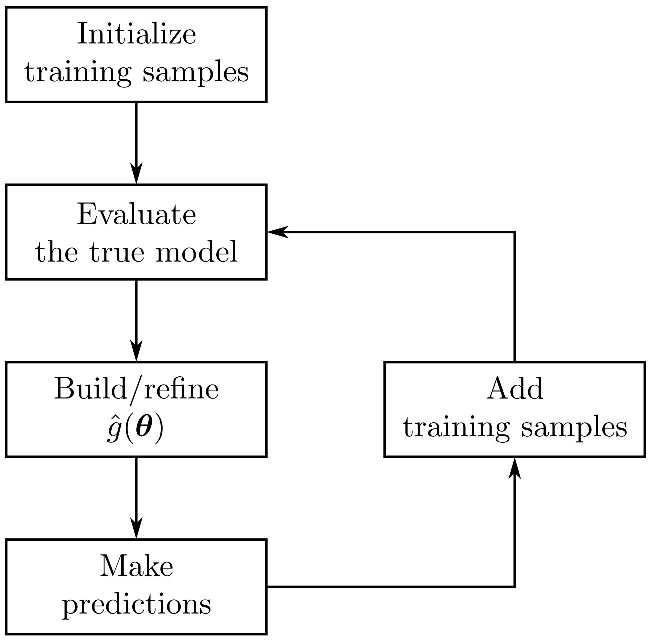

Initialize training samples
Evaluate the true model
Build/refine $\hat{g}(\boldsymbol{\theta})$
Add training samples
Make predictions

Figure 3.5: *Schema of surrogate modeling techniques.*

predictions throughout the entire parameter space. In general, the initial training samples can be drawn according to the distribution of uncertain parameters. Next, these training samples are evaluated employing the true model $g(\boldsymbol{\theta})$. The pairs of training samples and the corresponding true model responses $\mathcal{T} = \{(\boldsymbol{\theta}_t^{(i)}, g(\boldsymbol{\theta}_t^{(i)}))\}$ are referred to as the *training set*. The collected information is then utilized to construct a surrogate model $\hat{g}(\boldsymbol{\theta})$. Various surrogate models can be exploited and three of them will be presented in detail in the subsequent chapters. After that, the responses of the remaining non-training points $\boldsymbol{\theta}_r^{(i)}$ are predicted by the surrogate model. So far, the predictions may not be sufficiently accurate. To improve the prediction accuracy, it is necessary to add more samples to the training set and refine the surrogate model using the updated training set. These new training points are selected based on a certain criterion. This selection process is known as *active learning*. Before the refinement of the surrogate model, the new training points are evaluated with the true model. The enrichment and refinement steps may be repeated several times until the updated surrogate model is sufficiently accurate. During the refinement phase, it is possible that not all the training samples are necessary for updating the surrogate model. In this case, a subset of the training set is chosen for the refinement. This step is called *experimental design*.

$$\begin{tikzcd}
& \boldsymbol{\theta}_t^{(i)} \arrow[r] & g(\boldsymbol{\theta}) \arrow[r] & g(\boldsymbol{\theta}_t^{(i)}) & \\
N \text{ samples} & & & & N \text{ responses} \\
& \boldsymbol{\theta}_r^{(i)} \arrow[r] & \hat{g}(\boldsymbol{\theta}) \arrow[r] & \hat{g}(\boldsymbol{\theta}_r^{(i)}) &
\end{tikzcd}$$

Figure 3.6: *Surrogate-accelerated simulation methods.*

By employing the surrogate modeling techniques, only a fraction of samples $\boldsymbol{\theta}_t^{(i)}$ are evaluated using the true model, whereas others $\boldsymbol{\theta}_r^{(i)}$ are approximately evaluated via the surrogate model, as shown in Figure 3.6. In this way, the number of calls to the computationally demanding true model is reduced, thus accelerating the simulation methods for reliability analysis.

3.4 Summary

This chapter proposed a novel control design optimization method called *reliability-based control optimization* (RBCO). It verifies compliance with probabilistic requirements within an optimization loop. This approach can be used to search for control parameters that satisfy the probabilistic requirements with a formal guarantee. In order to evaluate the failure probability in an accurate and efficient way, the framework combining simulation methods with surrogate modeling techniques was presented.

Chapter 4

Uncertainty Propagation Using Polynomial-Based Global Surrogates

This chapter employs two polynomial-based global surrogate models, namely, *response surface method* (RSM) and *polynomial chaos expansion* (PCE), to achieve the uncertainty propagation task. The principles of RSM and PCE are introduced first. The following section introduces the uncertainty propagation schemes based on RSM and PCE. Statistical characteristics of the system output are evaluated analytically or numerically. Subsequently, the last section combines the global surrogates with SuS to accelerate the process of the rare failure probability estimation. Aiming at constructing the best surrogate at subset levels, an adaptive experimental design strategy is proposed.

4.1 Response Surface Method (RSM)

4.1.1 RSM Model

RSM [43] is an elementary tool to establish the relationship between input variables and model responses. Generally, the relationship, i.e., *response surface*, is approximated by a low-order polynomial as follows:

$$\hat{g}(\boldsymbol{\theta}) = \sum_{k=1}^{p} \alpha_k \psi_k(\boldsymbol{\theta}) = \boldsymbol{\alpha}^\mathsf{T} \boldsymbol{\psi}(\boldsymbol{\theta}), \tag{4.1}$$

where $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_n]^\mathsf{T}$ is a vector of *input parameters*, p is the *number of expansion items*, $\boldsymbol{\psi}(\boldsymbol{\theta}) = [\psi_1(\boldsymbol{\theta}), \ldots, \psi_p(\boldsymbol{\theta})]^\mathsf{T}$ is a vector of *basis functions*, and $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_p]^\mathsf{T}$ is a constant vector that consists of unknown *expansion coefficients*. The basis functions are usually monomials up to a certain *order* denoted by d (≥ 1). For $n = 2$, for example, the basis functions of the second-order expansion ($d = 2$) are given by

$$\boldsymbol{\psi}(\boldsymbol{\theta}) = \left[1, \theta_1, \theta_2, \theta_1^2, \theta_1\theta_2, \theta_2^2\right]^\mathsf{T}. \tag{4.2}$$

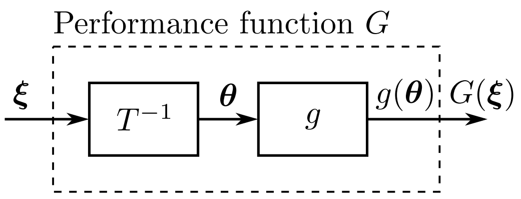


Figure 4.1: *Transformed performance function.*

The number of expansion items is calculated by

$$p = \frac{(n+d)!}{n!d!}. \tag{4.3}$$

4.1.2 Computation of the Coefficients

Given the training set $\mathcal{T} = \{(\boldsymbol{\theta}_t^{(i)}, g(\boldsymbol{\theta}_t^{(i)})) : i = 1, \dots, N_t\}$, the coefficients can be obtained using the *ordinary least-squares* (OLS) regression [136, Ch. 1] that minimizes the residual sum of squares:

$$\begin{aligned} \boldsymbol{\alpha} &= \arg\min \sum_{i=1}^{N_t} \left(\boldsymbol{\alpha}^\mathsf{T} \boldsymbol{\psi}(\boldsymbol{\theta}_t^{(i)}) - g(\boldsymbol{\theta}_t^{(i)})\right)^2 \\ &= \left(\boldsymbol{\Psi}^\mathsf{T}\boldsymbol{\Psi}\right)^{-1} \boldsymbol{\Psi}^\mathsf{T}\boldsymbol{y}, \end{aligned} \tag{4.4}$$

where

$$\boldsymbol{\Psi} = \begin{bmatrix} \boldsymbol{\psi}(\boldsymbol{\theta}_t^{(1)})^\mathsf{T} \\ \boldsymbol{\psi}(\boldsymbol{\theta}_t^{(2)})^\mathsf{T} \\ \vdots \\ \boldsymbol{\psi}(\boldsymbol{\theta}_t^{(N_t)})^\mathsf{T} \end{bmatrix} = \begin{bmatrix} \psi_1(\boldsymbol{\theta}_t^{(1)}) & \psi_2(\boldsymbol{\theta}_t^{(1)}) & \cdots & \psi_p(\boldsymbol{\theta}_t^{(1)}) \\ \psi_1(\boldsymbol{\theta}_t^{(2)}) & \psi_2(\boldsymbol{\theta}_t^{(2)}) & \cdots & \psi_p(\boldsymbol{\theta}_t^{(2)}) \\ \vdots & \vdots & \ddots & \vdots \\ \psi_1(\boldsymbol{\theta}_t^{(N_t)}) & \psi_2(\boldsymbol{\theta}_t^{(N_t)}) & \cdots & \psi_p(\boldsymbol{\theta}_t^{(N_t)}) \end{bmatrix} \tag{4.5}$$

is the *experimental matrix*, and $\boldsymbol{y} = [g(\boldsymbol{\theta}_t^{(1)}), \dots, g(\boldsymbol{\theta}_t^{(N_t)})]^\mathsf{T}$.

4.2 Polynomial Chaos Expansion (PCE)

4.2.1 PCE Model

Similar with RSM, PCE [47, 49, 50] employs a weighted sum of basis functions to approximate the performance function $g(\boldsymbol{\theta})$. However, in PCE, orthogonal polynomials for independent standard random variables are adopted as basis functions. Therefore, it is necessary to transform any given random vector $\boldsymbol{\Theta}$ into an independent standard random vector $\boldsymbol{\Xi}$ via the isoprobabilistic transformation T (see Section 2.1.3). Accordingly, the performance function to be approximated is converted from $g(\boldsymbol{\theta})$ into $G(\boldsymbol{\xi})$, as illustrated in Figure 4.1.

Assume that the response of the performance function has a finite variance:

$$\mathbb{E}\left[G(\boldsymbol{\xi})^2\right] = \int_{\boldsymbol{\Omega}} G(\boldsymbol{\xi})^2 f(\boldsymbol{\xi}) \mathrm{d}\boldsymbol{\xi} < +\infty, \tag{4.6}$$

where $\boldsymbol{\Omega}$ and $f(\boldsymbol{\xi})$ are the support and PDF of $\boldsymbol{\Xi}$, respectively. The response can be expressed as an infinite weighted sum of orthogonal polynomials:

$$G(\boldsymbol{\xi}) = \sum_{k=0}^{\infty} \alpha_k \psi_k(\boldsymbol{\xi}), \tag{4.7}$$

where $\psi_k(\boldsymbol{\xi})$ denote the *multivariate orthogonal polynomial basis functions* and α_k denote the corresponding *expansion coefficients*. In practical problems, this expansion is usually truncated up to a certain degree d:

$$\hat{G}(\boldsymbol{\xi}) = \sum_{k=0}^{p-1} \alpha_k \psi_k(\boldsymbol{\xi}) = \boldsymbol{\alpha}^{\mathsf{T}} \boldsymbol{\psi}(\boldsymbol{\xi}), \tag{4.8}$$

where $\boldsymbol{\psi}(\boldsymbol{\xi}) = [\psi_0(\boldsymbol{\xi}), \psi_1(\boldsymbol{\xi}), \ldots, \psi_{p-1}(\boldsymbol{\xi})]^{\mathsf{T}}$, $\boldsymbol{\alpha} = [\alpha_0, \alpha_1, \ldots, \alpha_{p-1}]^{\mathsf{T}}$, and the number of expansion items p is obtained by

$$p = \frac{(n+d)!}{n!d!}. \tag{4.9}$$

4.2.2 Construction of Orthogonal Polynomial Bases

4.2.2.1 Univariate Orthogonal Polynomials

For a single random variable Ξ whose realization is denoted by ξ and any functions $h_1(\cdot)$ and $h_2(\cdot)$, the *inner product* of $h_1(\xi)$ and $h_2(\xi)$ is defined as

$$\langle h_1(\xi), h_2(\xi) \rangle = \mathbb{E}\left[h_1(\xi) h_2(\xi)\right] = \int_{\Omega} h_1(\xi) h_2(\xi) f(\xi) \mathrm{d}\xi, \tag{4.10}$$

where Ω is the support of Ξ and $f(\xi)$ is the PDF of Ξ. Functions $h_1(\cdot)$ and $h_2(\cdot)$ are *orthogonal* with respect to the probability measure $\mathbb{P}[\mathrm{d}\xi] = f(\xi)\mathrm{d}\xi$ if $\langle h_1(\xi), h_2(\xi) \rangle = 0$. One can build a family of *orthogonal polynomials* $\{\phi_i(\xi), i \in \mathbb{N}_0\}$ that satisfies

$$\langle \phi_i(\xi), \phi_j(\xi) \rangle = \int_{\Omega} \phi_i(\xi) \phi_j(\xi) f(\xi) \mathrm{d}\xi = \gamma_i \delta_{ij}, \tag{4.11}$$

where δ_{ij} is the *Kronecker function*:

$$\delta_{ij} = \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{if } i \neq j, \end{cases} \tag{4.12}$$

and

$$\gamma_i = \langle \phi_i(\xi), \phi_i(\xi) \rangle = \mathbb{E}\left[\phi_i(\xi)^2\right]. \tag{4.13}$$

Monic orthogonal polynomials can be built by the following 3-term recurrence relation:

$$\begin{aligned} \phi_{-1}(\xi) &= 0, \\ \phi_0(\xi) &= 1, \\ \phi_{i+1}(\xi) &= (\xi - a_i)\,\phi_i(\xi) - b_i \phi_{i-1}(\xi), \quad i \in \mathbb{N}_0, \end{aligned} \tag{4.14}$$

where a_i and b_i are constants computed by

$$a_i = \frac{\langle \xi\phi_i(\xi), \phi_i(\xi)\rangle}{\langle \phi_i(\xi), \phi_i(\xi)\rangle}, \tag{4.15}$$

$$b_i = \frac{\langle \phi_i(\xi), \phi_i(\xi)\rangle}{\langle \phi_{i-1}(\xi), \phi_{i-1}(\xi)\rangle}. \tag{4.16}$$

The classical families of orthogonal polynomials [47, 49] are listed in Table 4.1. Herein, $\Gamma(\alpha+1)$ is the *gamma function* and $B(\alpha+1,\beta+1) = \frac{\Gamma(\alpha+1)\Gamma(\beta+1)}{\Gamma(\alpha+\beta+2)}$ is the *beta function*. Take Hermite polynomials [49] for an example, they satisfy the recurrence relation:

$$\begin{aligned} H_{-1}(\xi) &= 0, \\ H_0(\xi) &= 1, \\ H_{i+1}(\xi) &= \xi H_i(\xi) - iH_{i-1}(\xi), \quad i \in \mathbb{N}_0, \end{aligned} \tag{4.17}$$

and

$$\langle H_i(\xi), H_j(\xi)\rangle = \int_{-\infty}^{\infty} H_i(\xi)H_j(\xi)f(\xi)\mathrm{d}\xi = i!\delta_{ij}. \tag{4.18}$$

This indicates

$$\gamma_i = \mathbb{E}\left[H_i(\xi)^2\right] = i!. \tag{4.19}$$

The first few Hermite polynomials are given in Table 4.2 and plotted in Figure 4.2. The details for the remaining types of polynomials in Table 4.1 can be found in Appendix B.

Table 4.1: *Classical families of orthogonal polynomials.*

Type of variable Ξ	Distribution $f(\xi)$	Support Ω	Orthogonal polynomials $\phi_i(\xi)$
Gaussian $\mathcal{N}(0,1)$	$\frac{1}{\sqrt{2\pi}}e^{-\xi^2/2}$	$(-\infty,\infty)$	Hermite $H_i(\xi)$
Uniform $\mathcal{U}(-1,1)$	$\frac{1}{2}$	$[-1,1]$	Legendre $P_i(\xi)$
Gamma $\mathcal{G}(\alpha,1)$	$\frac{\xi^\alpha e^{-\xi}}{\Gamma(\alpha+1)}$	$[0,\infty)$	Laguerre $L_i^\alpha(\xi)$
Beta $\mathcal{B}(\alpha,\beta)$	$\frac{(1-\xi)^\alpha(1+\xi)^\beta}{2^{\alpha+\beta+1}B(\alpha+1,\beta+1)}$	$[-1,1]$	Jacobi $J_i^{\alpha,\beta}(\xi)$

Table 4.2: *First few Hermite polynomials.*

i	$H_i(\xi)$	$\mathbb{E}\left[H_i(\xi)^2\right]$
0	1	1
1	ξ	1
2	ξ^2-1	2
3	$\xi^3-3\xi$	6
4	$\xi^4-6\xi^2+3$	24

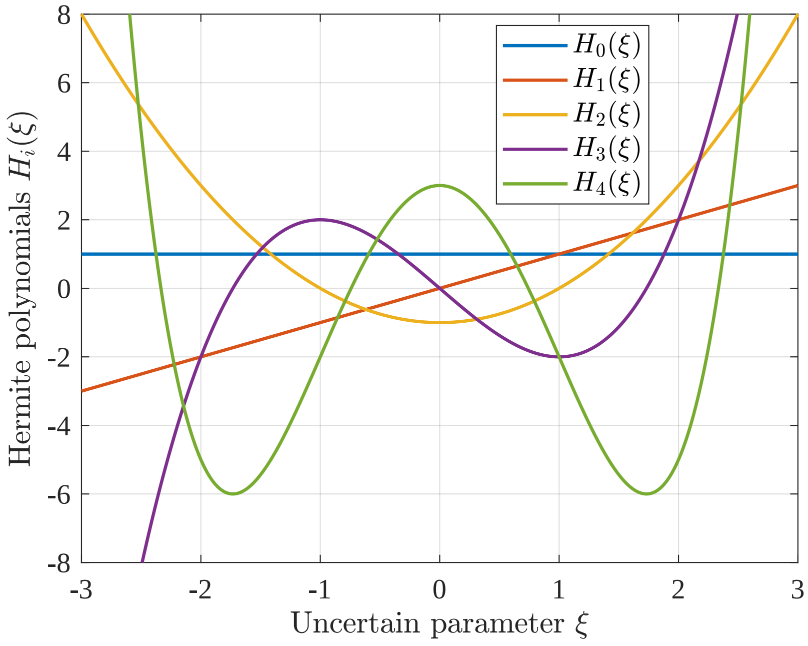


Figure 4.2: *First few Hermite polynomials.*

4.2.2.2 Multivariate Orthogonal Polynomials

For random vectors, the orthogonal polynomials are constructed by the tensor product of univariate orthogonal polynomials:

$$\psi_k(\boldsymbol{\xi}) = \prod_{j=1}^{n} \phi_{m_j^k}(\xi_j), \tag{4.20}$$

where $\boldsymbol{m}^k = [m_1^k, \ldots, m_n^k]$ represents the *multi-index* that contains all possible combinations of univariate orthogonal polynomials. An example of the multi-index and corresponding orthogonal polynomials in $n = 3$ dimensions is shown in Table 4.3, where

$$|\boldsymbol{m}^k| = \sum_{j=1}^{n} m_j^k \tag{4.21}$$

denotes the degree of the polynomial $\psi_k(\boldsymbol{\xi})$. The support of $\boldsymbol{\Xi}$ is the Cartesian product of that of each element:

$$\boldsymbol{\Omega} = \prod_{j=1}^{n} \Omega_j. \tag{4.22}$$

The inner product of $\psi_k(\boldsymbol{\xi})$ and $\psi_r(\boldsymbol{\xi})$ is given by

$$\begin{aligned} \langle \psi_k(\boldsymbol{\xi}), \psi_r(\boldsymbol{\xi}) \rangle = \mathbb{E}\left[\psi_k(\boldsymbol{\xi})\psi_r(\boldsymbol{\xi})\right] &= \int_{\boldsymbol{\Omega}} \psi_k(\boldsymbol{\xi})\psi_r(\boldsymbol{\xi}) f(\boldsymbol{\xi}) \mathrm{d}\boldsymbol{\xi} \\ &= \int_{\Omega_1} \cdots \int_{\Omega_n} \prod_{j=1}^{n} \left[\phi_{m_j^k}(\xi_j)\phi_{m_j^r}(\xi_j) f(\xi_j)\right] \mathrm{d}\xi_n \cdots \mathrm{d}\xi_1 \\ &= \prod_{j=1}^{n} \left[\int_{\Omega_j} \phi_{m_j^k}(\xi_j)\phi_{m_j^r}(\xi_j) f(\xi_j) \mathrm{d}\xi_j\right] \\ &= \tilde{\gamma}_k \delta_{kr}, \end{aligned} \tag{4.23}$$

Table 4.3: *Example of the multi-index and corresponding polynomials in 3 dimensions.*

Single index k	Multi-index $\boldsymbol{m}^k$	$\lvert\boldsymbol{m}^k\rvert$	Orthogonal polynomials $\psi_k(\boldsymbol{\xi})$
0	[0 0 0]	0	$\psi_0(\boldsymbol{\xi}) = \phi_0(\xi_1)\phi_0(\xi_2)\phi_0(\xi_3)$
1	[1 0 0]	1	$\psi_1(\boldsymbol{\xi}) = \phi_1(\xi_1)\phi_0(\xi_2)\phi_0(\xi_3)$
2	[0 1 0]		$\psi_2(\boldsymbol{\xi}) = \phi_0(\xi_1)\phi_1(\xi_2)\phi_0(\xi_3)$
3	[0 0 1]		$\psi_3(\boldsymbol{\xi}) = \phi_0(\xi_1)\phi_0(\xi_2)\phi_1(\xi_3)$
4	[2 0 0]	2	$\psi_4(\boldsymbol{\xi}) = \phi_2(\xi_1)\phi_0(\xi_2)\phi_0(\xi_3)$
5	[1 1 0]		$\psi_5(\boldsymbol{\xi}) = \phi_1(\xi_1)\phi_1(\xi_2)\phi_0(\xi_3)$
6	[1 0 1]		$\psi_6(\boldsymbol{\xi}) = \phi_1(\xi_1)\phi_0(\xi_2)\phi_1(\xi_3)$
7	[0 2 0]		$\psi_7(\boldsymbol{\xi}) = \phi_0(\xi_1)\phi_2(\xi_2)\phi_0(\xi_3)$
8	[0 1 1]		$\psi_8(\boldsymbol{\xi}) = \phi_0(\xi_1)\phi_1(\xi_2)\phi_1(\xi_3)$
9	[0 0 2]		$\psi_9(\boldsymbol{\xi}) = \phi_0(\xi_1)\phi_0(\xi_2)\phi_2(\xi_3)$
10	[3 0 0]	3	$\psi_{10}(\boldsymbol{\xi}) = \phi_3(\xi_1)\phi_0(\xi_2)\phi_0(\xi_3)$
11	[2 1 0]		$\psi_{11}(\boldsymbol{\xi}) = \phi_2(\xi_1)\phi_1(\xi_2)\phi_0(\xi_3)$
...	...		...

where

$$\tilde{\gamma}_k = \mathbb{E}\left[\psi_k(\boldsymbol{\xi})^2\right] = \prod_{j=1}^{n} \gamma_{m_j^k}. \tag{4.24}$$

This proves that the constructed multivariate polynomials are orthogonal to each other.

4.2.3 Truncation Error

Because of the orthogonal polynomial bases, the residual error of the estimation in Equation (4.8) is orthogonal to the chosen bases:

$$\left\langle G(\boldsymbol{\xi}) - \hat{G}(\boldsymbol{\xi}), \psi_r(\boldsymbol{\xi})\right\rangle = \left\langle \sum_{k=p}^{\infty} \alpha_k \psi_k(\boldsymbol{\xi}), \psi_r(\boldsymbol{\xi})\right\rangle = \sum_{k=p}^{\infty} \alpha_k \left\langle \psi_k(\boldsymbol{\xi}), \psi_r(\boldsymbol{\xi})\right\rangle = 0, \quad 0 \le r < p. \tag{4.25}$$

The norm of the residual error is given by

$$\begin{aligned} \|G(\boldsymbol{\xi}) - \hat{G}(\boldsymbol{\xi})\| &= \sqrt{\mathbb{E}\left[\left(G(\boldsymbol{\xi}) - \hat{G}(\boldsymbol{\xi})\right)^2\right]} = \sqrt{\mathbb{E}\left[\sum_{k=p}^{\infty}\sum_{r=p}^{\infty} \alpha_k \alpha_r \psi_k(\boldsymbol{\xi})\psi_r(\boldsymbol{\xi})\right]} \\ &= \sqrt{\sum_{k=p}^{\infty}\sum_{r=p}^{\infty} \alpha_k \alpha_r \, \mathbb{E}\left[\psi_k(\boldsymbol{\xi})\psi_r(\boldsymbol{\xi})\right]} = \sqrt{\sum_{k=p}^{\infty} \alpha_k^2 \tilde{\gamma}_k}, \end{aligned} \tag{4.26}$$

which implies that the norm decreases with the addition of the polynomial basis and

$$\lim_{p\to\infty} \|G(\boldsymbol{\xi}) - \hat{G}(\boldsymbol{\xi})\| = 0. \tag{4.27}$$

This guarantees the convergence of the PCE surrogate model.

4.2.4 Computation of the Coefficients

Once the polynomial bases are selected, the expansion coefficients shall be determined. To achieve that, various techniques have been developed, which can be classified into two categories, i.e., *intrusive* and *non-intrusive* approaches. Using the intrusive methods, such as the *stochastic Galerkin method* [49, Ch. 6], deterministic equations which are coupled with the system model are derived to compute the coefficients. This derivation is nontrivial or even infeasible for complex or black-box problems. Moreover, it is necessary to derive these deterministic equations tailored to different systems. By contrast, the non-intrusive methods simply rely on the repeated run of the computational model, thus being independent with the system model and easy to implement. In this subsection, two popular non-intrusive schemes, i.e., spectral projection and regression, are presented.

4.2.4.1 Spectral Projection

The *spectral projection* method [49, Ch. 7] projects the response in Equation (4.7) against each basis:

$$\begin{aligned}\langle G(\boldsymbol{\xi}), \psi_r(\boldsymbol{\xi})\rangle &= \left\langle \sum_{k=0}^{\infty} \alpha_k \psi_k(\boldsymbol{\xi}), \psi_r(\boldsymbol{\xi}) \right\rangle = \sum_{k=0}^{\infty} \alpha_k \langle \psi_k(\boldsymbol{\xi}), \psi_r(\boldsymbol{\xi})\rangle \\ &= \alpha_r \langle \psi_r(\boldsymbol{\xi}), \psi_r(\boldsymbol{\xi})\rangle = \alpha_r \tilde{\gamma}_r.\end{aligned} \tag{4.28}$$

Hence, the expansion coefficients are obtained by

$$\alpha_r = \frac{1}{\tilde{\gamma}_r} \langle G(\boldsymbol{\xi}), \psi_r(\boldsymbol{\xi})\rangle = \frac{1}{\tilde{\gamma}_r} \mathbb{E}\left[G(\boldsymbol{\xi})\psi_r(\boldsymbol{\xi})\right] = \frac{1}{\tilde{\gamma}_r} \int_{\boldsymbol{\Omega}} G(\boldsymbol{\xi})\psi_r(\boldsymbol{\xi}) f(\boldsymbol{\xi}) \mathrm{d}\boldsymbol{\xi}. \tag{4.29}$$

The estimation of α_r in Equation (4.29) can be achieved by approximating the expectation using the MCS approach. This, however, suffers from inefficiency. An alternative solution is to evaluate the integral in Equation (4.29) via *quadrature* methods.

The 1-dimensional *Gaussian quadrature* rule is given by

$$\int_{\Omega} h(\xi) f(\xi) \mathrm{d}\xi \approx \sum_{i=1}^{q} h(\xi^{(i)}) w^{(i)}, \tag{4.30}$$

where $h(\cdot)$ denotes the function to be integrated against the weight function $f(\xi)$, $\xi^{(i)}$ and $w^{(i)}$ are *quadrature nodes* and the corresponding *weights* with q being the *number of quadrature nodes*. This rule integrates exactly all polynomials of degree no greater than $2q-1$. Given a d-th order univariate PCE, the integrand $h(\xi) = G(\xi)\phi_r(\xi)$, which is the 1-dimensional case of $G(\boldsymbol{\xi})\psi_r(\boldsymbol{\xi})$ in Equation (4.29), is of at most order $2d$, implying that $q = d+1$ integration nodes are required to obtain highly accurate integrals. These nodes can be the zeros of polynomial $\phi_q(\xi)$, and the corresponding weights are calculated by

$$w^{(i)} = \int_{\Omega} f(\xi) \prod_{\substack{1 \le j \le q \\ j \ne i}} \frac{\xi - \xi^{(j)}}{\xi^{(i)} - \xi^{(j)}} \mathrm{d}\xi. \tag{4.31}$$

For the multi-dimensional case, Gaussian quadrature is achieved by a tensor product of the previous 1-dimensional integration rule:

$$\begin{aligned}\int_{\mathbf{\Omega}} h(\boldsymbol{\xi}) f(\boldsymbol{\xi}) \mathrm{d}\boldsymbol{\xi} &\approx \sum_{m_1^i=1}^{q_1} \cdots \sum_{m_n^i=1}^{q_n} h(\xi_1^{(m_1^i)}, \ldots, \xi_n^{(m_n^i)}) w_1^{(m_1^i)} \cdots w_n^{(m_n^i)} \\ &= \sum_{i=1}^{Q} h(\boldsymbol{\xi}^{(i)}) w^{(i)},\end{aligned} \tag{4.32}$$

where $\xi_k^{(m_k^i)}$ are the quadrature nodes in the k-th dimension, and $w_k^{(m_k^i)}$ are the corresponding weights. The multi-index $[m_1^i, \ldots, m_n^i]$ contains all possible combinations of nodes in each dimension. The multivariate quadrature nodes and weights are $\boldsymbol{\xi}^{(i)} = [\xi_1^{(m_1^i)}, \ldots, \xi_n^{(m_n^i)}]^\mathsf{T}$ and $w^{(i)} = w_1^{(m_1^i)} \cdots w_n^{(m_n^i)}$, respectively. This quadrature requires $Q = \prod_{j=1}^{n} q_j$ evaluations of the integrand.

An example applying Gaussian quadrature is given as follows. Consider the integral of a bivariate polynomial:

$$\mathcal{I} = \int_{-1}^{1} \int_{-1}^{1} \frac{1}{4} (\xi_1^2 + 1) \xi_2^4 \mathrm{d}\xi_1 \mathrm{d}\xi_2 = \frac{4}{15}. \tag{4.33}$$

Using Legendre polynomials $P_k(\xi)$ in each component, the weight function $f(\boldsymbol{\xi}) = \frac{1}{4}$ and the integrand $h(\boldsymbol{\xi}) = (\xi_1^2 + 1)\xi_2^4$. To integrate it exactly, 2 nodes are required in ξ_1 whereas 3 nodes in ξ_2. The quadrature nodes and the corresponding weights are listed in Table 4.4. The integral in Equation (4.33) is estimated by

$$\begin{aligned}\mathcal{I} \approx\ & \frac{1}{2}\frac{5}{18} h\left(-\frac{1}{\sqrt{3}}, -\sqrt{\frac{3}{5}}\right) + \frac{1}{2}\frac{4}{9} h\left(-\frac{1}{\sqrt{3}}, 0\right) + \frac{1}{2}\frac{5}{18} h\left(-\frac{1}{\sqrt{3}}, \sqrt{\frac{3}{5}}\right) + \\ & \frac{1}{2}\frac{5}{18} h\left(\frac{1}{\sqrt{3}}, -\sqrt{\frac{3}{5}}\right) + \frac{1}{2}\frac{4}{9} h\left(\frac{1}{\sqrt{3}}, 0\right) + \frac{1}{2}\frac{5}{18} h\left(\frac{1}{\sqrt{3}}, \sqrt{\frac{3}{5}}\right) = \frac{4}{15}.\end{aligned} \tag{4.34}$$

This result is equal to the analytical solution given in Equation (4.33).

Using Gaussian quadrature, Equation (4.29) is estimated by

$$\alpha_r = \frac{1}{\tilde{\gamma}_r} \int_{\mathbf{\Omega}} G(\boldsymbol{\xi}) \psi_r(\boldsymbol{\xi}) f(\boldsymbol{\xi}) \mathrm{d}\boldsymbol{\xi} \approx \frac{1}{\tilde{\gamma}_r} \sum_{i=1}^{Q} G(\boldsymbol{\xi}^{(i)}) \psi_r(\boldsymbol{\xi}^{(i)}) w^{(i)}. \tag{4.35}$$

Table 4.4: *Quadrature nodes and the corresponding weights.*

k	Zeros of $P_k(\xi)$	Weights
1	0	1
2	$-1/\sqrt{3}, 1/\sqrt{3}$	$1/2, 1/2$
3	$-\sqrt{3/5}, 0, \sqrt{3/5}$	$5/18, 4/9, 5/18$

If the order of PCE in each dimension $d_i = d, i = 1, \ldots, n$, then $Q = (d+1)^n$, which means the required number of samples grows exponentially with the increase of the number of uncertain parameters. This limits the tensor product quadrature to low-dimensional problems. Alternatively, one can resort to *Smolyak's sparse quadrature* [49, Ch. 7] to reduce the number of required quadrature points in high dimensions.

4.2.4.2 Regression

The expansion coefficients can be estimated by the *ordinary least-squares* (OLS) regression [50] minimizing the residual sum of squares:

$$\begin{aligned}\boldsymbol{\alpha} &= \arg\min \sum_{i=1}^{N_t} \left(\boldsymbol{\alpha}^\mathsf{T}\boldsymbol{\psi}(\boldsymbol{\xi}_t^{(i)}) - G(\boldsymbol{\xi}_t^{(i)})\right)^2 \\ &= \left(\boldsymbol{\Psi}^\mathsf{T}\boldsymbol{\Psi}\right)^{-1}\boldsymbol{\Psi}^\mathsf{T}\boldsymbol{y},\end{aligned} \tag{4.36}$$

where $\mathcal{T} = \{(\boldsymbol{\xi}_t^{(i)}, g(\boldsymbol{\xi}_t^{(i)})) : i = 1, \ldots, N_t\}$ is a given training set,

$$\boldsymbol{\Psi} = \begin{bmatrix} \boldsymbol{\psi}(\boldsymbol{\xi}_t^{(1)})^\mathsf{T} \\ \boldsymbol{\psi}(\boldsymbol{\xi}_t^{(2)})^\mathsf{T} \\ \vdots \\ \boldsymbol{\psi}(\boldsymbol{\xi}_t^{(N_t)})^\mathsf{T} \end{bmatrix} = \begin{bmatrix} \psi_0(\boldsymbol{\xi}_t^{(1)}) & \psi_1(\boldsymbol{\xi}_t^{(1)}) & \cdots & \psi_{p-1}(\boldsymbol{\xi}_t^{(1)}) \\ \psi_0(\boldsymbol{\xi}_t^{(2)}) & \psi_1(\boldsymbol{\xi}_t^{(2)}) & \cdots & \psi_{p-1}(\boldsymbol{\xi}_t^{(2)}) \\ \vdots & \vdots & \ddots & \vdots \\ \psi_0(\boldsymbol{\xi}_t^{(N_t)}) & \psi_1(\boldsymbol{\xi}_t^{(N_t)}) & \cdots & \psi_{p-1}(\boldsymbol{\xi}_t^{(N_t)}) \end{bmatrix} \tag{4.37}$$

is the experimental matrix, and $\boldsymbol{y} = [G(\boldsymbol{\xi}_t^{(1)}), \ldots, G(\boldsymbol{\xi}_t^{(N_t)})]^\mathsf{T}$. A rule of thumb for designating the *cardinality of training set* is $N_t \in [2p, 3p]$ [50]. The required number of training samples is significantly more affordable than that in the isotropic tensor product quadrature method (i.e., $(d+1)^n$) for high-dimensional applications.

4.2.5 Statistical Information

After the computation of the expansion coefficients, the estimation of statistical information is only a post-processing [50]. Given $\psi_0(\boldsymbol{\xi}) = 1$, the mean of $G(\boldsymbol{\xi})$ is estimated by

$$\begin{aligned}\hat{\mu} &= \mathbb{E}\left[\hat{G}(\boldsymbol{\xi})\right] = \mathbb{E}\left[\hat{G}(\boldsymbol{\xi})\psi_0(\boldsymbol{\xi})\right] = \mathbb{E}\left[\sum_{k=0}^{p-1}\alpha_k\psi_k(\boldsymbol{\xi})\psi_0(\boldsymbol{\xi})\right] \\ &= \sum_{k=0}^{p-1}\alpha_k\,\mathbb{E}\left[\psi_k(\boldsymbol{\xi})\psi_0(\boldsymbol{\xi})\right] = \alpha_0\,\mathbb{E}\left[\psi_0(\boldsymbol{\xi})\psi_0(\boldsymbol{\xi})\right] = \alpha_0.\end{aligned} \tag{4.38}$$

The variance, which is the square of the standard deviation σ, can be approximated as

$$\begin{aligned}\hat{\sigma}^2 &= \mathbb{E}\left[\left(\hat{G}(\boldsymbol{\xi}) - \hat{\mu}\right)^2\right] = \mathbb{E}\left[\left(\sum_{k=1}^{p-1}\alpha_k\psi_k(\boldsymbol{\xi})\right)^2\right] = \mathbb{E}\left[\sum_{k=1}^{p-1}\sum_{r=1}^{p-1}\alpha_k\alpha_r\psi_k(\boldsymbol{\xi})\psi_r(\boldsymbol{\xi})\right] \\ &= \sum_{k=1}^{p-1}\sum_{r=1}^{p-1}\alpha_k\alpha_r\,\mathbb{E}\left[\psi_k(\boldsymbol{\xi})\psi_r(\boldsymbol{\xi})\right] = \sum_{k=1}^{p-1}\alpha_k^2\,\mathbb{E}\left[\psi_k(\boldsymbol{\xi})^2\right] = \sum_{k=1}^{p-1}\alpha_k^2\tilde{\gamma}_k.\end{aligned} \tag{4.39}$$

The skewness and kurtosis of $G(\boldsymbol{\xi})$ can be estimated by

$$\begin{aligned}
\hat{\tau} &= \frac{1}{\hat{\sigma}^3}\,\mathbb{E}\left[\left(\hat{G}(\boldsymbol{\xi})-\hat{\mu}\right)^3\right] = \frac{1}{\hat{\sigma}^3}\,\mathbb{E}\left[\left(\sum_{k=1}^{p-1}\alpha_k\psi_k(\boldsymbol{\xi})\right)^3\right] \\
&= \frac{1}{\hat{\sigma}^3}\sum_{k=1}^{p-1}\sum_{r=1}^{p-1}\sum_{j=1}^{p-1}\alpha_k\alpha_r\alpha_j\,\mathbb{E}\left[\psi_k(\boldsymbol{\xi})\psi_r(\boldsymbol{\xi})\psi_j(\boldsymbol{\xi})\right] \\
&= \frac{1}{\hat{\sigma}^3}\sum_{k=1}^{p-1}\sum_{r=1}^{p-1}\sum_{j=1}^{p-1}\alpha_k\alpha_r\alpha_j\prod_{i=1}^{n} e_{m_i^k m_i^r m_i^j},
\end{aligned}\tag{4.40}$$

and

$$\begin{aligned}
\hat{\kappa} &= \frac{1}{\hat{\sigma}^4}\,\mathbb{E}\left[\left(\hat{G}(\boldsymbol{\xi})-\hat{\mu}\right)^4\right] = \frac{1}{\hat{\sigma}^4}\,\mathbb{E}\left[\left(\sum_{k=1}^{p-1}\alpha_k\psi_k(\boldsymbol{\xi})\right)^4\right] \\
&= \frac{1}{\hat{\sigma}^4}\sum_{k=1}^{p-1}\sum_{r=1}^{p-1}\sum_{j=1}^{p-1}\sum_{l=1}^{p-1}\alpha_k\alpha_r\alpha_j\alpha_l\,\mathbb{E}\left[\psi_k(\boldsymbol{\xi})\psi_r(\boldsymbol{\xi})\psi_j(\boldsymbol{\xi})\psi_l(\boldsymbol{\xi})\right] \\
&= \frac{1}{\hat{\sigma}^4}\sum_{k=1}^{p-1}\sum_{r=1}^{p-1}\sum_{j=1}^{p-1}\sum_{l=1}^{p-1}\alpha_k\alpha_r\alpha_j\alpha_l\prod_{i=1}^{n} e_{m_i^k m_i^r m_i^j m_i^l},
\end{aligned}\tag{4.41}$$

where

$$e_{m_i^k m_i^r m_i^j} = \mathbb{E}\left[\phi_{m_i^k}(\xi_i)\phi_{m_i^r}(\xi_i)\phi_{m_i^j}(\xi_i)\right], \tag{4.42}$$

and

$$e_{m_i^k m_i^r m_i^j m_i^l} = \mathbb{E}\left[\phi_{m_i^k}(\xi_i)\phi_{m_i^r}(\xi_i)\phi_{m_i^j}(\xi_i)\phi_{m_i^l}(\xi_i)\right]. \tag{4.43}$$

Once the types of orthogonal polynomials are determined, $e_{m_i^k m_i^r m_i^j}$ and $e_{m_i^k m_i^r m_i^j m_i^l}$ are constant and sometimes can be evaluated analytically. For Hermite polynomials [49, p. 70], for example,

$$e_{m_i^k m_i^r m_i^j} = \begin{cases} \frac{m_i^k!\,m_i^r!\,m_i^j!}{(s-m_i^k)!(s-m_i^r)!(s-m_i^j)!}, & \text{if } 2s = m_i^k + m_i^r + m_i^j \text{ is even and } s \geq m_i^k, m_i^r, m_i^j, \\ 0, & \text{otherwise.} \end{cases} \tag{4.44}$$

4.3 Uncertainty Propagation Using RSM and PCE

4.3.1 Uncertainty Propagation Framework

The uncertainty propagation schemes based on RSM and PCE are illustrated in Figures 4.3 and 4.4, respectively. Using RSM, a surrogate model $\hat{g}(\boldsymbol{\theta})$ is built regardless of the PDF of $\boldsymbol{\theta}$. The inputs of RSM are thereby deterministic variables. With this cheap-to-evaluate model, MCS is carried out to obtain a large number of simulations, and thus to estimate the statistical characteristics of model responses, including moments, failure probabilities,

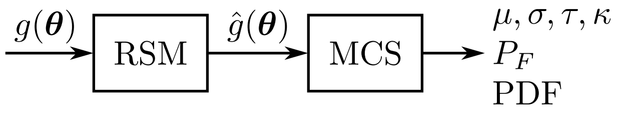
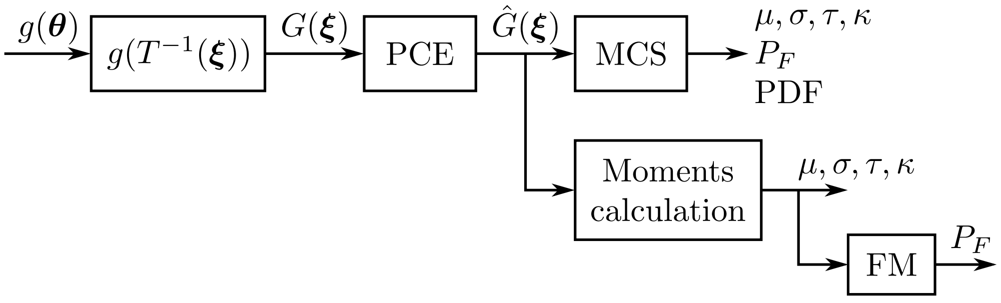


Figure 4.3: *Uncertainty propagation using RSM.*



$g(\boldsymbol{\theta})$
$g(T^{-1}(\boldsymbol{\xi}))$
$G(\boldsymbol{\xi})$
PCE
$\hat{G}(\boldsymbol{\xi})$
MCS
$\mu, \sigma, \tau, \kappa$
P_F
PDF
Moments calculation
$\mu, \sigma, \tau, \kappa$
FM
P_F

Figure 4.4: *Uncertainty propagation using PCE.*

and PDFs. Given MCS samples $\{\boldsymbol{\theta}^{(i)} : i = 1, \ldots, N\}$, the moments up to the fourth order are approximated by

$$\hat{\mu} = \mathbb{E}\left[\hat{g}(\boldsymbol{\theta})\right] \approx \frac{1}{N}\sum_{i=1}^{N} \hat{g}(\boldsymbol{\theta}^{(i)}), \tag{4.45}$$

$$\hat{\sigma}^2 = \mathbb{E}\left[(\hat{g}(\boldsymbol{\theta}) - \hat{\mu})^2\right] \approx \frac{1}{N-1}\sum_{i=1}^{N} \left(\hat{g}(\boldsymbol{\theta}^{(i)}) - \hat{\mu}\right)^2, \tag{4.46}$$

$$\hat{\tau} = \frac{1}{\hat{\sigma}^3}\mathbb{E}\left[(\hat{g}(\boldsymbol{\theta}) - \hat{\mu})^3\right] \approx \frac{1}{\hat{\sigma}^3}\frac{1}{N}\sum_{i=1}^{N} \left(\hat{g}(\boldsymbol{\theta}^{(i)}) - \hat{\mu}\right)^3, \tag{4.47}$$

$$\hat{\kappa} = \frac{1}{\hat{\sigma}^4}\mathbb{E}\left[(\hat{g}(\boldsymbol{\theta}) - \hat{\mu})^4\right] \approx \frac{1}{\hat{\sigma}^4}\frac{1}{N}\sum_{i=1}^{N} \left(\hat{g}(\boldsymbol{\theta}^{(i)}) - \hat{\mu}\right)^4. \tag{4.48}$$

The failure probability is estimated by

$$\hat{P}_F = \mathbb{E}\left[I(\hat{g}(\boldsymbol{\theta}) \le 0)\right] \approx \frac{1}{N}\sum_{i=1}^{N} I\left(\hat{g}(\boldsymbol{\theta}^{(i)}) \le 0\right), \tag{4.49}$$

where $I(\cdot)$ is an indicator function. The PDF of the model response $z = g(\boldsymbol{\theta})$ is approximated using the *Kernel density estimation* [137, Ch. 6]:

$$\hat{f}(z) = \frac{1}{Nh}\sum_{i=1}^{N} K\left(\frac{z - \hat{g}(\boldsymbol{\theta}^{(i)})}{h}\right), \tag{4.50}$$

where h is a smoothing parameter called the *bandwidth*, and $K(\cdot)$ is the *Gaussian Kernel function*:

$$K(t) = \frac{1}{\sqrt{2\pi}}e^{-t^2/2}. \tag{4.51}$$

By contrast, PCE considers the distribution of the uncertain inputs, which means the inputs of PCE are random variables. Due to this fact, on the one hand, the isoprobabilistic transformation may be required before the implementation of PCE. On the other hand,

Table 4.5: *Comparisons between RSM and PCE.*

Method	Inputs	Isoprobabilistic transformation	Expansion bases	Convergence of the approximation
RSM	Deterministic variables	Not required	Monomials	Not guaranteed
PCE	Random variables	Required	Orthogonal polynomials	Guaranteed

the moments of the output can be computed directly once the expansion is determined (see Section 4.2.5). Given the moment information, the failure probability based on the *fourth-moment* (FM) method [138] is approximated as

$$\begin{aligned}\beta_{\mathrm{SM}} &= \frac{\mu}{\sigma},\\ \beta_{\mathrm{FM}} &= \frac{3(\kappa-1)\beta_{\mathrm{SM}} + \tau(\beta_{\mathrm{SM}}^2 - 1)}{\sqrt{(9\kappa - 5\tau^2 - 9)(\kappa - 1)}},\\ \hat{P}_F &\approx \Phi(-\beta_{\mathrm{FM}}),\end{aligned} \tag{4.52}$$

where β_{SM} and β_{FM} are the *reliability indexes* based on the second-moment method and the fourth-moment method, respectively, and $\Phi(\cdot)$ is the CDF of the standard normal distribution. Note that the fourth-moment method is more appropriate for unimodal bell-shaped distributions. Alternatively, like in RSM, the statistical characteristics can be estimated by MCS given the constructed surrogate model.

Besides the above-mentioned differences between RSM and PCE, another is that PCE guarantees the convergence of the approximation (see Section 4.2.3), but there is no such guarantee in RSM. The comparisons between these two surrogate modeling techniques are summarized in Table 4.5. Compared with RSM, PCE enjoys the advantage of easy access to statistical moments and better convergence property, but the isoprobabilistic transformation required by PCE may increase the complexity of the transformed model $G(\boldsymbol{\xi})$ to be approximated.

4.3.2 Adaptive RSM/PCE

In practical problems, one may not know where to truncate the expansions in RSM or PCE. This subsection presents how to choose the optimal expansion bases in the regression approach for both surrogate modeling techniques.

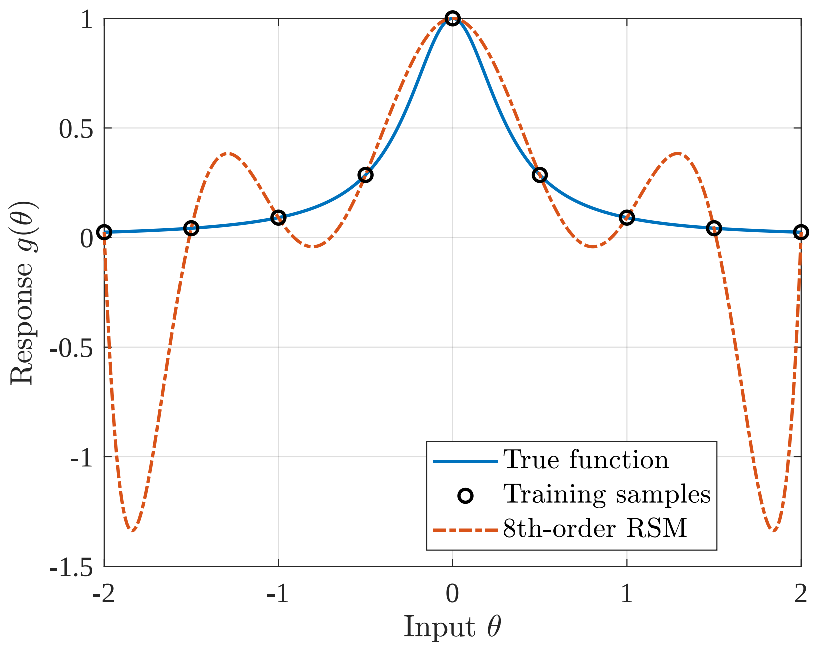


Figure 4.5: *Example of the overfitting phenomenon.*

4.3.2.1 Error Estimation

The quality of the constructed surrogate model can be measured by the mean square error of the residual, which is also called the *empirical error*:

$$\varepsilon_{\text{emp}} = \frac{1}{N_t} \sum_{i=1}^{N_t} \left(g(\boldsymbol{\theta}_t^{(i)}) - \hat{g}(\boldsymbol{\theta}_t^{(i)}) \right)^2 , \tag{4.53}$$

where $\mathcal{T} = \{(\boldsymbol{\theta}_t^{(i)}, g(\boldsymbol{\theta}_t^{(i)})) : i = 1, \ldots, N_t\}$ is a given training set. The *normalized empirical error* is defined as

$$\tilde{\varepsilon}_{\text{emp}} = \frac{\varepsilon_{\text{emp}}}{\text{Var}\left[g(\boldsymbol{\theta}_t^{(i)}) \right]} . \tag{4.54}$$

The error is reduced with the increase of the expansion order until the surrogate model fits the training set perfectly, i.e., ε_{emp} or $\tilde{\varepsilon}_{\text{emp}}$ is almost zero. However, with sufficiently high order, the risk involved is that the approximation of the training samples can be extremely good but very bad elsewhere. It means the surrogate model can be quite different with that built by another training set. This situation is known as *overfitting*. An illustrative example is given in Figure 4.5, where the true function is expressed as

$$g(\theta) = \frac{1}{1 + 10\theta^2} . \tag{4.55}$$

The 8th-order approximation fits the training points very well with $\tilde{\varepsilon}_{\text{emp}} = 7.5 \times 10^{-24}$. However, this approximation leads to erroneous predictions. Therefore, the surrogate modeling error is usually underestimated by the empirical error.

The overfitting phenomenon can be avoided by cross validation, which separates the known data into a training set and a validation set. The validation set is only used to assess the prediction ability of the trained model. Specifically, *leave-one-out* (LOO) *cross*

Algorithm 9 Adaptive RSM/PCE algorithm

```
Input: Training set $\mathcal{T} = \{(\boldsymbol{\theta}_t^{(i)}, g(\boldsymbol{\theta}_t^{(i)})) : i = 1, \ldots, N_t\}$; The possible orders of the
    expansion $d_{\min} : d_{\max}$.
Output: Surrogate model $\hat{g}(\boldsymbol{\theta})$.
1: for $d = d_{\min} : d_{\max}$ do
2:     Generate polynomial bases $\boldsymbol{\psi}$;
3:     Calculate the experimental matrix $\boldsymbol{\Psi}$;
4:     Solve the ordinary least-squares problem and obtain the expansion coefficients $\boldsymbol{\alpha}$;
5:     Compute $\tilde{\varepsilon}_{\text{LOO}}(d)$ according to Equations (4.57) and (4.58);
6: end for
7: Identify the optimal expansion order $d^* = \arg\min \tilde{\varepsilon}_{\text{LOO}}(d)$;
8: Obtain the $d^*$-th order expansion $\hat{g}(\boldsymbol{\theta})$;
9: return $\hat{g}(\boldsymbol{\theta})$.
```

validation [50] employs a single validation sample $\boldsymbol{\theta}_t^{(i)}$ to test the surrogate model $\hat{g}^{(-i)}(\boldsymbol{\theta})$ built from the remaining $N_t - 1$ points. The predicted residual error at $\boldsymbol{\theta}^{(i)}$ is given by

$$\varepsilon_i = g(\boldsymbol{\theta}_t^{(i)}) - \hat{g}^{(-i)}(\boldsymbol{\theta}_t^{(i)}) = \frac{g(\boldsymbol{\theta}_t^{(i)}) - \hat{g}(\boldsymbol{\theta}_t^{(i)})}{1 - h_i}, \tag{4.56}$$

where $\hat{g}(\boldsymbol{\theta})$ is the surrogate model constructed from the full training set, and h_i is the i-th diagonal term of matrix $\boldsymbol{\Psi}(\boldsymbol{\Psi}^\mathsf{T}\boldsymbol{\Psi})^{-1}\boldsymbol{\Psi}^\mathsf{T}$. The *LOO cross validation error* is defined as

$$\varepsilon_{\text{LOO}} = \frac{1}{N_t}\sum_{i=1}^{N_t} \varepsilon_i^2 = \frac{1}{N_t}\sum_{i=1}^{N_t}\left(\frac{g(\boldsymbol{\theta}_t^{(i)}) - \hat{g}(\boldsymbol{\theta}_t^{(i)})}{1 - h_i}\right)^2. \tag{4.57}$$

Similarly, the *normalized LOO error* is obtained by

$$\tilde{\varepsilon}_{\text{LOO}} = \frac{\varepsilon_{\text{LOO}}}{\text{Var}\left[g(\boldsymbol{\theta}_t^{(i)})\right]}. \tag{4.58}$$

In the previous example where $\tilde{\varepsilon}_{\text{emp}} = 7.5 \times 10^{-24}$, the corresponding LOO error is $\tilde{\varepsilon}_{\text{LOO}} = 9.5$, which better reflects the generalization ability of the surrogate model.

4.3.2.2 Adaptive RSM/PCE Algorithm

A polynomial-based global surrogate model with a low-order expansion may not be able to capture the high-order features of the true model, whereas that with a high-order expansion may suffer from the overfitting situation. To avoid such problems, an adaptive algorithm that selects the optimal expansion order by minimizing the LOO error is presented in Algorithm 9.

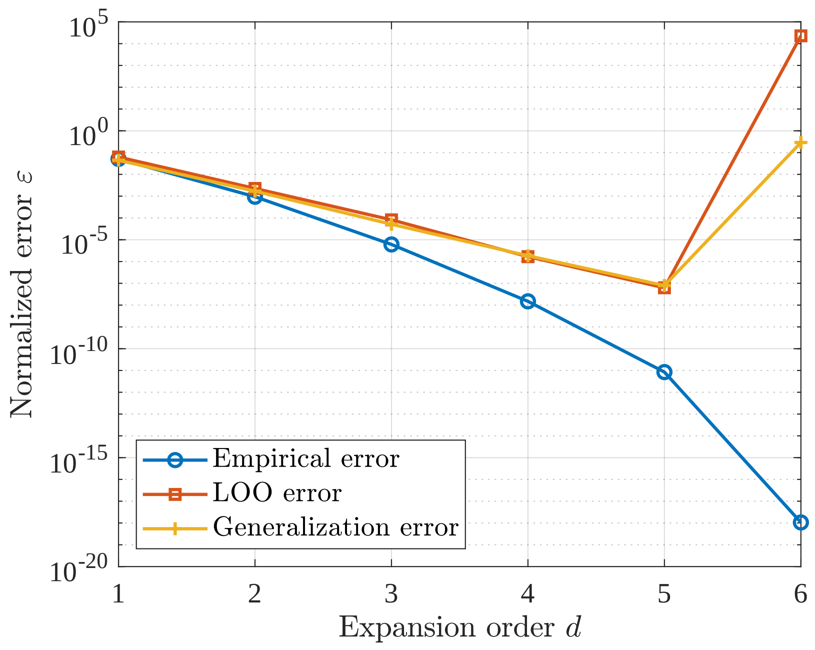


Figure 4.6: *PCE surrogate modeling errors.*

4.3.3 Illustrative Examples

4.3.3.1 Uncertainty Propagation

Consider a 3-dimensional function as follows:

$$g(\boldsymbol{\theta}) = a - \left(e^{0.3\theta_1+1} + e^{0.3\theta_2+1} + e^{0.3\theta_3+1}\right), \tag{4.59}$$

where $a = 10$ and $\theta_i, i = 1, 2, 3$, are independent standard normal random variables, i.e., $\theta_i \sim \mathcal{N}(0, 1)$. Both RSM and PCE are applied to approximate this function and then estimate the statistical properties of the system response. Since the function inputs are already independent standard random variables, there is no need to implement the isoprobabilistic transformation for PCE.

In this example, the adaptive approach in Algorithm 9 is employed with $N_t = 80$ training samples and possible expansion orders $d_{\text{min}} : d_{\text{max}} = 1 : 6$. The approximation errors of the surrogate models constructed by PCE of different orders are shown in Figure 4.6. The empirical error and LOO error are calculated by Equations (4.54) and (4.58), respectively. The normalized generalization error $\tilde{\varepsilon}_{\text{gen}}$ is estimated by MCS using a large test set with $N = 10^5$ samples:

$$\varepsilon_{\text{gen}} = \frac{1}{N}\sum_{i=1}^{N}\left(g(\boldsymbol{\theta}^{(i)}) - \hat{g}(\boldsymbol{\theta}^{(i)})\right)^2, \tag{4.60}$$

$$\tilde{\varepsilon}_{\text{gen}} = \frac{\varepsilon_{\text{gen}}}{\text{Var}\left[g(\boldsymbol{\theta}^{(i)})\right]}. \tag{4.61}$$

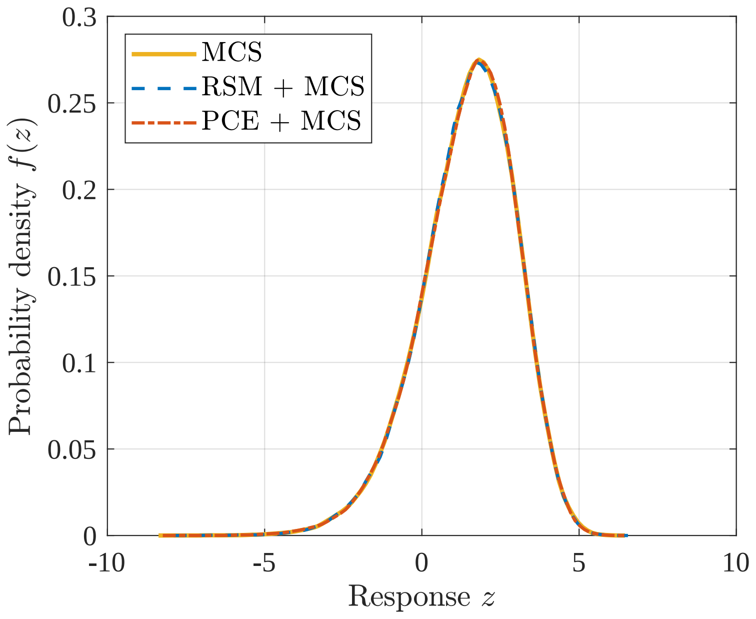


Figure 4.7: *PDF estimates of the function output.*

This figure illustrates that the empirical error underestimates the generalization error, especially when excessively high-order surrogate models are adopted. In contrast, the LOO error is able to detect the overfitting situation, thus it is more reasonable to assess the generalization ability of the surrogate model. Note that similar results can be obtained using adaptive RSM and are omitted here. The overfitting problem here arises from the lack of training samples for a 6th-order expansion. In practice, this can be easily avoided by selecting sufficient training points for a desired expansion order. But the overfitting may be caused by choosing a excessively high expansion order.

Table 4.6 and Figure 4.7 show the uncertainty propagation results exploiting different methods. To compare the performance of different strategies, the results of MCS with $N = 10^5$ samples are regarded as reference values. The uncertainty propagation schemes in Section 4.3.1, i.e., MCS based on the RSM surrogate model (RSM + MCS), MCS based on the PCE surrogate model (PCE + MCS), and the analytical approach based on the PCE surrogate model (PCE + FM), are implemented with only $N_{\text{call}} = 80$ true model

Table 4.6: *Moment and failure probability estimation results of the function output.*

Method	μ	σ	τ	κ	P_F	N_{call}
MCS	1.4698	1.5140	-0.5439	3.5360	0.1605	10^5
RSM + MCS	1.4698	1.5098	-0.5461	3.5472	0.1590	80
PCE + MCS	1.4698	1.5150	-0.5572	3.5415	0.1607	80
PCE + FM	1.4698	1.5113	-0.5478	3.5467	0.1562	80

evaluations. The results obtained by the surrogate-assisted methods are almost equal to those by the reference MCS. This demonstrates that the two global surrogate modeling techniques are accurate and efficient, providing the nonlinearity of the true model can be successfully captured by the selected basis functions.

4.3.3.2 The Influence of Isoprobabilistic Transformation

Consider the following 3-dimensional linear function:

$$g(\boldsymbol{\theta}) = a - (\theta_1 + \theta_2 + \theta_3)\,, \tag{4.62}$$

where $a = 10$ and $\theta_i, i = 1, 2, 3$, are i.i.d. lognormal random variables: $\theta_i = e^{\mu+\sigma\xi_i}$, in which $\mu = 1$, $\sigma = 0.3$, and ξ_i are i.i.d. standard normal random variables: $\xi_i \sim \mathcal{N}(0, 1)$. Both adaptive RSM and adaptive PCE are implemented to approximate the true model. RSM is able to deal with this linear function directly, whereas PCE needs to transform the lognormal random variables into the standard normal space via the isoprobabilistic transformation

$$\begin{aligned} T : \xi_i &= \frac{\log\theta_i - \mu}{\sigma}, \\ T^{-1} : \theta_i &= e^{\mu+\sigma\xi_i}. \end{aligned} \tag{4.63}$$

Hence, the transformed model $G(\boldsymbol{\xi})$ is given by

$$G(\boldsymbol{\xi}) = g(T^{-1}(\boldsymbol{\xi})) = a - \left(e^{0.3\xi_1+1} + e^{0.3\xi_2+1} + e^{0.3\xi_3+1}\right), \tag{4.64}$$

which is the same as the function in the previous example.

The results of these two methods are listed in Table 4.7. The adaptive RSM recognizes the linear approximation (expansion order $d = 1$) with $p = 4$ expansion items as the best surrogate model. With only $N_t = 2p = 8$ training samples, this linear surrogate model achieves a quite high accuracy level. In comparison, the adaptive PCE prefers to use a higher-order expansion since the transformed model is nonlinear. Given 80 training samples, the 5th-order polynomial expansion with 56 expansion items is the optimal choice. It is suggested in this example that the isoprobabilistic transformation may increase the complexity of the transformed model. This usually happens when the original model shows weak nonlinearity but the distribution of uncertain parameters is uncommon or complicated. In such cases, RSM can be a better choice than PCE.

Table 4.7: *RSM and PCE surrogate modeling results.*

Methods	d	p	N_t	$\tilde{\varepsilon}_{\mathrm{LOO}}$
RSM	1	4	8	1.2×10^{-28}
PCE	5	56	80	8.4×10^{-12}

4.4 Rare Event Estimation Based on PCE and RSM

The uncertainty propagation schemes in Section 4.3.1 estimate failure probabilities by performing MCS based on the PCE or RSM surrogate model. However, PCE and RSM are global surrogate modeling techniques that focus on capturing the major trend of the true model throughout the entire parameter space. Therefore, these schemes may bring in large errors when estimating rare-event probabilities. SuS specializes in efficiently evaluating rare events, but it may still require at least thousands of true model evaluations. Aiming at estimating small failure probabilities in an accurate and efficient way, this section proposes a new method called *polynomial surrogate-based SuS* (PS-SuS).

4.4.1 Polynomial Surrogate-Based SuS

As presented in Section 2.5, SuS explores the rare failure domain level by level. To enhance its efficiency by reducing the number of true model evaluations, a surrogate model is constructed to replace most of the calls to the true model with those to the surrogate model. The surrogate model is refined progressively as the SuS samples approach the rare failure domain. This general idea is visualized in Figure 4.8.

At the initial level of SuS, samples $\boldsymbol{\theta}_0^{(i)}$ are first generated by MCS. A fraction of these points are selected as training samples $\boldsymbol{\theta}_{0,t}^{(i)}$, whereas the remaining non-training samples are denoted by $\boldsymbol{\theta}_{0,r}^{(i)}$. Subsequently, the training samples are evaluated by the true model $g(\boldsymbol{\theta})$, and the training set is assigned as $\mathcal{T} = \{(\boldsymbol{\theta}_{0,t}^{(i)}, g(\boldsymbol{\theta}_{0,t}^{(i)}))\}$. A PCE or RSM surrogate model $\hat{g}(\boldsymbol{\theta})$ is constructed employing this training set, and then used to estimate the model outputs of the non-training points $\boldsymbol{\theta}_{0,r}^{(i)}$. After that, the intermediate threshold b_1 is set as the p_0-percentile of the function values $\{g(\boldsymbol{\theta}_{0,t}^{(i)})\} \cup \{\hat{g}(\boldsymbol{\theta}_{0,r}^{(i)})\}$, and the intermediate failure domain $\mathcal{F}_1$ is thereby defined. The samples $\boldsymbol{\theta}_0^{(i)} \in \mathcal{F}_1$ are chosen as seeds for the next subset level, evaluated using the true model, and added to the training set $\mathcal{T}$. The surrogate model is refined as soon as the training set is updated. At the j-th subset level, samples $\boldsymbol{\theta}_j^{(i)}$ are generated by the MCMC sampling strategy, in which candidates $\tilde{\boldsymbol{\theta}}_j^{(i)}$ are proposed and then accepted or rejected according to whether or not they lie in the intermediate failure domain $\mathcal{F}_j$. The acceptance or rejection of the candidates is carried out exploiting the latest surrogate model. $\boldsymbol{\theta}_j^{(i)}$ represent the resulting samples of this step. Afterwards, the intermediate threshold b_{j+1} and the intermediate failure domain $\mathcal{F}_{j+1}$ are determined. The samples $\boldsymbol{\theta}_j^{(i)}$ that fall in $\mathcal{F}_{j+1}$ are chosen as seeds for the next level and added to $\mathcal{T}$ to refine the surrogate model. The previous steps are repeated until there are enough samples in the target failure domain.

In addition to the initial training samples $\boldsymbol{\theta}_{0,t}^{(i)}$, the points $\boldsymbol{\theta}_j^{(i)} \in \mathcal{F}_{j+1}$ at the j-th level ($j = 0, 1, \ldots, m-1$) are added to the training set. The new training samples spread over the following subset to be explored, thus favors the refinement of the surrogate model in

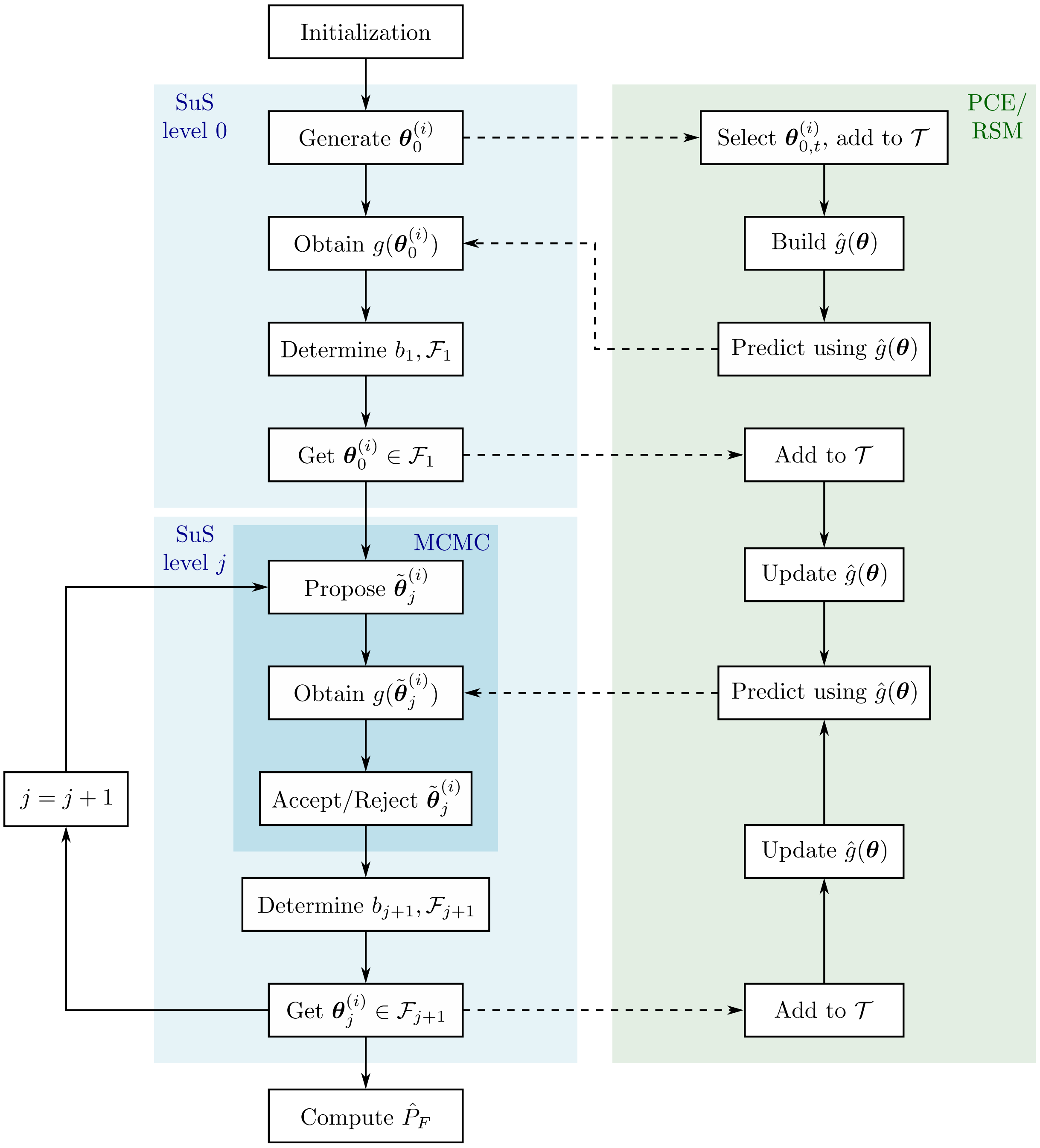

Initialization
SuS level 0
Generate $\boldsymbol{\theta}_0^{(i)}$
Obtain $g(\boldsymbol{\theta}_0^{(i)})$
Determine $b_1, \mathcal{F}_1$
Get $\boldsymbol{\theta}_0^{(i)} \in \mathcal{F}_1$
PCE/ RSM
Select $\boldsymbol{\theta}_{0,t}^{(i)}$, add to $\mathcal{T}$
Build $\hat{g}(\boldsymbol{\theta})$
Predict using $\hat{g}(\boldsymbol{\theta})$
Add to $\mathcal{T}$
SuS level j
MCMC
Propose $\tilde{\boldsymbol{\theta}}_j^{(i)}$
Obtain $g(\tilde{\boldsymbol{\theta}}_j^{(i)})$
Accept/Reject $\tilde{\boldsymbol{\theta}}_j^{(i)}$
Determine $b_{j+1}, \mathcal{F}_{j+1}$
Get $\boldsymbol{\theta}_j^{(i)} \in \mathcal{F}_{j+1}$
$j = j + 1$
Update $\hat{g}(\boldsymbol{\theta})$
Predict using $\hat{g}(\boldsymbol{\theta})$
Update $\hat{g}(\boldsymbol{\theta})$
Add to $\mathcal{T}$
Compute $\hat{P}_F$

Figure 4.8: *Polynomial surrogate-based SuS.*

the domain of interest. The conventional SuS needs to evaluate the true model whenever new samples are generated. If the component-wise MH algorithm (see Algorithm 6) is applied, the *number of calls to the true model* for SuS with m levels is given by

$$N_{\text{call}} \approx mN, \tag{4.65}$$

where N is the number of generated samples at each subset level. Due to the rejection of candidates with a given probability, N_{call} is smaller than mN but gets close to mN as the dimension increases. In comparison, the PS-SuS method only calls the true model for training samples, which means the number of true model evaluations equals the size of the training set. We denote $\tilde{p}_0 N$ as the *number of initial training samples* $\boldsymbol{\theta}_{0,t}^{(i)}$. The number of calls to the true model for PS-SuS is given by

$$N_{\text{call}} \approx \tilde{p}_0 N + (m-1) p_0 N = \left(\frac{\tilde{p}_0 - p_0}{m} + p_0 \right) mN. \tag{4.66}$$

Similarly, because of the rejection step in the component-wise MH algorithm, not all the $p_0 N$ seeds at each level need to be assessed using the true model. Compared with SuS, PS-SuS requires only a small proportion of true model evaluations, thus gaining much higher efficiency.

The initial level of SuS is the direct MCS, thus users can choose whether to utilize PCE or RSM in accordance with the distribution of uncertain parameters and the nonlinearity of the true model. At the subsequent subset levels ($j = 1, \ldots, m-1$), the samples follow the conditional distribution

$$f(\boldsymbol{\theta} | \mathcal{F}_j) = \frac{f(\boldsymbol{\theta}) I_{\mathcal{F}_j}(\boldsymbol{\theta})}{\mathbb{P}[\mathcal{F}_j]}, \tag{4.67}$$

which is not known, thus making it nontrivial to implement the isoprobabilistic transformation for PCE. As a consequence, it is recommended to employ RSM, which is independent with the PDF of the uncertain parameters.

4.4.2 Adaptive Experimental Design Strategy

The fundamental idea of the PS-SuS approach has been presented in the previous subsection. This subsection introduces an adaptive experimental design strategy that chooses the most valuable training samples from the full training set to refine the surrogate model.

At the j-th level ($j = 0, 1, \ldots, m-2$) of SuS, the training set is enriched with seeds for the next level $\boldsymbol{\theta}_j^{(i)} \in \mathcal{F}_{j+1}$. One can use the full training set to update the global surrogate model. However, a high percentage of existing training samples locate outside the domain of interest $\mathcal{F}_{j+1}$. This may result in capturing the main trend throughout the entire space instead of the unique features in the interested domain $\mathcal{F}_{j+1}$, thus leading to large prediction errors in this intermediate failure domain. Alternatively, one can use only training samples in $\mathcal{F}_{j+1}$ to refine the surrogate model. But the refined surrogate lacks confidence in predicting model responses outside this domain, or even generates erroneous predictions. An example is given in Figure 4.9a, where the surrogate model built by the last three training samples makes completely wrong predictions at the first two training samples. To find a tradeoff that the obtained surrogate model can not only mimic the model behaviors locally in $\mathcal{F}_{j+1}$ but also detect the global trend, an adaptive strategy is proposed in the following.

It is desirable that the updated surrogate model at the end of the j-th level is able to accurately predict the responses of samples in $\mathcal{F}_{j+1} = \{\boldsymbol{\theta} \in \boldsymbol{\Omega} : g(\boldsymbol{\theta}) \leq b_{j+1}\}$ and distinguish whether or not a given sample should lie in $\mathcal{F}_{j+1}$. With this target, the full

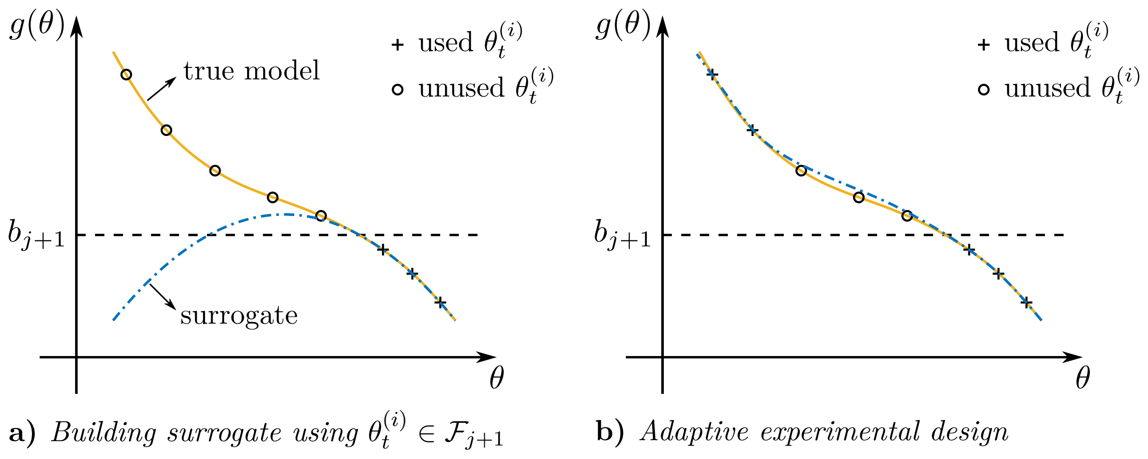


Figure 4.9: *Adaptive experimental design for PS-SuS.*

Algorithm 10 Adaptive experimental design strategy for PS-SuS

```
Input: Training set $\mathcal{T} = \{(\boldsymbol{\theta}_t^{(i)}, g(\boldsymbol{\theta}_t^{(i)})) : i = 1, \dots, N_t\}$; Intermediate threshold $b_{j+1}$.
Output: Surrogate model $\hat{g}(\boldsymbol{\theta})$.
 1: Divide $\mathcal{T}$ into a smaller training set $\widetilde{\mathcal{T}} = \{(\boldsymbol{\theta}_t^{(i)}, g(\boldsymbol{\theta}_t^{(i)})) : g(\boldsymbol{\theta}_t^{(i)}) \leq b_{j+1}\}$ and a
    validation set $\mathcal{V} = \{(\boldsymbol{\theta}_t^{(i)}, g(\boldsymbol{\theta}_t^{(i)})) : g(\boldsymbol{\theta}_t^{(i)}) > b_{j+1}\}$;
 2: Set the possible expansion orders $d = d_{\min} : d_{\max}$;
 3: Construct an RSM surrogate model $\hat{g}(\boldsymbol{\theta})$ using $\widetilde{\mathcal{T}}$ (Algorithm 9);
 4: Estimate the function outputs of the samples in $\mathcal{V}$ employing $\hat{g}(\boldsymbol{\theta})$ and denote the
    number of samples satisfying $\hat{g}(\boldsymbol{\theta}_t^{(i)}) \leq b_{j+1}$ by $n_f$;
 5: while $n_f > 0$ do
 6:     Move the $n_f$ misclassified samples from $\mathcal{V}$ to $\widetilde{\mathcal{T}}$;
 7:     Set the possible expansion orders $d = d_{\min} : d_{\max}$;
 8:     Build an RSM surrogate model $\hat{g}(\boldsymbol{\theta})$ using $\widetilde{\mathcal{T}}$ (Algorithm 9);
 9:     Estimate the function outputs of the samples in $\mathcal{V}$ exploiting $\hat{g}(\boldsymbol{\theta})$ and denote the
        number of samples satisfying $\hat{g}(\boldsymbol{\theta}_t^{(i)}) \leq b_{j+1}$ by $n_f$;
10: end while
11: return $\hat{g}(\boldsymbol{\theta})$.
```

training set $\mathcal{T}$ is first divided into two sets, namely, a smaller training set $\widetilde{\mathcal{T}}$ including all the training samples falling in $\mathcal{F}_{j+1}$ and a *validation set* $\mathcal{V} = \mathcal{T} \setminus \widetilde{\mathcal{T}}$. A global surrogate model $\hat{g}(\boldsymbol{\theta})$ is constructed using $\widetilde{\mathcal{T}}$. The samples in $\mathcal{V}$ are then adopted to assess the generalization ability of $\hat{g}(\boldsymbol{\theta})$ outside $\mathcal{F}_{j+1}$. If a sample $\boldsymbol{\theta}^{(i)}$ in $\mathcal{V}$, which does not lie in $\mathcal{F}_{j+1}$, is predicted to fall in $\mathcal{F}_{j+1}$ by $\hat{g}(\boldsymbol{\theta})$, e.g. the first two training samples in Figure 4.9a, the generalization ability is not strong enough and the current surrogate $\hat{g}(\boldsymbol{\theta})$ should be further improved. In this case, all the misclassified samples in $\mathcal{V}$ are moved to $\widetilde{\mathcal{T}}$ and subsequently a new surrogate $\hat{g}(\boldsymbol{\theta})$ is built using the updated $\widetilde{\mathcal{T}}$, as depicted in Figure 4.9b. These steps are repeated until all the samples in $\mathcal{V}$ are correctly classified. This adaptive experimental design strategy is summarized in Algorithm 10. In essence, SuS can be

Algorithm 11 The PS-SuS approach

```
Input: The PDF of uncertain parameters $f(\boldsymbol{\theta})$; Limit-state function $g(\boldsymbol{\theta})$; Conditional
    probability $p_0$; The number of samples per subset level $N$; The percentage of points
    that are chosen as the initial training samples $\tilde{p}_0$.
Output: The estimate of failure probability $\hat{P}_F$.
 1: Draw $N$ i.i.d. samples $\{\boldsymbol{\theta}_0^{(i)} : i = 1, \dots, N\}$ in accordance with $f(\boldsymbol{\theta})$ and initialize an
    empty training set $\mathcal{T}$;
 2: Randomly select $\tilde{p}_0 N$ training samples $\{\boldsymbol{\theta}_{0,t}^{(i)} : i = 1, \dots, \tilde{p}_0 N\}$ from the $N$ samples,
    where $\tilde{p}_0 N$ is an integer, calculate the corresponding limit-state values $\{g(\boldsymbol{\theta}_{0,t}^{(i)}) : i =
    1, \dots, \tilde{p}_0 N\}$, and add these training samples to $\mathcal{T}$;
 3: Build a PCE or RSM surrogate model $\hat{g}(\boldsymbol{\theta})$ with $\mathcal{T}$ and estimate the limit-state values
    of the non-training samples $\{\boldsymbol{\theta}_{0,r}^{(i)} : i = 1, \dots, (1 - \tilde{p}_0)N\}$ using $\hat{g}(\boldsymbol{\theta})$;
 4: Find $b_1$ as the $p_0$-percentile of the $N$ responses $\{g(\boldsymbol{\theta}_{0,t}^{(i)})\} \cup \{\hat{g}(\boldsymbol{\theta}_{0,r}^{(i)})\}$ and set $\mathcal{F}_1 =
    \{\boldsymbol{\theta} \in \boldsymbol{\Omega} : g(\boldsymbol{\theta}) \le b_1\}$;
 5: Set $j = 1$;
 6: while $b_j > 0$ do
 7:     Consider samples $\boldsymbol{\theta}_{j-1}^{(i)} \in \mathcal{F}_j$ as seeds $\{\boldsymbol{\theta}_{j-1,s}^{(i)} : i = 1, \dots, N_s\}$, where $N_s = p_0 N$ is
        an integer;
 8:     Calculate the corresponding limit-state values $\{g(\boldsymbol{\theta}_{j-1,s}^{(i)}) : i = 1, \dots, N_s\}$ and add
        these seeds to $\mathcal{T}$;
 9:     Update $\hat{g}(\boldsymbol{\theta})$ employing the adaptive experimental design strategy (Algorithm 10);
10:     Propose $N$ candidate samples $\{\tilde{\boldsymbol{\theta}}_j^{(i)} : i = 1, \dots, N\}$ from the seeds;
11:     Estimate the limit-state values of these candidate samples using $\hat{g}(\boldsymbol{\theta})$;
12:     Accept or reject $\tilde{\boldsymbol{\theta}}_j^{(i)}$ according to whether $\hat{g}(\tilde{\boldsymbol{\theta}}_j^{(i)}) \le b_j$; After rejection, the samples
        are denoted by $\{\boldsymbol{\theta}_j^{(i)} : i = 1, \dots, N\} = \{\boldsymbol{\theta}_{j,t}^{(i)}\} \cup \{\boldsymbol{\theta}_{j,r}^{(i)}\}$;
13:     Find $b_{j+1}$ as the $p_0$-percentile of the $N$ responses $\{g(\boldsymbol{\theta}_{j,t}^{(i)})\} \cup \{\hat{g}(\boldsymbol{\theta}_{j,r}^{(i)})\}$ and set
        $\mathcal{F}_{j+1} = \{\boldsymbol{\theta} \in \boldsymbol{\Omega} : g(\boldsymbol{\theta}) \le b_{j+1}\}$;
14:     Set $j = j + 1$;
15: end while
16: Set the total number of subsets $m = j$, set $b_m = 0$, denote the number of samples
    $\boldsymbol{\theta}_{m-1}^{(i)} \in \mathcal{F}$ by $N_F$, and estimate the failure probability as in Eq. (2.54);
17: return $\hat{P}_F$.
```

considered as a series of classification problems. At the j-th level, samples are classified to two categories: those fall in $\mathcal{F}_{j+1}$ and those do not. The proposed adaptive strategy aims to achieve this classification task and provide accurate predictions for samples in $\mathcal{F}_{j+1}$. The latter goal is necessary to realize the precise classification at the following subset level. Consequently, this novel scheme attempts to gain the best accuracy for PS-SuS. Applying this strategy, the PS-SuS approach is given in Algorithm 11.

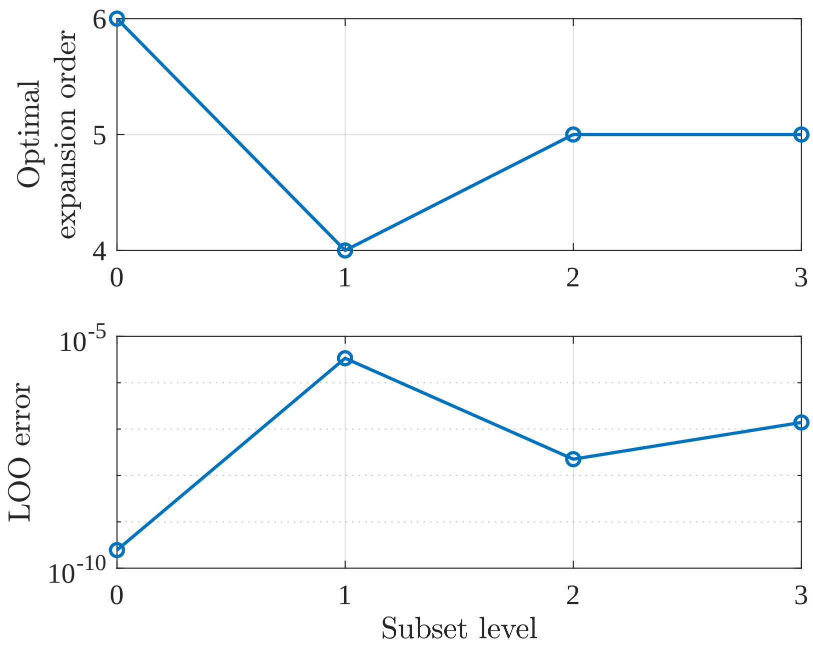


Figure 4.10: *Expansion orders and LOO errors at each level of PS-SuS*

The generalization ability of the constructed surrogate model is evaluated by the LOO error in Section 4.3.2. To assess the classification performance of the surrogate in the PS-SuS method, the *classification error* is defined as the ratio of training samples that are incorrectly classified:

$$\varepsilon_c = \frac{N_c}{N_t}, \tag{4.68}$$

where N_c is the number of misclassified training samples in $\mathcal{T}$ and N_t is the cardinality of $\mathcal{T}$. A large classification error indicates that the current global surrogate cannot successfully achieve the classification task.

4.4.3 Illustrative Examples

Consider the 3-dimensional limit-state function in Equation (4.59) where $a = 15$ and $\theta_i, i = 1, 2, 3$, are i.i.d. Gaussian random variables with zero means and unit standard deviations. The proposed PS-SuS method is implemented to evaluate the failure probability $\mathbb{P}\left[g(\boldsymbol{\theta}) \leq 0\right]$, with $N = 2000$, $p_0 = 0.1$, $\tilde{p}_0 = 0.12$, and $d_{\min} : d_{\max} = 1 : 7$.

At each subset level, adaptive PCE/RSM is performed to choose the best expansion order that results in the minimum LOO error. The optimal expansion orders and the corresponding LOO errors are depicted in Figure 4.10. Relatively high (4th–6th) orders turn out to be the best choices which obtain very small LOO errors. Table 4.8 shows the adaptive experimental design results. At each level, N_{call} samples are added to the full training set $\mathcal{T}$, and a training set $\widetilde{\mathcal{T}}$, which is a subset of $\mathcal{T}$ with N_{train} samples, is employed to build a surrogate. The initial surrogate at Lv. 2, for example, leads to

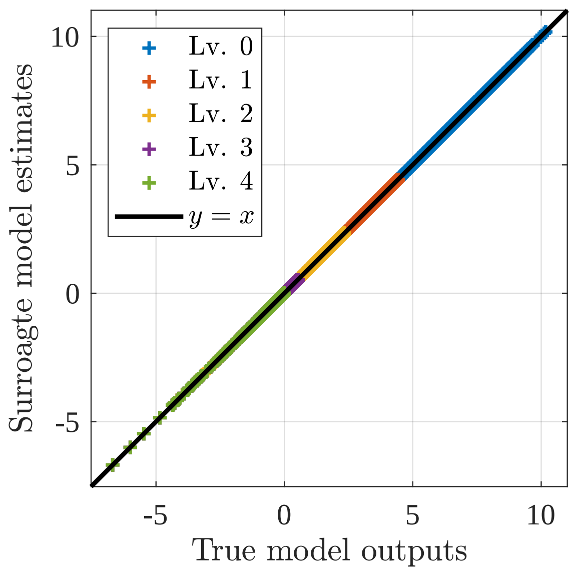


Figure 4.11: *Surrogate estimates at each level of PS-SuS.*

a classification error $\varepsilon_c = 0.031$. With the adaptive experimental design strategy (see Algorithm 10), the training set $\widetilde{\mathcal{T}}$ is enriched by 19 samples and the refined surrogate is able to properly classify all the samples in $\mathcal{T}$. By contrast, the initial surrogates at the first two levels are already accurate enough for the classification problem. In the end, the surrogates at all levels provide sufficiently accurate predictions for samples in $\widetilde{\mathcal{T}}$ (due to the very small LOO errors shown in Figure 4.10), as well as precise classification results for elements in $\mathcal{T}$.

Figure 4.11 compares the system outputs evaluated by the true model and the surrogate model. These points lie close to the line $y = x$. In accordance with the small LOO errors in Figure 4.10, this again proves the high accuracy of the surrogates. The failure probability estimation results of SuS and PS-SuS with $N = 2000$ are presented in Figure 4.12 and Table 4.9, where σ denotes the standard deviation of the estimation approximated by Equation (2.36) with $\mu \approx \hat{P}_F$ and $c_v \approx \hat{c}_{v,\mathrm{lb}}$. The estimation results of PS-SuS are quite similar to those of SuS, but only a fraction of true model evaluations are required by PS-SuS. This thereby proves that the PS-SuS approach achieves comparable accuracy but higher efficiency in contrast with SuS.

Table 4.8: *Adaptive experimental design results at each level of PS-SuS.*

Item	Lv. 0	Lv. 1	Lv. 2	Lv. 3
N_{call}	240	177	200	200
N_{train}	240	200	$214 \rightarrow 233$	$218 \rightarrow 231$
ε_c	0	0	$0.031 \rightarrow 0$	$0.016 \rightarrow 0$

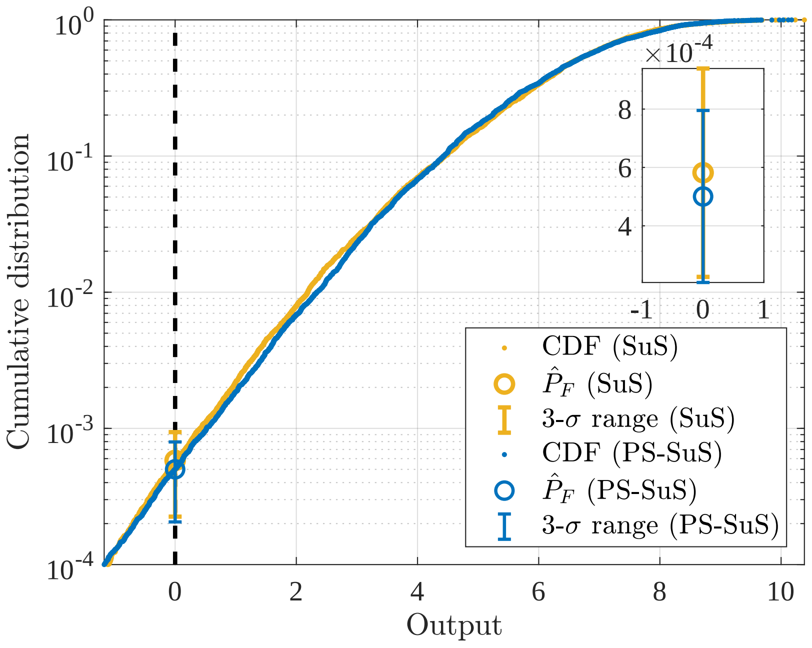


Figure 4.12: *Failure probability estimation results using SuS and PS-SuS.*

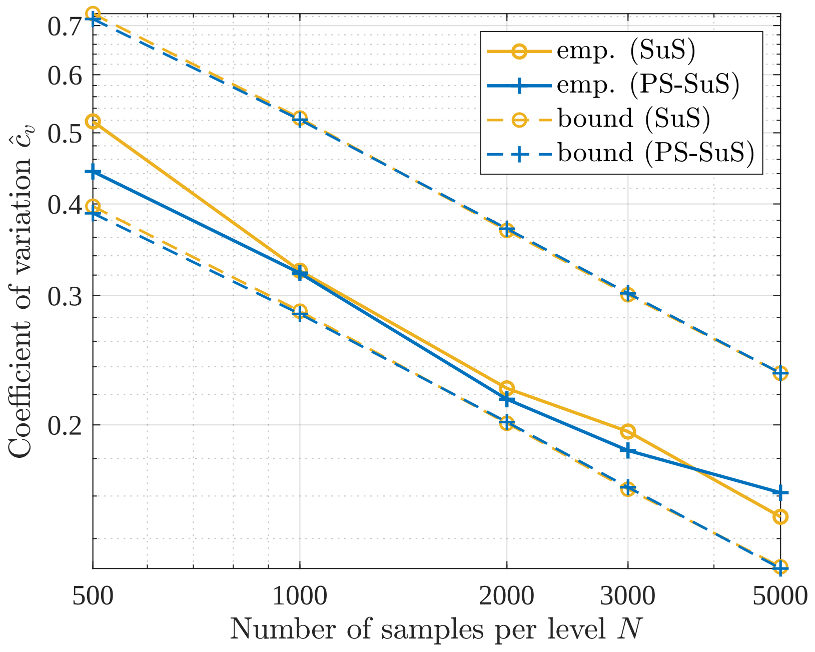
The efficiency and accuracy of the PS-SuS approach are further illustrated in Table 4.10 and Figure 4.13, where SuS and PS-SuS are carried out with different numbers of samples per level: $N = [500, 1000, 2000, 3000, 5000]^\mathsf{T}$. For each N, 50 repeated applications of these methods are conducted. Table 4.10 shows that the proposed method saves about 90% true model evaluations in comparison with conventional SuS. In Figure 4.13a, the c.o.v. upper and lower bounds are the average values of the 50 estimates of $\hat{c}_{v,\text{lb}}$ and $\hat{c}_{v,\text{ub}}$, respectively, whereas the empirical c.o.v., denoted by "emp.", is the sample c.o.v. of the

Table 4.9: *Failure probability estimation results using SuS and PS-SuS.*

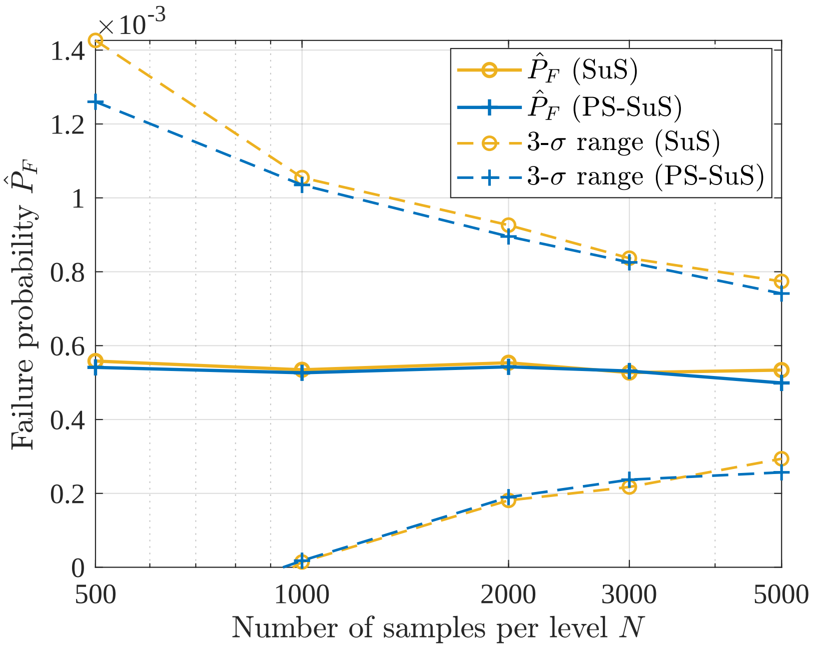
Method	$\hat{P}_F$	$[\hat{c}_{v,\text{lb}}, \hat{c}_{v,\text{ub}}]$	N_call
SuS	5.82×10^{-4}	$[0.20, 0.37]$	7933
PS-SuS	5.01×10^{-4}	$[0.20, 0.36]$	817

Table 4.10: *Average numbers of calls to the true model using SuS and PS-SuS.*

N	N_call^SuS	$N_\text{call}^\text{PS-SuS}$
500	1938	202
1000	3954	408
2000	7947	815
3000	11918	1221
5000	19867	2035



a) *Estimates of coefficients of variation*



b) *Estimates of failure probabilities*

Figure 4.13: *Failure probability estimation results using SuS and PS-SuS with different N.*

50 failure probability estimates. It is shown that the empirical results lie between the lower and upper bounds. In Figure 4.13b, the failure probability estimates are the mean values of the 50 results, and the 3-σ ranges are approximated using the average probability estimate and the empirical c.o.v. The statistical results of the two methods are very similar, demonstrating that the highly efficient PS-SuS achieves the same accuracy level as SuS in this example.

4.5 Summary

This chapter first introduced the principles of two polynomial-based global surrogates, i.e., *response surface method* (RSM) and *polynomial chaos expansion* (PCE). After that, the uncertainty propagation schemes based on RSM and PCE were summarized. Then, this chapter proposed the *polynomial surrogate-based SuS* (PS-SuS) method integrating these global surrogates into SuS. Herein, adaptive PCE or RSM was applied to progressively refine the surrogate and an experimental design strategy was designed to choose the most valuable training samples for the surrogate refinement.

Chapter 5

Moving Least-Squares-Accelerated Reliability Assessment

Two global surrogate models, RSM and PCE, have been utilized to mimic the behaviors of performance functions in Chapter 4. Though they are easy to implement, they may suffer from large estimation errors for highly nonlinear applications. To alleviate this problem, this chapter employs a local surrogate model called *moving least-squares* (MLS) to accelerate the simulation-based reliability analysis approaches. The principles of MLS are presented first. The following section introduces the integration of MLS into simulation methods and the strategy adaptively enriching the training set. After that, a two-stage framework for MLS-based simulation approaches is proposed to tackle high-dimensional problems based on a novel dimensionality reduction strategy. Finally, illustrative examples are given to demonstrate the accuracy and efficiency of the proposed methods.

5.1 Moving Least-Squares (MLS)

5.1.1 MLS Method

The MLS method was first proposed in [139] to interpolate and smooth data. It allows us to construct a surrogate model between interpolation and standard polynomial regression. Similar to standard RSM, the limit-state function $g(\boldsymbol{\theta})$ can be estimated in terms of *basis functions* $\psi_k(\boldsymbol{\theta})$ and corresponding adjusting *coefficients* $\alpha_k(\boldsymbol{\theta})$ as

$$\hat{g}(\boldsymbol{\theta}) = \sum_{k=1}^{p} \alpha_k(\boldsymbol{\theta})\psi_k(\boldsymbol{\theta}) = \boldsymbol{\alpha}(\boldsymbol{\theta})^\mathsf{T}\boldsymbol{\psi}(\boldsymbol{\theta}), \tag{5.1}$$

where p is the *number of expansion items*, $\boldsymbol{\psi}(\boldsymbol{\theta}) = [\psi_1(\boldsymbol{\theta}), \dots, \psi_p(\boldsymbol{\theta})]^\mathsf{T}$, and $\boldsymbol{\alpha}(\boldsymbol{\theta}) = [\alpha_1(\boldsymbol{\theta}), \dots, \alpha_p(\boldsymbol{\theta})]^\mathsf{T}$. The function $\boldsymbol{\psi}(\cdot)$ is also known as *feature mapping*, and the elements $\psi_k(\boldsymbol{\theta})$ are thus called *features*. In the absence of any specific knowledge about the nonlinearity of $g(\boldsymbol{\theta})$, it is common to choose the basis functions as linear and quadratic monomials [45]:

$$\boldsymbol{\psi}(\boldsymbol{\theta}) = \left[1, \theta_1, \theta_2, \dots, \theta_n, \theta_1\theta_2 \dots, \theta_i\theta_j, \dots, \theta_1^2, \theta_2^2, \dots, \theta_n^2\right]^\mathsf{T}. \tag{5.2}$$

In this thesis, monomials up to the 2nd order are utilized as basis functions.

Given the training set $\mathcal{T} = \{(\boldsymbol{\theta}_t^{(i)}, g(\boldsymbol{\theta}_t^{(i)})) : i = 1, \dots, N_t\}$, the coefficients can be estimated by the *weighted least-squares* (WLS) method [45] minimizing the weighted residual sum of squares:

$$\begin{aligned}\boldsymbol{\alpha}(\boldsymbol{\theta}) &= \arg\min \sum_{i=1}^{N_t} w(\boldsymbol{\theta}_t^{(i)} - \boldsymbol{\theta}) \left(\boldsymbol{\alpha}(\boldsymbol{\theta})^\mathsf{T}\boldsymbol{\psi}(\boldsymbol{\theta}_t^{(i)}) - g(\boldsymbol{\theta}_t^{(i)})\right)^2 \\ &= \left(\boldsymbol{\Psi}^\mathsf{T}\boldsymbol{W}\boldsymbol{\Psi}\right)^{-1} \boldsymbol{\Psi}^\mathsf{T}\boldsymbol{W}\boldsymbol{y},\end{aligned} \tag{5.3}$$

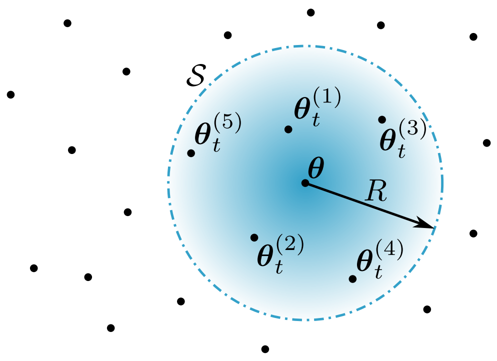
where $w(\boldsymbol{\theta}_t^{(i)} - \boldsymbol{\theta})$ is the *weight* defining the relative importance of $(\boldsymbol{\theta}_t^{(i)}, g(\boldsymbol{\theta}_t^{(i)}))$, $\boldsymbol{W}$ is the *weight matrix*, $\boldsymbol{\Psi}$ is the *experimental matrix*, and $\boldsymbol{y}$ is a vector of *true function responses*. Let n_t denote the *number of activated samples* whose weights are greater than 0. Without loss of generality, assume that the first n_t samples in the training set are activated, then

$$\boldsymbol{W} = \begin{bmatrix} w(\boldsymbol{\theta}_t^{(1)} - \boldsymbol{\theta}) & 0 & \cdots & 0 \\ 0 & w(\boldsymbol{\theta}_t^{(2)} - \boldsymbol{\theta}) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & w(\boldsymbol{\theta}_t^{(n_t)} - \boldsymbol{\theta}) \end{bmatrix}, \tag{5.4}$$

$$\boldsymbol{\Psi} = \begin{bmatrix} \boldsymbol{\psi}(\boldsymbol{\theta}_t^{(1)})^\mathsf{T} \\ \boldsymbol{\psi}(\boldsymbol{\theta}_t^{(2)})^\mathsf{T} \\ \vdots \\ \boldsymbol{\psi}(\boldsymbol{\theta}_t^{(n_t)})^\mathsf{T} \end{bmatrix} = \begin{bmatrix} \psi_1(\boldsymbol{\theta}_t^{(1)}) & \psi_2(\boldsymbol{\theta}_t^{(1)}) & \cdots & \psi_p(\boldsymbol{\theta}_t^{(1)}) \\ \psi_1(\boldsymbol{\theta}_t^{(2)}) & \psi_2(\boldsymbol{\theta}_t^{(2)}) & \cdots & \psi_p(\boldsymbol{\theta}_t^{(2)}) \\ \vdots & \vdots & \ddots & \vdots \\ \psi_1(\boldsymbol{\theta}_t^{(n_t)}) & \psi_2(\boldsymbol{\theta}_t^{(n_t)}) & \cdots & \psi_p(\boldsymbol{\theta}_t^{(n_t)}) \end{bmatrix}, \tag{5.5}$$

and $\boldsymbol{y} = [g(\boldsymbol{\theta}_t^{(1)}), \dots, g(\boldsymbol{\theta}_t^{(n_t)})]^\mathsf{T}$. The weight function $w(\cdot)$ plays a crucial role as it determines the degree to which the samples $\boldsymbol{\theta}_t^{(i)}$ influence the *query point* $\boldsymbol{\theta}$, i.e., the point to be predicted. It decays continuously with increasing distance $\|\boldsymbol{\theta}_t^{(i)} - \boldsymbol{\theta}\|_2$ and vanishes beyond a certain region. This means that only the points closest to $\boldsymbol{\theta}$ contribute to the prediction of $g(\boldsymbol{\theta})$, and thus the approximation is local. The fixed region is called *support domain* $\mathcal{S}$. This concept is illustrated in Figure 5.1. Here, samples $\boldsymbol{\theta}_t^{(i)}$, $i = 1, \dots, 5$, fall in the support domain of the query point $\boldsymbol{\theta}$. Darker blue shades represent larger weights and R denotes the *radius of the support domain*. The support domain moves with the query point. This is the reason why this method is called "moving" least-squares. It also indicates that the computation in Equations (5.1) and (5.3) must be carried out at every query point. In this thesis, the *cubic spline weight function* [140] as follows is applied:

$$w(\boldsymbol{\theta}_t^{(i)} - \boldsymbol{\theta}) = \begin{cases} 4r_\theta^3 - 4r_\theta^2 + \dfrac{2}{3}, & r_\theta \leq \dfrac{1}{2}, \\ -\dfrac{4}{3}r_\theta^3 + 4r_\theta^2 - 4r_\theta + \dfrac{4}{3}, & \dfrac{1}{2} < r_\theta \leq 1, \\ 0, & r_\theta > 1, \end{cases} \tag{5.6}$$



$\mathcal{S}$
$\boldsymbol{\theta}_t^{(1)}$
$\boldsymbol{\theta}_t^{(2)}$
$\boldsymbol{\theta}_t^{(3)}$
$\boldsymbol{\theta}_t^{(4)}$
$\boldsymbol{\theta}_t^{(5)}$
$\boldsymbol{\theta}$
R

Figure 5.1: *Support domain of MLS.*

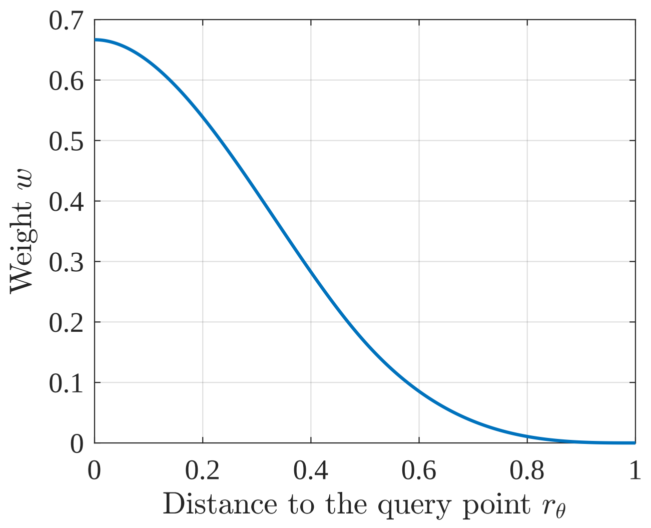


Figure 5.2: *Shape of the cubic spline weight function.*

where $r_\theta = \|\boldsymbol{\theta}_t^{(i)} - \boldsymbol{\theta}\|_2/R$. The shape of the cubic spline weight function is shown in Figure 5.2. Besides, the *quartic spline function* and the *exponential function* [140] are another two popular weight functions:

$$w(\boldsymbol{\theta}_t^{(i)} - \boldsymbol{\theta}) = \begin{cases} -3r_\theta^4 + 8r_\theta^3 - 6r_\theta^2 + 1, & r_\theta \le 1, \\ 0, & r_\theta > 1, \end{cases} \tag{5.7}$$

$$w(\boldsymbol{\theta}_t^{(i)} - \boldsymbol{\theta}) = \begin{cases} \exp\left(-\frac{r_\theta^2}{\beta^2}\right), & r_\theta \le 1, \\ 0, & r_\theta > 1, \end{cases} \tag{5.8}$$

where β is the *shape parameter*. These two functions exhibit similar bell shape as the cubic spline weight function depicted in Figure 5.2.

In practice, to address the multicollinearity problem, the *ridge regression* (RR) estimator [136, Ch. 1] minimizing the penalized weighted residual sum of squares is applied:

$$\begin{aligned} \boldsymbol{\alpha}(\boldsymbol{\theta}) &= \arg\min \sum_{i=1}^{N_t} w(\boldsymbol{\theta}_t^{(i)} - \boldsymbol{\theta}) \left(\boldsymbol{\alpha}(\boldsymbol{\theta})^\mathsf{T} \boldsymbol{\psi}(\boldsymbol{\theta}_t^{(i)}) - g(\boldsymbol{\theta}_t^{(i)})\right)^2 + \lambda \|\boldsymbol{\alpha}(\boldsymbol{\theta})\|_2^2 \\ &= \left(\boldsymbol{\Psi}^\mathsf{T} \boldsymbol{W} \boldsymbol{\Psi} + \lambda \boldsymbol{I}_p\right)^{-1} \boldsymbol{\Psi}^\mathsf{T} \boldsymbol{W} \boldsymbol{y}, \end{aligned} \tag{5.9}$$

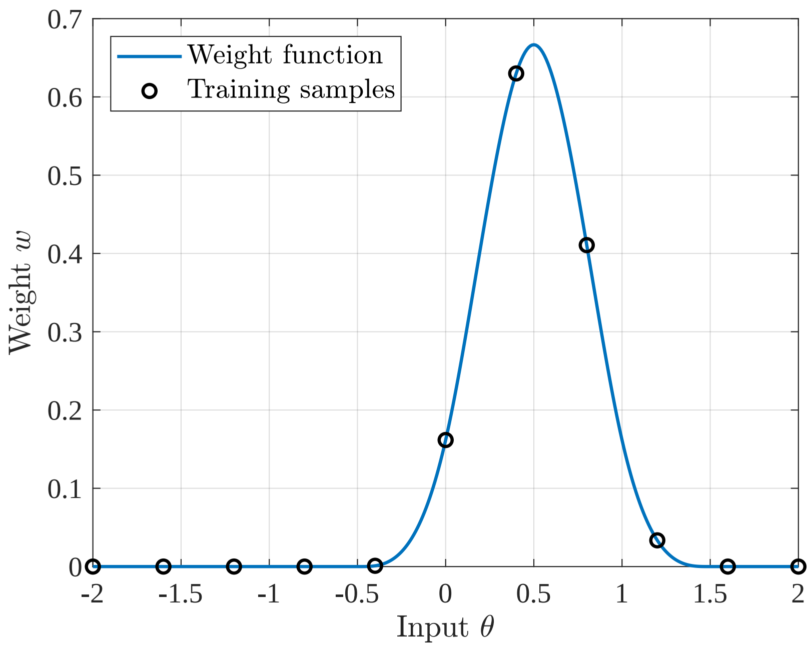


Figure 5.3: *MLS weights given query point $\theta = 0.5$.*

where $\boldsymbol{I}_p$ is the $p \times p$ *identity matrix* and λ is the *ridge parameter* which is usually a small positive constant.

An example implementing the MLS surrogate model is given in the following. Consider a 1-dimensional function expressed as

$$g(\theta) = \frac{1}{1 + 10\theta^2}, \quad -2 \leq \theta \leq 2. \tag{5.10}$$

Both MLS and RSM are applied to build surrogate models based on 11 given training samples. Basis functions up to the 2nd order are used in MLS, whereas a 6th-order polynomial expansion is adopted in RSM. Given a query point $\theta = 0.5$, the weight function for MLS is depicted in Figure 5.3. Only the training samples close to the query point are considered when predicting the function value of the query point. Figure 5.4 shows the fitting results of both surrogate modeling methods. In comparison with RSM, the MLS regression model, which provides low-order local approximations, is more accurate and flexible for approximating functions without a priori knowledge about the nonlinearity.

5.1.2 Kernel Ridge Regression

In this subsection, *kernel ridge regression* (KRR) [141, Ch. 14], which combines RR with the kernel trick, is applied to solve the MLS regression problem in a more flexible manner.

The concept of kernel is first introduced. Given vectors $\boldsymbol{x}, \boldsymbol{z} \in \mathbb{R}^n$ and a feature mapping $\boldsymbol{\psi}(\cdot)$, the *kernel function* [142] is defined as

$$k(\boldsymbol{x}, \boldsymbol{z}) = \boldsymbol{\psi}(\boldsymbol{x})^{\mathsf{T}} \boldsymbol{\psi}(\boldsymbol{z}). \tag{5.11}$$

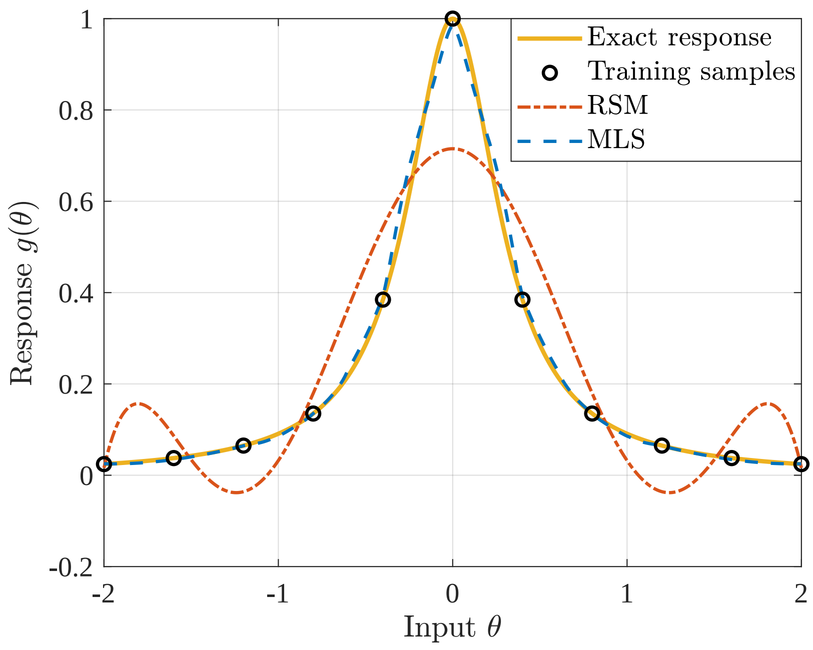


Figure 5.4: *Response approximation using MLS and RSM*

The kernel function enables us to conduct the inner product in the feature space by simply calculating the inner product in the original space, instead of explicitly computing the inner product of the features. In general, the former operation is computationally cheaper than the latter one. This approach is called the *kernel trick*.

Consider an example where the following polynomial kernel is used:

$$k(\boldsymbol{x}, \boldsymbol{z}) = \left(1 + \boldsymbol{x}^{\mathsf{T}} \boldsymbol{z}\right)^{d}. \tag{5.12}$$

Let $n = 2$, $\boldsymbol{x} = [x_1, x_2]^{\mathsf{T}}$, $\boldsymbol{z} = [z_1, z_2]^{\mathsf{T}}$, and $d = 2$, Equation (5.12) can be written as

$$\begin{aligned} k(\boldsymbol{x}, \boldsymbol{z}) &= (1 + \langle \boldsymbol{x}, \boldsymbol{z} \rangle)^2 \\ &= 1 + 2x_1 z_1 + 2x_2 z_2 + 2x_1 x_2 z_1 z_2 + x_1^2 z_1^2 + x_2^2 z_2^2 \\ &= \langle \boldsymbol{\psi}(\boldsymbol{x}), \boldsymbol{\psi}(\boldsymbol{z}) \rangle, \end{aligned} \tag{5.13}$$

which corresponds to the feature mapping

$$\boldsymbol{\psi}(\boldsymbol{x}) = \left[1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1 x_2, x_1^2, x_2^2\right]^{\mathsf{T}}. \tag{5.14}$$

This mapping is the same as the basis functions in Equation (5.2). The computation of the inner product in a high-dimensional feature space $\langle \boldsymbol{\psi}(\boldsymbol{x}), \boldsymbol{\psi}(\boldsymbol{z}) \rangle_{\mathbb{R}^6}$ can be simplified to its counterpart in the original space $\langle \boldsymbol{x}, \boldsymbol{z} \rangle_{\mathbb{R}^2}$. More generally, the polynomial kernel in Equation (5.12) corresponds to a feature mapping of dimension

$$p = \frac{(n+d)!}{n!d!}. \tag{5.15}$$

Though computing the corresponding $\boldsymbol{\psi}(\cdot)$ is of complexity $\mathcal{O}(n^d)$, calculating $k(\boldsymbol{x}, \boldsymbol{z})$ is of complexity $\mathcal{O}(n)$.

Recall that from Equation (5.9), one gets

$$\begin{aligned}\left(\boldsymbol{\Psi}^\mathsf{T}\boldsymbol{W}\boldsymbol{\Psi}+\lambda\boldsymbol{I}_p\right)\boldsymbol{\alpha}&=\boldsymbol{\Psi}^\mathsf{T}\boldsymbol{W}\boldsymbol{y}\\ \Rightarrow\boldsymbol{\alpha}&=\lambda^{-1}\left(\boldsymbol{\Psi}^\mathsf{T}\boldsymbol{W}\boldsymbol{y}-\boldsymbol{\Psi}^\mathsf{T}\boldsymbol{W}\boldsymbol{\Psi}\boldsymbol{\alpha}\right)\\ &=\lambda^{-1}\boldsymbol{\Psi}^\mathsf{T}\left(\boldsymbol{W}\boldsymbol{y}-\boldsymbol{W}\boldsymbol{\Psi}\boldsymbol{\alpha}\right).\end{aligned}\tag{5.16}$$

Let $\boldsymbol{\alpha}=\boldsymbol{\Psi}^\mathsf{T}\boldsymbol{\beta}$, then

$$\boldsymbol{\Psi}^\mathsf{T}\boldsymbol{\beta}=\lambda^{-1}\boldsymbol{\Psi}^\mathsf{T}\left(\boldsymbol{W}\boldsymbol{y}-\boldsymbol{W}\boldsymbol{\Psi}\boldsymbol{\alpha}\right).\tag{5.17}$$

This equation holds if the following equation is satisfied:

$$\boldsymbol{\beta}=\lambda^{-1}\left(\boldsymbol{W}\boldsymbol{y}-\boldsymbol{W}\boldsymbol{\Psi}\boldsymbol{\alpha}\right).\tag{5.18}$$

Equation (5.18) can be written as

$$\begin{aligned}\lambda\boldsymbol{\beta}&=\boldsymbol{W}\boldsymbol{y}-\boldsymbol{W}\boldsymbol{\Psi}\boldsymbol{\alpha}\\ &=\boldsymbol{W}\boldsymbol{y}-\boldsymbol{W}\boldsymbol{\Psi}\boldsymbol{\Psi}^\mathsf{T}\boldsymbol{\beta}\\ \Rightarrow\left(\boldsymbol{W}\boldsymbol{\Psi}\boldsymbol{\Psi}^\mathsf{T}+\lambda\boldsymbol{I}_{n_t}\right)\boldsymbol{\beta}&=\boldsymbol{W}\boldsymbol{y}.\end{aligned}\tag{5.19}$$

Thus, one obtains

$$\boldsymbol{\beta}=\left(\boldsymbol{W}\boldsymbol{K}+\lambda\boldsymbol{I}_{n_t}\right)^{-1}\boldsymbol{W}\boldsymbol{y},\tag{5.20}$$

where

$$\begin{aligned}\boldsymbol{K}=\boldsymbol{\Psi}\boldsymbol{\Psi}^\mathsf{T}&=\begin{bmatrix}\boldsymbol{\psi}(\boldsymbol{\theta}_t^{(1)})^\mathsf{T}\\ \boldsymbol{\psi}(\boldsymbol{\theta}_t^{(2)})^\mathsf{T}\\ \vdots\\ \boldsymbol{\psi}(\boldsymbol{\theta}_t^{(n_t)})^\mathsf{T}\end{bmatrix}\left[\boldsymbol{\psi}(\boldsymbol{\theta}_t^{(1)}),\boldsymbol{\psi}(\boldsymbol{\theta}_t^{(2)}),\ldots,\boldsymbol{\psi}(\boldsymbol{\theta}_t^{(n_t)})\right]\\ &=\begin{bmatrix}k(\boldsymbol{\theta}_t^{(1)},\boldsymbol{\theta}_t^{(1)}) & k(\boldsymbol{\theta}_t^{(1)},\boldsymbol{\theta}_t^{(2)}) & \cdots & k(\boldsymbol{\theta}_t^{(1)},\boldsymbol{\theta}_t^{(n_t)})\\ k(\boldsymbol{\theta}_t^{(2)},\boldsymbol{\theta}_t^{(1)}) & k(\boldsymbol{\theta}_t^{(2)},\boldsymbol{\theta}_t^{(2)}) & \cdots & k(\boldsymbol{\theta}_t^{(2)},\boldsymbol{\theta}_t^{(n_t)})\\ \vdots & \vdots & \ddots & \vdots\\ k(\boldsymbol{\theta}_t^{(n_t)},\boldsymbol{\theta}_t^{(1)}) & k(\boldsymbol{\theta}_t^{(n_t)},\boldsymbol{\theta}_t^{(2)}) & \cdots & k(\boldsymbol{\theta}_t^{(n_t)},\boldsymbol{\theta}_t^{(n_t)})\end{bmatrix}\end{aligned}\tag{5.21}$$

is the kernel matrix.

Therefore, Equation (5.1) can be represented as

$$\hat{g}(\boldsymbol{\theta})=\boldsymbol{\beta}^\mathsf{T}\boldsymbol{\Psi}\boldsymbol{\psi}(\boldsymbol{\theta})=\boldsymbol{\beta}^\mathsf{T}\boldsymbol{\kappa}(\boldsymbol{\theta})\tag{5.22}$$

with

$$\boldsymbol{\kappa}(\boldsymbol{\theta})=\boldsymbol{\Psi}\boldsymbol{\psi}(\boldsymbol{\theta})=\begin{bmatrix}\boldsymbol{\psi}(\boldsymbol{\theta}_t^{(1)})^\mathsf{T}\\ \boldsymbol{\psi}(\boldsymbol{\theta}_t^{(2)})^\mathsf{T}\\ \vdots\\ \boldsymbol{\psi}(\boldsymbol{\theta}_t^{(n_t)})^\mathsf{T}\end{bmatrix}\boldsymbol{\psi}(\boldsymbol{\theta})=\begin{bmatrix}k(\boldsymbol{\theta}_t^{(1)},\boldsymbol{\theta})\\ k(\boldsymbol{\theta}_t^{(2)},\boldsymbol{\theta})\\ \vdots\\ k(\boldsymbol{\theta}_t^{(n_t)},\boldsymbol{\theta})\end{bmatrix}.\tag{5.23}$$

Compared with Equations (5.1) and (5.9), the MLS solution applying Equations (5.20) and (5.22) does not require the calculation of features, but the computation of the kernel function instead. Consequently, KRR is more flexible for complex fitting than RR. For basis

Table 5.1: *Comparisons between RR and KRR.*

Method	Flexibility	Efficiency when $n_t \gg p$
RR	More restricted	More efficient
KRR	More flexible	Less efficient

functions up to the 2nd order (see Equation (5.2)), one can simply use the polynomial kernel in Equation (5.12) with $d = 2$. Likewise, d can be modified accordingly if a polynomial expansion of different order is required. Moreover, users can specify any type of basis functions by defining the corresponding kernel function.

To obtain sufficient regression accuracy, the number of activated training samples n_t, i.e., the number of samples in the support domain, generally should be larger than the number of unknown coefficients p. In this case, calculating the inverse of a square matrix of order n_t for KRR (see Equation (5.20)) requires more computational cost than calculating the inverse of a square matrix of order p for RR (see Equation (5.9)). This suggests that RR is more efficient than KRR when $n_t \gg p$. The comparisons between RR and KRR are summarized in Table 5.1. These two estimators are different tradeoffs between flexibility and efficiency. In the context of MLS, KRR with $n_t \in [1.2p, 2p]$ is recommended to find a balance between estimation accuracy and efficiency.

5.2 MLS-Based Simulation Methods

This section combines MLS with MCS and SuS, and then proposes an active learning strategy to reduce the influence of the surrogate error on the final estimation result.

5.2.1 MLS-Accelerated MCS

Similar with RSM and PCE (see Section 4.3.1), the MLS surrogate model can be employed to accelerate MCS by replacing a large proportion of true model evaluations with the surrogate model predictions. The general idea of the *MLS-accelerated MCS* (MLS-MCS) method is illustrated in Figure 5.5.

First, N samples $\boldsymbol{\theta}^{(i)}, i = 1, \ldots, N$, are generated by MCS. After that, a fraction of them are chosen as training samples $\boldsymbol{\theta}_t^{(i)}, i = 1, \ldots, N_t$, and the rest non-training points are represented by $\boldsymbol{\theta}_r^{(i)}$. Then, the training samples are evaluated using the true model $g(\boldsymbol{\theta})$, and the training set is specified as $\mathcal{T} = \{(\boldsymbol{\theta}_t^{(i)}, g(\boldsymbol{\theta}_t^{(i)}))\}$. Exploiting this training set, an MLS surrogate model $\hat{g}(\boldsymbol{\theta})$ is built to predict the model outputs of $\boldsymbol{\theta}_r^{(i)}$. With the function values $\{g(\boldsymbol{\theta}_t^{(i)})\} \cup \{\hat{g}(\boldsymbol{\theta}_r^{(i)})\}$, the failure probability is estimated as the percentage of samples lying in the failure domain. The numbers of calls to the true model for MCS

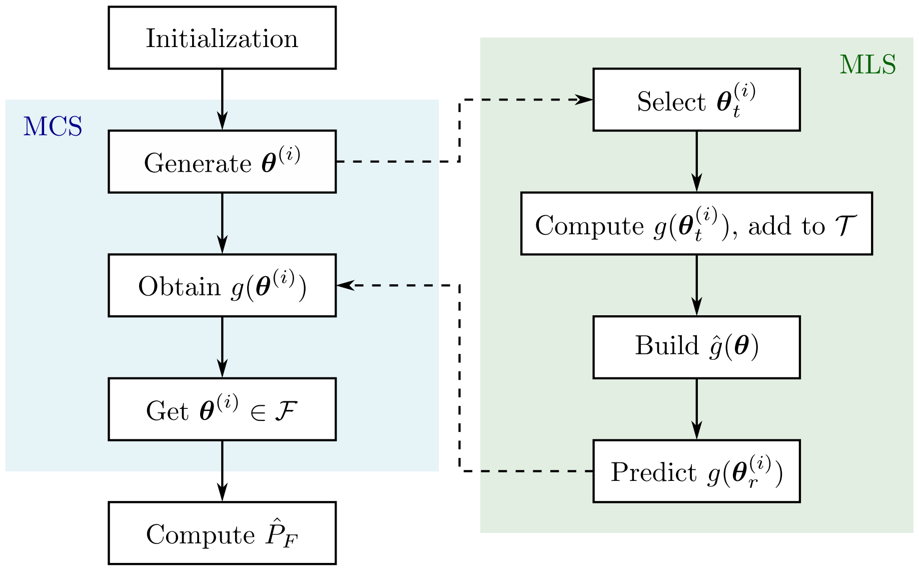

Initialization
MCS
Generate $\boldsymbol{\theta}^{(i)}$
Obtain $g(\boldsymbol{\theta}^{(i)})$
Get $\boldsymbol{\theta}^{(i)} \in \mathcal{F}$
Compute $\hat{P}_F$
MLS
Select $\boldsymbol{\theta}_t^{(i)}$
Compute $g(\boldsymbol{\theta}_t^{(i)})$, add to $\mathcal{T}$
Build $\hat{g}(\boldsymbol{\theta})$
Predict $g(\boldsymbol{\theta}_r^{(i)})$

Figure 5.5: *General idea of MLS-MCS.*

and MLS-MCS are given by

$$N_{\text{call}}^{\text{MCS}} = N, \tag{5.24}$$

$$N_{\text{call}}^{\text{MLS-MCS}} = N_t. \tag{5.25}$$

5.2.2 MLS-Accelerated SuS

For rare events, the MLS surrogate can be integrated into the SuS approach to achieve efficient and precise estimations. The general idea of *MLS-accelerated SuS* (MLS-SuS) is visualized in Figure 5.6.

At the initial level of SuS, the samples $\boldsymbol{\theta}_0^{(i)}$ are generated by MCS. Whereas at the following subset levels ($j = 1, \ldots, m-1$), the samples $\boldsymbol{\theta}_j^{(i)}$ are generated by the MCMC sampling method. MCMC sampling strategies can generally be divided into two main steps, i.e., the proposal of candidate samples $\tilde{\boldsymbol{\theta}}_j^{(i)}$ and the rejection of candidates according to whether or not they lie in the intermediate failure domain. The samples after the rejection step are denoted by $\boldsymbol{\theta}_j^{(i)}$. With conventional SuS, the limit-state values of samples $\boldsymbol{\theta}_0^{(i)}$ and $\tilde{\boldsymbol{\theta}}_j^{(i)}$ are all evaluated using the true model $g(\boldsymbol{\theta})$. In MLS-SuS, however, a small set of training points $\boldsymbol{\theta}_{0,t}^{(i)}$ or $\tilde{\boldsymbol{\theta}}_{j,t}^{(i)}$ are selected from these samples and added to the training set $\mathcal{T}$ to create or refine a surrogate model $\hat{g}(\boldsymbol{\theta})$. Afterwards, the limit-state values of the remaining non-training samples $\boldsymbol{\theta}_{0,r}^{(i)}$ or $\tilde{\boldsymbol{\theta}}_{j,r}^{(i)}$ are predicted by the surrogate model. The intermediate thresholds $b_j, j = 1, \ldots, m-1$, are set as the p_0-percentiles of function values $\{g(\boldsymbol{\theta}_{j-1,t}^{(i)})\} \cup \{\hat{g}(\boldsymbol{\theta}_{j-1,r}^{(i)})\}$, and b_m is set as 0. Accordingly, the intermediate failure domains $\mathcal{F}_j$, $j = 1, \ldots, m$, are defined as $\mathcal{F}_j = \{\boldsymbol{\theta} \in \boldsymbol{\Omega} : g(\boldsymbol{\theta}) \leq b_j\}$. The samples $\boldsymbol{\theta}_{j-1}^{(i)} \in \mathcal{F}_j, j = 1, \ldots, m-1$, are selected as seeds for proposing new samples at the next subset level.

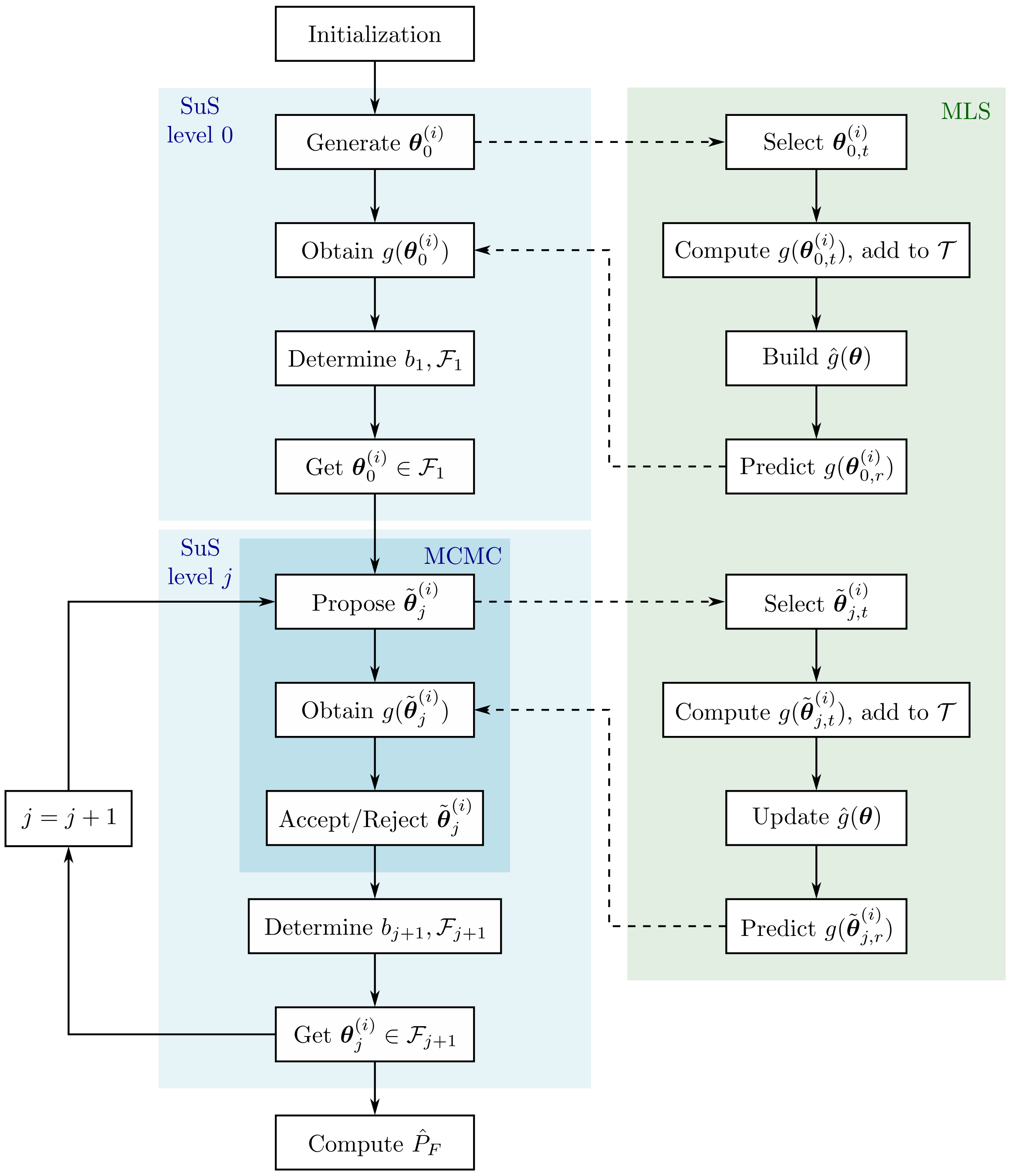

Initialization
SuS level 0
Generate $\boldsymbol{\theta}_0^{(i)}$
Obtain $g(\boldsymbol{\theta}_0^{(i)})$
Determine $b_1, \mathcal{F}_1$
Get $\boldsymbol{\theta}_0^{(i)} \in \mathcal{F}_1$
MLS
Select $\boldsymbol{\theta}_{0,t}^{(i)}$
Compute $g(\boldsymbol{\theta}_{0,t}^{(i)})$, add to $\mathcal{T}$
Build $\hat{g}(\boldsymbol{\theta})$
Predict $g(\boldsymbol{\theta}_{0,r}^{(i)})$
SuS level j
MCMC
Propose $\tilde{\boldsymbol{\theta}}_j^{(i)}$
Obtain $g(\tilde{\boldsymbol{\theta}}_j^{(i)})$
Accept/Reject $\tilde{\boldsymbol{\theta}}_j^{(i)}$
Determine $b_{j+1}, \mathcal{F}_{j+1}$
Get $\boldsymbol{\theta}_j^{(i)} \in \mathcal{F}_{j+1}$
$j = j + 1$
Select $\tilde{\boldsymbol{\theta}}_{j,t}^{(i)}$
Compute $g(\tilde{\boldsymbol{\theta}}_{j,t}^{(i)})$, add to $\mathcal{T}$
Update $\hat{g}(\boldsymbol{\theta})$
Predict $g(\tilde{\boldsymbol{\theta}}_{j,r}^{(i)})$
Compute $\hat{P}_F$

Figure 5.6: *General idea of MLS-SuS.*

In contrast with conventional SuS, the new MLS-SuS method significantly reduces the number of calls to the true model N_{call}. Applying the component-wise MH algorithm (see Algorithm 6), the numbers of true model evaluations for SuS and MLS-SuS with m levels are given by

$$N_{\text{call}}^{\text{SuS}} \approx mN, \tag{5.26}$$

$$N_{\text{call}}^{\text{MLS-SuS}} = \sum_{j=0}^{m-1} N_{j,t}, \tag{5.27}$$

where N is the number of generated samples at each subset level and $N_{j,t}$ is the number of training samples chosen at the j-th level. Because of the rejection of candidates with a given probability, $N_{\text{call}}^{\text{SuS}}$ is smaller than mN but gets close to mN as the dimension grows.

5.2.3 Active Learning Strategy

The previous subsections introduce the fundamental idea of combining the MLS surrogate with simulation-based reliability analysis methods. This subsection proposes an active learning strategy that adding more training samples to the training set to improve the estimation accuracy.

MCS can be regarded as the SuS with only a single level, and SuS can be summarized as a repeated process of sampling and classification. This interpretation is shown in Figure 5.7. At each level ($j = 0, 1, \ldots, m-1$), samples are drawn and classified into two categories, i.e., whether $\boldsymbol{\theta}_j^{(i)} \in \mathcal{F}_{j+1}$ or not. Besides that, at subset levels ($j = 1, \ldots, m-1$), the rejection of candidate points in MCMC sampling is also a classification problem: judging whether or not $\tilde{\boldsymbol{\theta}}_j^{(i)} \in \mathcal{F}_j$.

In MLS-MCS and MLS-SuS, the classification is performed based on function values $\{g(\boldsymbol{\theta}_{j,t}^{(i)})\} \cup \{\hat{g}(\boldsymbol{\theta}_{j,r}^{(i)})\}$ or $\{g(\tilde{\boldsymbol{\theta}}_{j,t}^{(i)})\} \cup \{\hat{g}(\tilde{\boldsymbol{\theta}}_{j,r}^{(i)})\}$. However, the predictions provided by the surrogate model may not be sufficiently precise, thus degrading the classification accuracy. Moreover, the thresholds b_j are set as the p_0-percentiles of $\{g(\boldsymbol{\theta}_{j,t}^{(i)})\} \cup \{\hat{g}(\boldsymbol{\theta}_{j,r}^{(i)})\}$, which means that the classification boundary is influenced by the surrogate error as well.

To mitigate the effect of the surrogate error on the classification, more training samples around the intermediate boundary $\{\boldsymbol{\theta} \in \boldsymbol{\Omega} : g(\boldsymbol{\theta}) = b_j\}$ are added to the training set and the surrogate model nearby is thus refined. For this purpose, we predict not only the limit-state values but also the variations of these estimations. Leave-one-out (LOO) cross validation [137, Ch. 7] is performed for training samples. It removes one point $\boldsymbol{\theta}_t^{(k)}$ from the training set $\mathcal{T} = \{(\boldsymbol{\theta}_t^{(i)}, g(\boldsymbol{\theta}_t^{(i)}))\}$ and builds a surrogate model $\hat{g}^{(-k)}(\boldsymbol{\theta})$ with the remaining training points. The LOO error at $\boldsymbol{\theta}_t^{(k)}$ is defined as

$$\varepsilon_t^{(k)} = \left| g(\boldsymbol{\theta}_t^{(k)}) - \hat{g}^{(-k)}(\boldsymbol{\theta}_t^{(k)}) \right|. \tag{5.28}$$

This error reflects the gap between the actual nonlinearity and the nonlinearity assumed in MLS. For non-training points $\boldsymbol{\theta}_r^{(i)}$, we build another MLS surrogate model $\hat{h}(\boldsymbol{\theta})$ using training samples $\{(\boldsymbol{\theta}_t^{(i)}, \varepsilon_t^{(i)})\}$, and then approximate the prediction errors of the non-training points $\hat{\epsilon}_r^{(i)}$. The prediction intervals are regarded as $[\hat{g}(\boldsymbol{\theta}_r^{(i)}) - \hat{\epsilon}_r^{(i)}, \hat{g}(\boldsymbol{\theta}_r^{(i)}) + \hat{\epsilon}_r^{(i)}]$. So far, two surrogate models are built: one for the limit-state function and the other one for its prediction error. Employing RR (see Section 5.1.1), these two surrogates are given by

$$\hat{g}(\boldsymbol{\theta}) = \boldsymbol{\psi}(\boldsymbol{\theta})^\mathsf{T} \left(\boldsymbol{\Psi}^\mathsf{T} \boldsymbol{W} \boldsymbol{\Psi} + \lambda \boldsymbol{I}_p\right)^{-1} \boldsymbol{\Psi}^\mathsf{T} \boldsymbol{W} \boldsymbol{y}, \tag{5.29}$$

$$\hat{h}(\boldsymbol{\theta}) = \boldsymbol{\psi}(\boldsymbol{\theta})^\mathsf{T} \left(\boldsymbol{\Psi}^\mathsf{T} \boldsymbol{W} \boldsymbol{\Psi} + \lambda \boldsymbol{I}_p\right)^{-1} \boldsymbol{\Psi}^\mathsf{T} \boldsymbol{W} \boldsymbol{\varepsilon}_t, \tag{5.30}$$

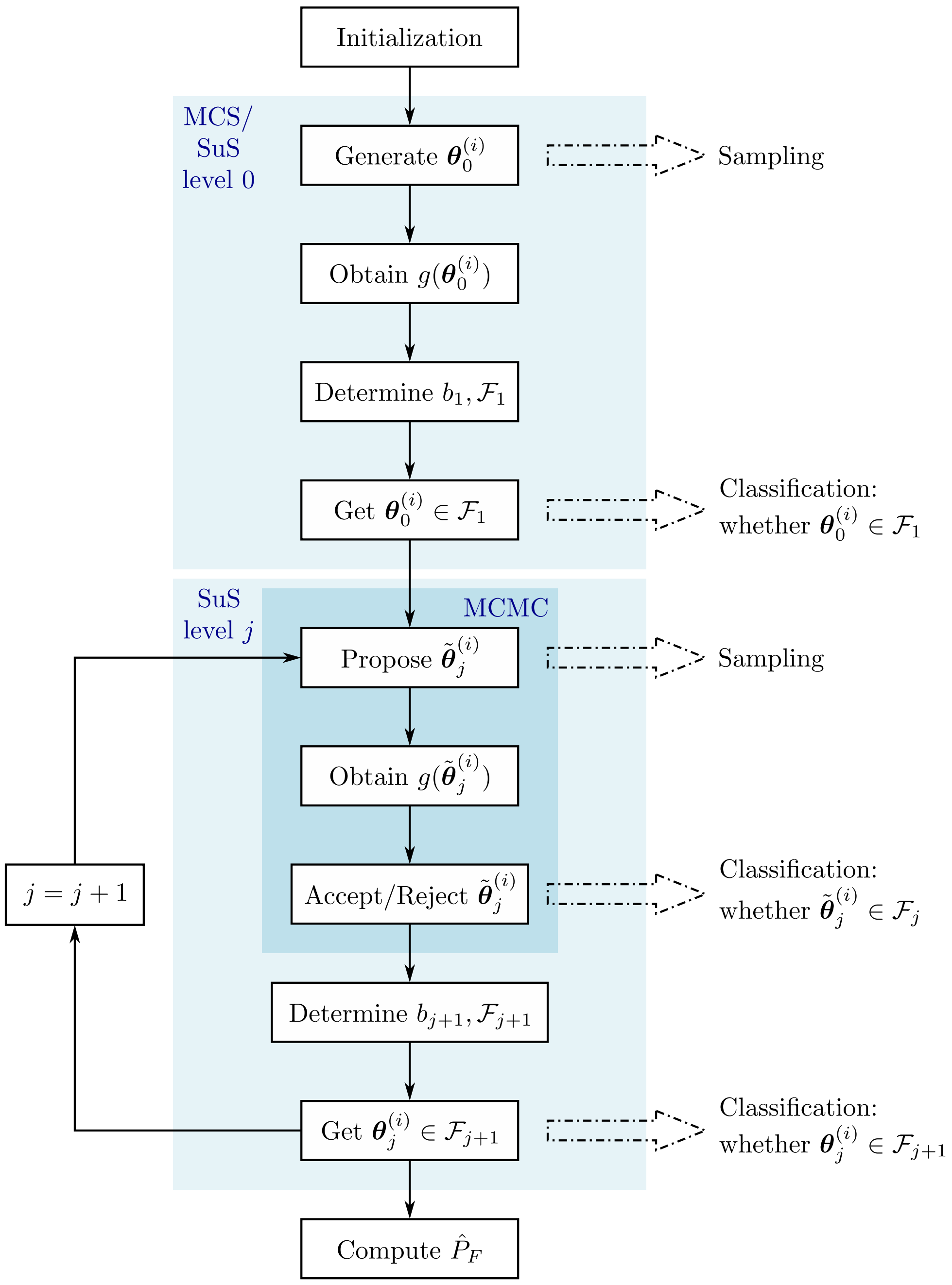
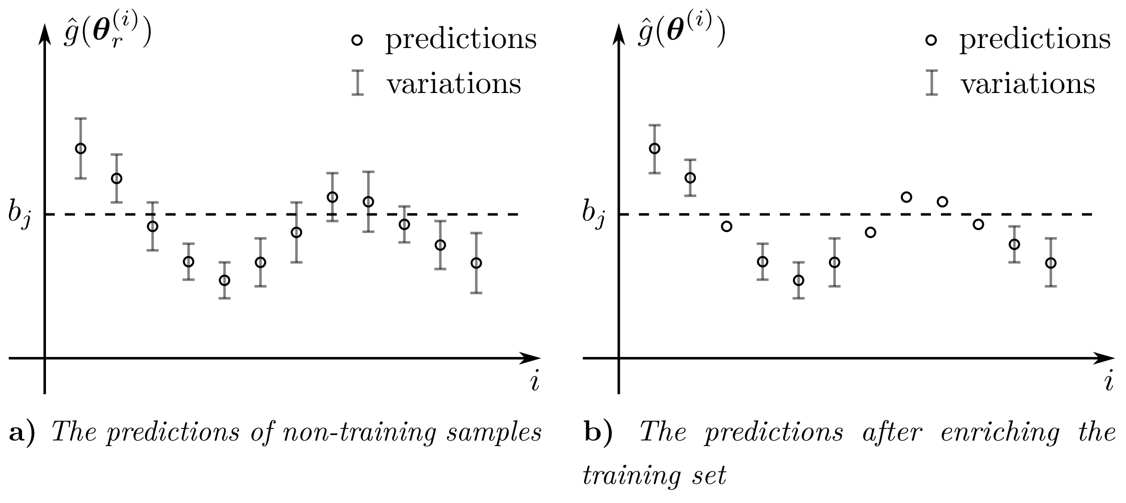

Initialization
MCS/ SuS level 0
Generate $\boldsymbol{\theta}_0^{(i)}$
Sampling
Obtain $g(\boldsymbol{\theta}_0^{(i)})$
Determine $b_1, \mathcal{F}_1$
Get $\boldsymbol{\theta}_0^{(i)} \in \mathcal{F}_1$
Classification: whether $\boldsymbol{\theta}_0^{(i)} \in \mathcal{F}_1$
SuS level j
MCMC
Propose $\tilde{\boldsymbol{\theta}}_j^{(i)}$
Sampling
Obtain $g(\tilde{\boldsymbol{\theta}}_j^{(i)})$
$j = j+1$
Accept/Reject $\tilde{\boldsymbol{\theta}}_j^{(i)}$
Classification: whether $\tilde{\boldsymbol{\theta}}_j^{(i)} \in \mathcal{F}_j$
Determine $b_{j+1}, \mathcal{F}_{j+1}$
Get $\boldsymbol{\theta}_j^{(i)} \in \mathcal{F}_{j+1}$
Classification: whether $\boldsymbol{\theta}_j^{(i)} \in \mathcal{F}_{j+1}$
Compute $\hat{P}_F$

Figure 5.7: *Interpretation of MCS and SuS.*

where $\boldsymbol{y} = [g(\boldsymbol{\theta}_t^{(1)}), \ldots, g(\boldsymbol{\theta}_t^{(n_t)})]^\mathsf{T}$ and $\boldsymbol{\varepsilon}_t = [\varepsilon_t^{(1)}, \ldots, \varepsilon_t^{(n_t)}]^\mathsf{T}$. The same basis functions are used for building these two MLS surrogate models. This indicates that the matrix $\boldsymbol{\psi}(\boldsymbol{\theta})^\mathsf{T} \left(\boldsymbol{\Psi}^\mathsf{T}\boldsymbol{W}\boldsymbol{\Psi} + \lambda \boldsymbol{I}_p\right)^{-1} \boldsymbol{\Psi}^\mathsf{T}\boldsymbol{W}$ does not need to be calculated again for the second surrogate. Consequently, the construction of the second surrogate requires almost no additional computational expense. This result holds if KRR (see Section 5.1.2) is adopted since the surrogates in this case are given by

$$\hat{g}(\boldsymbol{\theta}) = \boldsymbol{\kappa}(\boldsymbol{\theta})^\mathsf{T} \left(\boldsymbol{W}\boldsymbol{K} + \lambda \boldsymbol{I}_{n_t}\right)^{-1} \boldsymbol{W}\boldsymbol{y}, \tag{5.31}$$

$$\hat{h}(\boldsymbol{\theta}) = \boldsymbol{\kappa}(\boldsymbol{\theta})^\mathsf{T} \left(\boldsymbol{W}\boldsymbol{K} + \lambda \boldsymbol{I}_{n_t}\right)^{-1} \boldsymbol{W}\boldsymbol{\varepsilon}_t. \tag{5.32}$$



a) *The predictions of non-training samples* **b)** *The predictions after enriching the training set*

Figure 5.8: *Active learning for MLS-MCS and MLS-SuS.*

Algorithm 12 Active learning strategy for MLS-MCS and MLS-SuS

```
Input: Limit-state function $g(\boldsymbol{\theta})$; Training set $\mathcal{T} = \{(\boldsymbol{\theta}_t^{(i)}, g(\boldsymbol{\theta}_t^{(i)}))\}$; Surrogate model $\hat{g}(\boldsymbol{\theta})$;
    Non-training samples $\boldsymbol{\theta}_r^{(i)}$; Intermediate threshold $b_j$.
Output: Updated training set $\mathcal{T} = \{(\boldsymbol{\theta}_t^{(i)}, g(\boldsymbol{\theta}_t^{(i)}))\}$.
1: Compute the LOO errors of the training samples $\epsilon_t^{(i)}$ as in Equation (5.28);
2: Build a surrogate model $\hat{h}(\boldsymbol{\theta})$ using training samples $\{(\boldsymbol{\theta}_t^{(i)}, \epsilon_t^{(i)})\}$ and estimate the
   prediction errors of the non-training points $\hat{\epsilon}_r^{(i)}$;
3: Calculate the actual limit-state values $g(\boldsymbol{\theta}_r^{(i)})$ for the non-training samples satisfying
   $|\hat{g}(\boldsymbol{\theta}_r^{(i)}) - b_j| < \epsilon_r^{(i)}$ and add these samples to $\mathcal{T}$;
4: return $\mathcal{T} = \{(\boldsymbol{\theta}_t^{(i)}, g(\boldsymbol{\theta}_t^{(i)}))\}$.
```

Aiming at solving the classification problem as mentioned in Figure 5.7, it is unnecessary to guarantee that all the non-training samples are accurately predicted. The predictions of the non-training points and the variations of these predictions are depicted in Figure 5.8a. With the knowledge of the variations, for instance, the current surrogate model is confident to make a statement that the first two samples are above the boundary $\{\boldsymbol{\theta} \in \boldsymbol{\Omega} : g(\boldsymbol{\theta}) = b_j\}$. However, the third sample is predicted to lie below the boundary but it is probable to lie above instead. We select these maybe misclassified samples as additional training points to update the surrogate. After that, the predictions and variations are also updated, as presented in Figure 5.8b. The samples without variations denote the additional training points, which are evaluated using the true model and then added to the training set $\mathcal{T}$. By this means, if the prediction intervals are estimated properly, it is guaranteed that the samples are correctly classified and the introduction of the classification error is avoided. The algorithm to adaptively enrich the training set is summarized in Algorithm 12. Besides enhancing the classification accuracy, this strategy also benefits the optimization of the intermediate threshold b_j. In practice, the algorithm may be carried out several times until all the non-training samples are correctly classified.

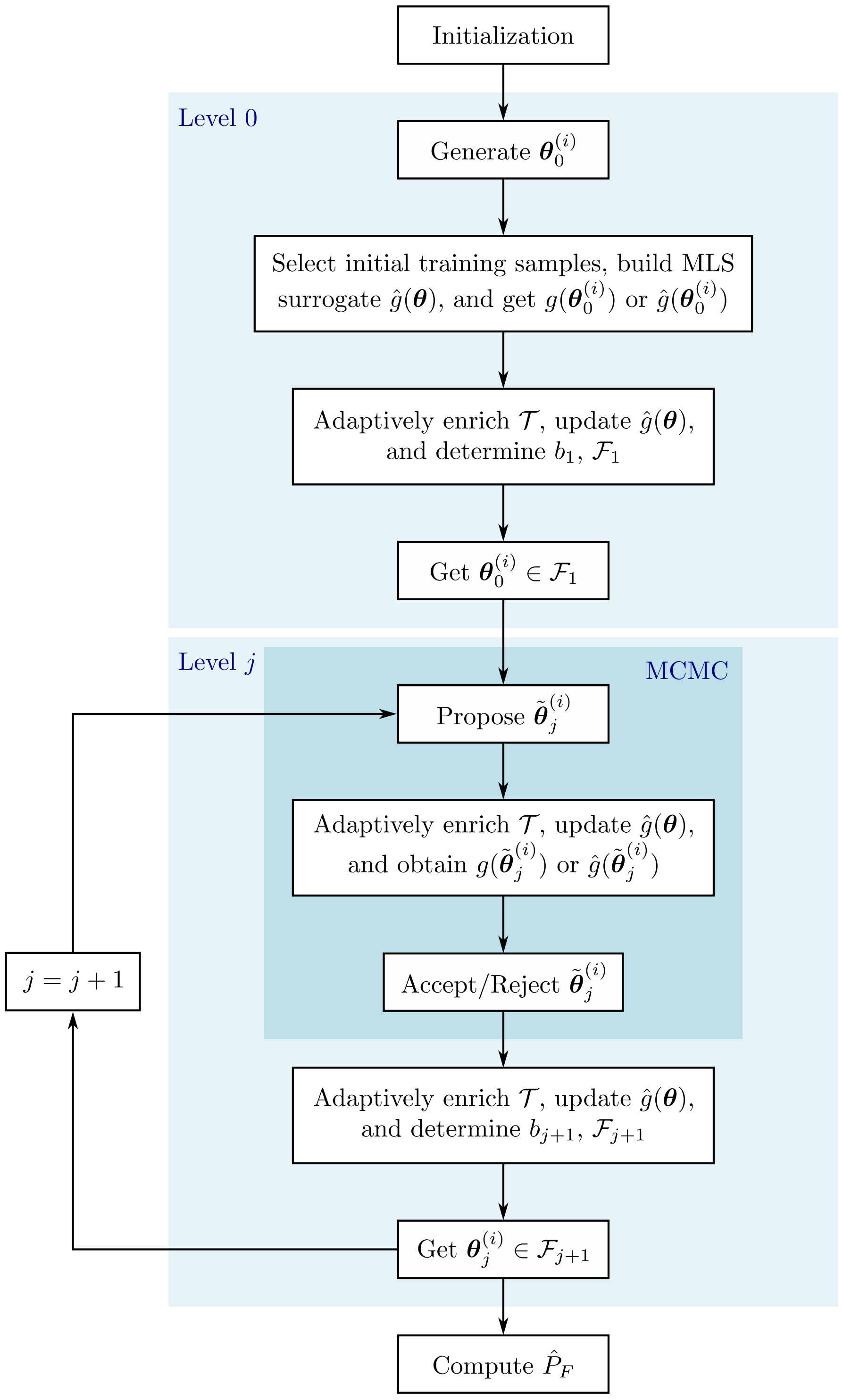

Initialization
Level 0
Generate $\boldsymbol{\theta}_0^{(i)}$
Select initial training samples, build MLS surrogate $\hat{g}(\boldsymbol{\theta})$, and get $g(\boldsymbol{\theta}_0^{(i)})$ or $\hat{g}(\boldsymbol{\theta}_0^{(i)})$
Adaptively enrich $\mathcal{T}$, update $\hat{g}(\boldsymbol{\theta})$, and determine b_1, $\mathcal{F}_1$
Get $\boldsymbol{\theta}_0^{(i)} \in \mathcal{F}_1$
Level j
MCMC
Propose $\tilde{\boldsymbol{\theta}}_j^{(i)}$
Adaptively enrich $\mathcal{T}$, update $\hat{g}(\boldsymbol{\theta})$, and obtain $g(\tilde{\boldsymbol{\theta}}_j^{(i)})$ or $\hat{g}(\tilde{\boldsymbol{\theta}}_j^{(i)})$
Accept/Reject $\tilde{\boldsymbol{\theta}}_j^{(i)}$
$j = j + 1$
Adaptively enrich $\mathcal{T}$, update $\hat{g}(\boldsymbol{\theta})$, and determine b_{j+1}, $\mathcal{F}_{j+1}$
Get $\boldsymbol{\theta}_j^{(i)} \in \mathcal{F}_{j+1}$
Compute $\hat{P}_F$

Figure 5.9: *Schema of the MLS-SuS approach.*

To summarize, the MLS surrogate is incorporated into simulation approaches to reduce the high demand of true model evaluations. But the introduction of the surrogate model brings in the classification error which may impair the estimation accuracy of the failure probability. To alleviate this side effect, an active learning strategy is proposed to improve the classification accuracy and thus the final estimation accuracy. Applying this strategy, the MLS-SuS method is shown schematically in Figure 5.9, and MLS-MCS is simply

Algorithm 13 The MLS-SuS approach

```
Input: The PDF of uncertain parameters $f(\boldsymbol{\theta})$; Limit-state function $g(\boldsymbol{\theta})$; Conditional probability $p_0$; The number of samples per subset level $N$; The percentage of points that are chosen as the initial training samples $\tilde{p}_0$.
Output: The estimate of failure probability $\hat{P}_F$.
 1: Draw $N$ i.i.d. samples $\{\boldsymbol{\theta}_0^{(i)} : i = 1, \dots, N\}$ in accordance with $f(\boldsymbol{\theta})$ and initialize an empty training set $\mathcal{T}$;
 2: Randomly select $\tilde{p}_0 N$ training samples $\{\boldsymbol{\theta}_{0,t}^{(i)} : i = 1, \dots, \tilde{p}_0 N\}$ from the $N$ samples, where $\tilde{p}_0 N$ is an integer, calculate the corresponding limit-state values $\{g(\boldsymbol{\theta}_{0,t}^{(i)}) : i = 1, \dots, \tilde{p}_0 N\}$, and add these training samples to $\mathcal{T}$;
 3: Build an MLS surrogate model $\hat{g}(\boldsymbol{\theta})$ with $\mathcal{T}$ and estimate the limit-state values of the non-training samples $\{\boldsymbol{\theta}_{0,r}^{(i)} : i = 1, \dots, (1 - \tilde{p}_0)N\}$ using $\hat{g}(\boldsymbol{\theta})$;
 4: Find $b_1$ as the $p_0$-percentile of the $N$ responses $\{g(\boldsymbol{\theta}_{0,t}^{(i)})\} \cup \{\hat{g}(\boldsymbol{\theta}_{0,r}^{(i)})\}$;
 5: Adaptively add samples that may be misclassified by the boundary $\{\boldsymbol{\theta} \in \boldsymbol{\Omega} : g(\boldsymbol{\theta}) = b_1\}$ to $\mathcal{T}$ (Algorithm 12), update $\hat{g}(\boldsymbol{\theta})$ and $b_1$, and set $\mathcal{F}_1 = \{\boldsymbol{\theta} \in \boldsymbol{\Omega} : g(\boldsymbol{\theta}) \le b_1\}$;
 6: Set $j = 1$;
 7: while $b_j > 0$ do
 8:     Consider samples $\boldsymbol{\theta}_{j-1}^{(i)} \in \mathcal{F}_j$ as seeds $\{\boldsymbol{\theta}_{j-1,s}^{(i)} : i = 1, \dots, N_s\}$, where $N_s = p_0 N$ is an integer;
 9:     Propose $N$ candidate samples $\{\tilde{\boldsymbol{\theta}}_j^{(i)} : i = 1, \dots, N\}$ from the seeds;
10:     Adaptively add samples that may be misclassified by $\{\boldsymbol{\theta} \in \boldsymbol{\Omega} : g(\boldsymbol{\theta}) = b_j\}$ to $\mathcal{T}$ (Algorithm 12), update $\hat{g}(\boldsymbol{\theta})$, and get $g(\tilde{\boldsymbol{\theta}}_j^{(i)})$ or $\hat{g}(\tilde{\boldsymbol{\theta}}_j^{(i)})$;
11:     Accept or reject $\tilde{\boldsymbol{\theta}}_j^{(i)}$ according to whether $\hat{g}(\tilde{\boldsymbol{\theta}}_j^{(i)}) \le b_j$; After rejection, the samples are denoted by $\{\boldsymbol{\theta}_j^{(i)} : i = 1, \dots, N\} = \{\boldsymbol{\theta}_{j,t}^{(i)}\} \cup \{\boldsymbol{\theta}_{j,r}^{(i)}\}$;
12:     Find $b_{j+1}$ as the $p_0$-percentile of the $N$ responses $\{g(\boldsymbol{\theta}_{j,t}^{(i)})\} \cup \{\hat{g}(\boldsymbol{\theta}_{j,r}^{(i)})\}$;
13:     Adaptively add samples that may be misclassified by $\{\boldsymbol{\theta} \in \boldsymbol{\Omega} : g(\boldsymbol{\theta}) = b_{j+1}\}$ to $\mathcal{T}$ (Algorithm 12), update $\hat{g}(\boldsymbol{\theta})$ and $b_{j+1}$, and set $\mathcal{F}_{j+1} = \{\boldsymbol{\theta} \in \boldsymbol{\Omega} : g(\boldsymbol{\theta}) \le b_{j+1}\}$;
14:     Set $j = j + 1$;
15: end while
16: Set the total number of subsets $m = j$, set $b_m = 0$, denote the number of samples $\boldsymbol{\theta}_{m-1}^{(i)} \in \mathcal{F}$ by $N_F$, and estimate the failure probability as in Equation (2.54);
17: return $\hat{P}_F$.
```

the single-level MLS-SuS. The detailed algorithm for MLS-SuS is given in Algorithm 13. Since the proposed active learning strategy seeks to eliminate the classification error, the c.o.v. approximation for the conventional SuS (see Section 2.5.3) is still employed to measure the estimation accuracy of the failure probability.

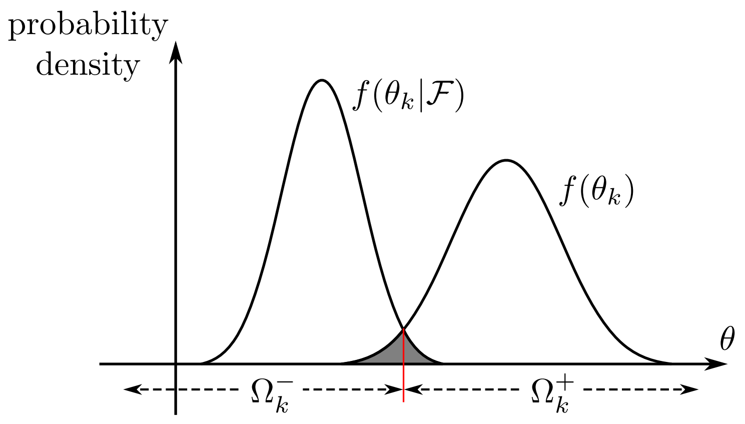


Figure 5.10: *Overlap between $f(\theta_k)$ and $f(\theta_k|\mathcal{F})$.*

5.3 Dimensionality Reduction for MLS-Based Simulation Methods

The computational expense for training an MLS surrogate grows significantly with the number of uncertain parameters. To mitigate this problem, this section reduces the dimensionality of the feature space based on the sensitivity analysis results.

5.3.1 Reliability Sensitivity Analysis

Given the MCS or SuS results, this subsection conducts global sensitivity analysis to determine parameters that are sensitive (influential) to the failure event.

In reliability analysis, the system output can be considered as a binary variable, i.e., whether the system fails or not. The effect of each input variable $\theta_k, k = 1, \dots, n$, on the system failure can be measured by the *sensitivity index* [40, 143], which is defined as the difference between the original PDF $f(\theta_k)$ and the failure-conditional PDF $f(\theta_k|\mathcal{F})$:

$$S_k = \frac{1}{2} \int_{\Omega_k} |f(\theta_k) - f(\theta_k|\mathcal{F})| \, \mathrm{d}\theta_k, \tag{5.33}$$

where Ω_k is the support of θ_k. Let $\Omega_k^+ = \{\theta_k \in \Omega_k : f(\theta_k) - f(\theta_k|\mathcal{F}) \geq 0\}$ and $\Omega_k^- = \{\theta_k \in \Omega_k : f(\theta_k) - f(\theta_k|\mathcal{F}) < 0\}$, as illustrated in Figure 5.10, the sensitivity index S_k can be represented as

$$\begin{aligned} S_k &= \frac{1}{2} \int_{\Omega_k^+} f(\theta_k) - f(\theta_k|\mathcal{F}) \, \mathrm{d}\theta_k + \frac{1}{2} \int_{\Omega_k^-} f(\theta_k|\mathcal{F}) - f(\theta_k) \, \mathrm{d}\theta_k \\ &= \frac{1}{2} \left(1 - \int_{\Omega_k} \min(f(\theta_k), f(\theta_k|\mathcal{F})) \, \mathrm{d}\theta_k \right) + \frac{1}{2} \left(1 - \int_{\Omega_k} \min(f(\theta_k), f(\theta_k|\mathcal{F})) \, \mathrm{d}\theta_k \right) \\ &= 1 - \int_{\Omega_k} \min\left(f(\theta_k), f(\theta_k|\mathcal{F})\right) \mathrm{d}\theta_k, \end{aligned} \tag{5.34}$$

where $\int_{\Omega_k} \min\left(f(\theta_k), f(\theta_k|\mathcal{F})\right) \mathrm{d}\theta_k$ is the overlapping area, i.e., the gray area shown in Figure 5.10, and $0 \leq \int_{\Omega_k} \min\left(f(\theta_k), f(\theta_k|\mathcal{F})\right) \mathrm{d}\theta_k \leq \int_{\Omega_k} f(\theta_k)\mathrm{d}\theta_k = 1$, thus the sensitivity index S_k is the non-overlapping area and $0 \leq S_k \leq 1$. The magnitude of S_k can be used to rank the parameters in the sense of the importance to the failure event. A larger S_k indicates that θ_k is more sensitive or influential to the system failure. Specifically, $S_k = 0$ means that θ_k is independent with the failure event.

For each intermediate failure domain of SuS $\mathcal{F}_j, j = 1, \ldots, m$, the sensitivity index can be generalized as

$$\begin{aligned} S_k^{(j)} &= \frac{1}{2}\int_{\Omega_k} \left|f(\theta_k) - f(\theta_k|\mathcal{F}_j)\right| \mathrm{d}\theta_k \\ &= 1 - \int_{\Omega_k} \min\left(f(\theta_k), f(\theta_k|\mathcal{F}_j)\right) \mathrm{d}\theta_k \end{aligned} \tag{5.35}$$

The properties of S_k described above still hold after the generalization.

5.3.2 Dimensionality Reduction and Hybrid Kernel

As in Equation (5.2), linear and quadratic monomials are usually chosen as basis functions for the MLS surrogate model [45]. Using all the first- and second-order combinations of variables, the number of basis functions is

$$p = \frac{1}{2}(n+1)(n+2), \tag{5.36}$$

which increases polynomially with n, namely, the dimension of uncertain parameters. To obtain sufficient regression accuracy, the number of training samples in the support domain n_t should be larger than the number of basis functions p. Hence, the number of training samples also has a polynomial growth rate. Meanwhile, training an MLS surrogate model requires repeated inversions of square matrices of order p or $n_t \in [1.2p, 2p]$, and the matrix inversion is of complexity $\mathcal{O}(p^3)$. As a consequence, the training time grows significantly as n increases. This indicates that the MLS method suffers from the curse of dimensionality.

To overcome this issue, one may use only linear basis functions instead for MLS modeling. However, the linear approximation usually sacrifices the fitting accuracy. Aiming at finding a balance between computational cost and accuracy, we implement quadratic expansions for sensitive variables but only linear approximations for insensitive ones. Motivated by the fact that some quadratic terms may only contribute little to the surrogate accuracy, this measure seeks to remove these terms with negligible impact. To be specific, if θ_k is recognized as an insensitive variable, all the quadratic items with regard to θ_k are excluded. For example, consider a 3-dimensional case and θ_1 and θ_2 are influential variables, the feature mapping can be simplified as follows:

$$\begin{aligned} \boldsymbol{\psi}(\boldsymbol{\theta}) &= \left[1, \theta_1, \theta_2, \theta_3, \theta_1\theta_2, \theta_1\theta_3, \theta_2\theta_3, \theta_1^2, \theta_2^2, \theta_3^2\right]^\mathsf{T} \\ \rightarrow \boldsymbol{\psi}(\boldsymbol{\theta}) &= \left[1, \theta_1, \theta_2, \theta_3, \theta_1\theta_2, \theta_1^2, \theta_2^2\right]^\mathsf{T}. \end{aligned} \tag{5.37}$$

By this means, compared with the second-order expansion, the dimension of the feature space is reduced, and the problem of the curse of dimensionality is thus alleviated. Though the use of hybrid basis functions may lead to the loss of accuracy in the surrogate modeling process, the active learning strategy in Section 5.2.3 is able to compensate for the classification error.

The linear and quadratic kernels for the regression with first-order and second-order expansions are given by

$$k(\boldsymbol{x}, \boldsymbol{z}) = 1 + \boldsymbol{x}^{\mathsf{T}}\boldsymbol{z}, \tag{5.38}$$

$$k(\boldsymbol{x}, \boldsymbol{z}) = \left(1 + \boldsymbol{x}^{\mathsf{T}}\boldsymbol{z}\right)^2, \tag{5.39}$$

where vectors $\boldsymbol{x}, \boldsymbol{z} \in \mathbb{R}^n$. The *hybrid kernel* for the regression with the above-mentioned simplified basis functions is defined as a combination of linear and quadratic kernels:

$$k(\boldsymbol{x}, \boldsymbol{z}) = 1 + \boldsymbol{x}^{\mathsf{T}}\boldsymbol{z} + \left(\boldsymbol{x}^{\mathsf{T}}\boldsymbol{Q}\boldsymbol{z}\right)^2, \tag{5.40}$$

where $\boldsymbol{Q} = \operatorname{diag}\{q_1, \ldots, q_n\}$ is an *indicator matrix*, wherein $q_k, k = 1, \ldots, n$, are the indicators of influential variables, which means $q_k = 1$ if the k-th variable is sensitive to the failure event and $q_k = 0$ otherwise. Let $n' = \sum_{k=1}^{n} q_k$ denote the *number of influential variables* so that $n' \leq n$, then the *number of the simplified basis functions* is calculated as

$$p' = 1 + n + \frac{1}{2}n'(n' + 1). \tag{5.41}$$

In the previous 3-dimensional example where only the first two variables are sensitive to the failure event, $\boldsymbol{Q} = \operatorname{diag}\{1, 1, 0\}$ and the corresponding kernel is

$$\begin{aligned}
k(\boldsymbol{x}, \boldsymbol{z}) &= 1 + \boldsymbol{x}^{\mathsf{T}}\boldsymbol{z} + \left(\boldsymbol{x}^{\mathsf{T}}\boldsymbol{Q}\boldsymbol{z}\right)^2 \\
&= 1 + \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} + \left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} \right)^2 \\
&= 1 + x_1 z_1 + x_2 z_2 + x_3 z_3 + 2 x_1 x_2 z_1 z_2 + x_1^2 z_1^2 + x_2^2 z_2^2 \\
&= \langle \boldsymbol{\psi}(\boldsymbol{x}), \boldsymbol{\psi}(\boldsymbol{z}) \rangle,
\end{aligned} \tag{5.42}$$

which corresponds to the feature mapping

$$\boldsymbol{\psi}(\boldsymbol{x}) = \left[1, x_1, x_2, x_3, \sqrt{2}x_1 x_2, x_1^2, x_2^2\right]^{\mathsf{T}}. \tag{5.43}$$

This result utilizes the same combinations of variables as that in Equation (5.37). Moreover, in this example, $n' = 2$ and $p' = 7$. Figure 5.11 compares the number of features for the second-order expansion (see Equation (5.36)) and that for the proposed dimensionality reduction strategy (see Equation (5.41)) assuming that $n' = \frac{1}{2}n$. With this strategy, the dimension of the feature space decreases greatly, especially for high-dimensional problems.

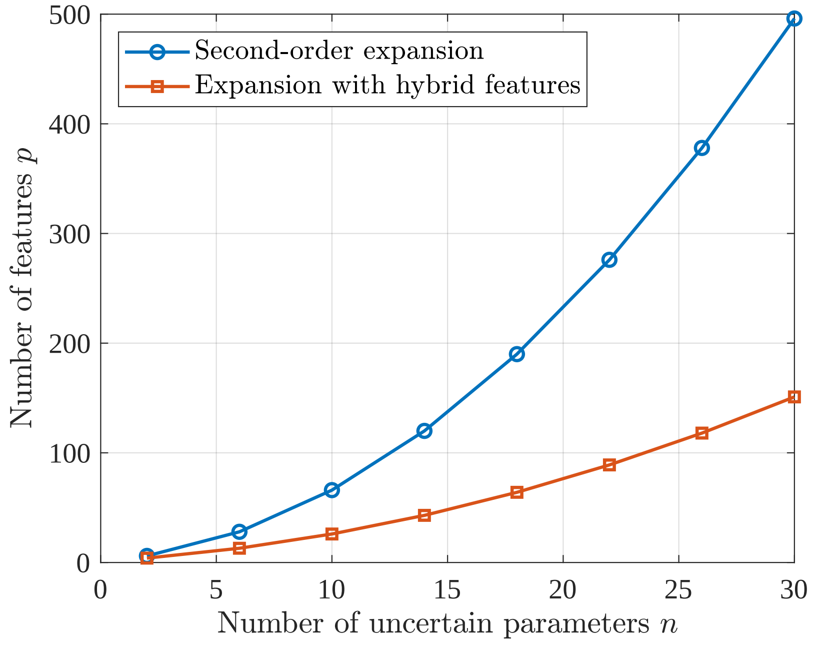


Figure 5.11: *Number of features for MLS.*

Additionally, predicting the function value of a non-training point is of complexity $\mathcal{O}(p^3)$. As a result, this dimensionality reduction strategy dramatically reduces the training time. In the case if it is necessary to update the sensitivity classification and rebuild the MLS surrogate model, the hybrid kernel is quite flexible because there is no need to recalculate the features and only the indicators q_k need to be modified.

5.3.3 Two-Stage MLS-Accelerated MCS/SuS

This subsection integrates the dimensionality reduction strategy into the proposed MLS-MCS and MLS-SuS approach in Section 5.2. Since MLS-MCS is simply a special case of MLS-SuS, this section focuses on the improvement of the MLS-SuS method.

Before performing the MLS-SuS approach which is shown in Figure 5.9, we first detect the sensitive components to determine the hybrid kernel, constituting the two-stage method summarized in Figure 5.12. The first stage aims at identifying influential variables. First of all, the SuS method is conducted with only few samples per subset level (the number of samples per level is denoted by N_0). Within this step, sparse points located in the safe region and every intermediate failure region are obtained. However, with few samples per subset level, the sensitive parameters may not be clearly distinguished [40]. Next, using these points as training samples, MLS-SuS is carried out with more samples per level but without enriching the training set. Afterwards, the sensitivity indices are evaluated and the components with larger indices are taken as influential variables. To distinguish the sensitive variables clearly, the last two steps may be repeated several times and the sensitive ones may be updated in each iteration. At the second stage, the MLS-SuS method

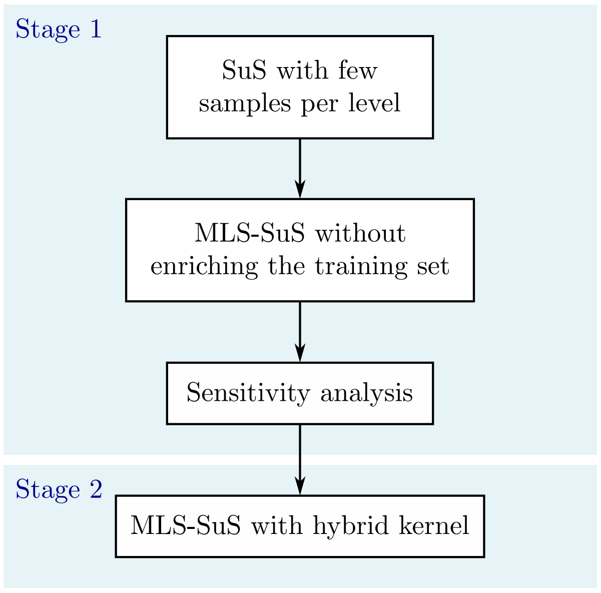

Stage 1
SuS with few samples per level
MLS-SuS without enriching the training set
Sensitivity analysis
Stage 2
MLS-SuS with hybrid kernel

Figure 5.12: *Schema of the two-stage MLS-SuS approach.*

Algorithm 14 The two-stage MLS-SuS approach

```
Input: The PDF of uncertain parameters $f(\boldsymbol{\theta})$; Limit-state function $g(\boldsymbol{\theta})$; Conditional
    probability $p_0$; The number of samples per subset level $N_0$, $N$.
Output: The estimate of failure probability $\hat{P}_F$.
 1: Conduct conventional SuS (Algorithm 3) with $N_0$ samples per subset level, and add
    all these samples into the training set $\mathcal{T}$;
 2: Initialize indicators $q_k = 0$ and $r_k = 0$, $k = 1, \dots, n$;
 3: while $\sum_{i=1}^{n} r_k < n$ do
 4:     Given $\mathcal{T}$, perform MLS-SuS ($N$ samples per level) using the hybrid kernel in
        Equation (5.40) without adding additional training samples;
 5:     Evaluate the sensitivity indices in Equation (5.33);
 6:     Set $q_k = 1$ and $r_k = 1$ if $\theta_k$ is clearly detected as a sensitive variable; Set $q_k = 0$
        and $r_k = 1$ if $\theta_k$ is clearly detected as an insensitive variable;
 7:     Set $r_k = 0$ if $\theta_k$ cannot be clearly distinguished;
 8: end while
 9: Perform the MLS-SuS approach (Algorithm 13) with $N$ samples per level using the
    hybrid kernel in Equation (5.40);
10: return $\hat{P}_F$.
```

in Figure 5.9 is performed using the determined hybrid kernel. The detailed algorithm for the two-stage MLS-SuS is given in Algorithm 14. Here, as there is not any a priori knowledge about the sensitivity ranking, the linear approximation ($q_k = 0$ for all k) is adopted in the beginning of the sensitivity analysis procedure. The influential variables are detected progressively within several iterations. The indicators r_i are introduced to mark the variables that are clearly distinguished.

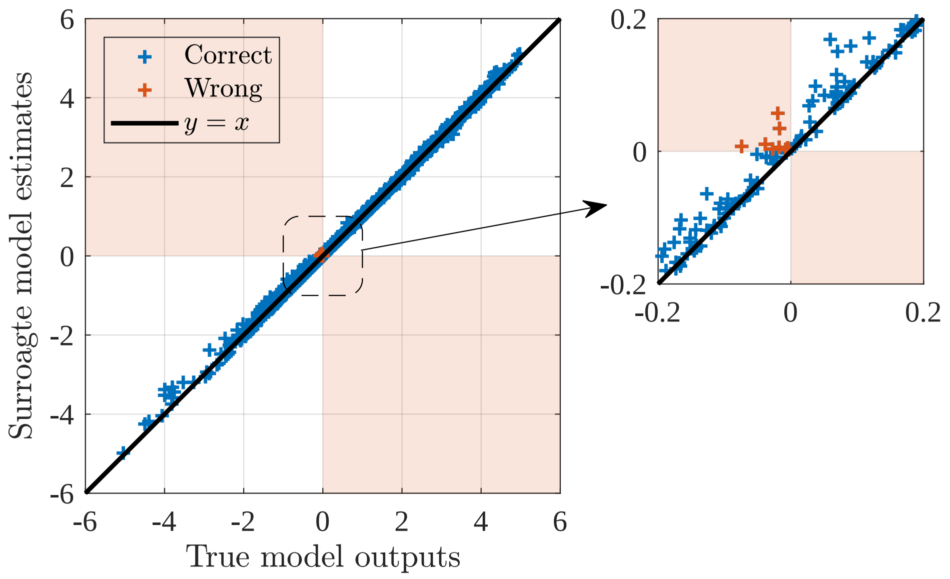


Figure 5.13: *Surrogate estimates before active learning.*

5.4 Illustrative Examples

This section presents two examples to demonstrate the accuracy and efficiency of the proposed MLS-based simulation methods.

5.4.1 Low-Dimensional Case

Consider the 3-dimensional function in Section 4.3.3:

$$g(\boldsymbol{\theta}) = a - \left(e^{0.3\theta_1+1} + e^{0.3\theta_2+1} + e^{0.3\theta_3+1}\right), \tag{5.44}$$

where $a = 10$ and $\theta_i, i = 1, 2, 3$, are independent standard normal random variables, i.e., $\theta_i \sim \mathcal{N}(0, 1)$. MLS-MCS with $N = 2000$ is implemented to assess the failure probability $\mathbb{P}\left[g(\boldsymbol{\theta}) \leq 0\right]$. The system outputs evaluated by the true model and the surrogate model are compared in Figures 5.13 and 5.14. They show the comparisons before and after the active learning process. The closer these points lie to the line $y = x$, the more accurate the estimations. These graphs thus provide an overview of the quality of the surrogates. However, from the viewpoint of classification, the surrogate model does not need to be accurate everywhere but should be able to properly classify the samples. In Figure 5.13, the points falling in the red area are misclassified (marked as "wrong" in the figure legend), whereas others are correctly classified (marked as "correct" in the legend). These misclassified samples directly influence the estimation results. The active learning strategy seeks to detect these erroneously classified samples and refine the surrogate model nearby. After conducting this procedure, as shown in Figure 5.14, all the samples are properly classified, thus avoiding the classification error caused by the surrogate error.

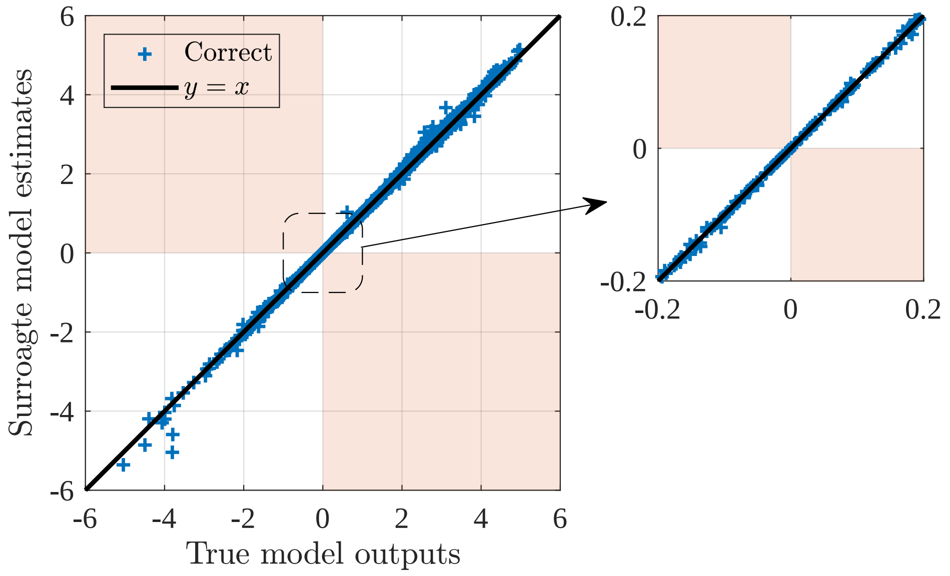


Figure 5.14: *Surrogate estimates after active learning.*

This is further shown in Table 5.2. The same 2000 samples are utilized for MCS as those for MLS-MCS. Before performing the active learning strategy, MLS-MCS results in a biased failure probability estimate, which corresponds to Figure 5.13. With the active learning step, MLS-MCS corrects the classification error and obtains the same estimate as MCS, corresponding to Figure 5.14. Indeed, this step requires more true model evaluations, but the total number of calls N_{call} is still far lower than that for MCS.

Table 5.3 shows the estimation results of 100 repeated applications of MCS and MLS-MCS. The failure probability estimates $\hat{P}_F$ and the number of calls to the true model N_{call} are the average values of the 100 results, and the c.o.v. estimates $\hat{c}_v$ are the sample

Table 5.2: *Failure probability estimation results using MCS and MLS-MCS.*

Method	$\hat{P}_F$	N_{call}
MCS	0.1600	2000
MLS-MCS without active learning	0.1555	20
MLS-MCS with active learning	0.1600	53

Table 5.3: *Failure probability estimation results using MCS and MLS-MCS with 100 runs.*

Method	$\hat{P}_F$	$\hat{c}_v$	N_{call}
MCS	0.1607	0.0323	2000
MLS-MCS without active learning	0.1598	0.0426	20
MLS-MCS with active learning	0.1601	0.0315	69

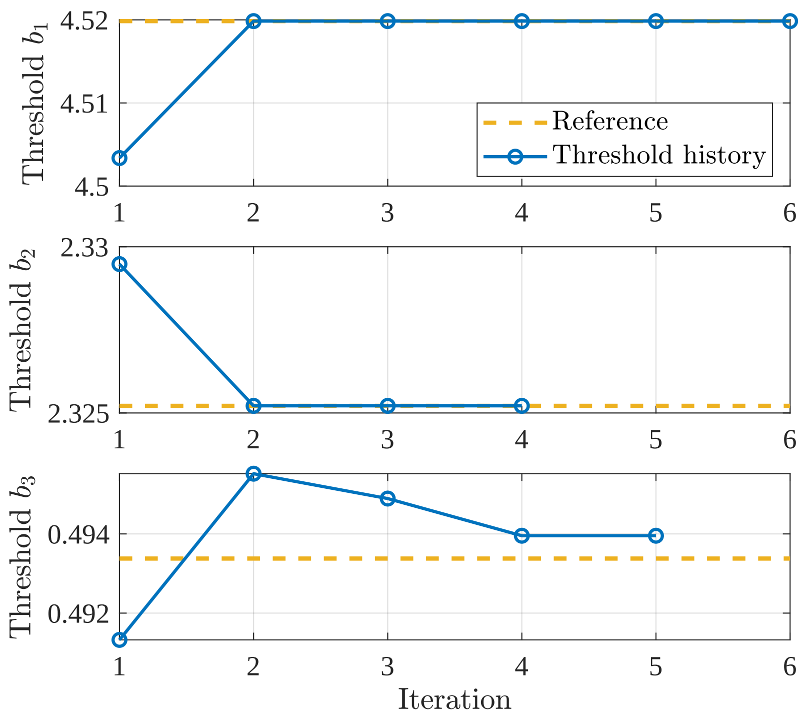


Figure 5.15: *Update history of MLS-SuS intermediate thresholds.*

c.o.v. of the 100 estimates of failure probability. The results show that MLS-MCS saves a significant number of true model evaluations compared to MCS, and the use of active learning improves the estimation accuracy of MLS-MCS to the same level as that of MCS. This demonstrates that the proposed MLS-MCS approach is able to achieve comparable estimates with MCS but requires much fewer true model evaluations.

Consider the limit-state function given in Equation (5.44) with $a = 15$. The MLS-SuS method is applied to solve the reliability analysis problem. The active learning not only refines the surrogate model around the intermediate boundary, but also adjusts the intermediate threshold, which is the p_0-percentile of the function responses at each subset level. The update history of the intermediate thresholds is given in Figure 5.15, where the reference value is the p_0-percentile of the true model outputs. It is shown that the thresholds tend to approach the exact values, thus improving the classification of samples. Figure 5.16 compares the system outputs evaluated by the true model and the surrogate model. It shows that the surrogates at all levels are sufficiently accurate to classify samples properly. Figure 5.17 and Table 5.4 present the failure probability estimation results of a single run using SuS and MLS-SuS with $N = 2000$ samples at each subset level, where σ denotes the standard deviation of the estimation approximated by Equation (2.36) with $\mu \approx \hat{P}_F$ and $c_v \approx \hat{c}_{v,\text{lb}}$. The estimation results of MLS-SuS, including the CDF, the failure probability, the c.o.v., and the 3-σ range, are quite close to those of SuS, thus demonstrating the accuracy of MLS-SuS. With the help of the MLS surrogate, the number of true model evaluations is reduced dramatically, from 7933 to 490.

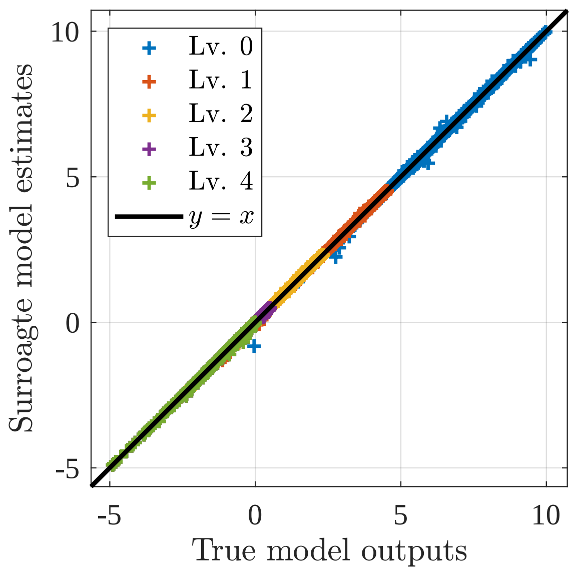


Figure 5.16: *Surrogate estimates at each level of MLS-SuS.*

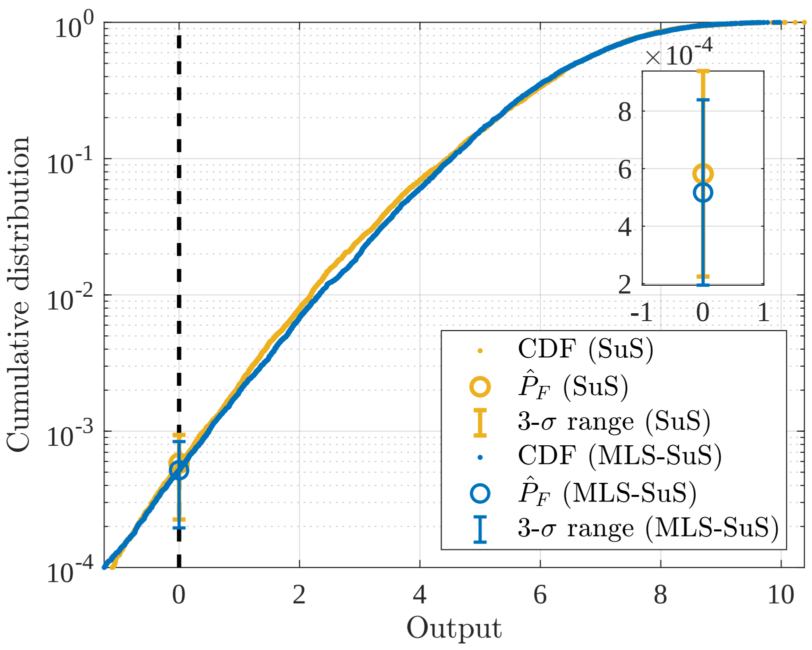


Figure 5.17: *Failure probability estimation results using SuS and MLS-SuS.*

The accuracy and efficiency of the MLS-SuS method are further illustrated in Table 5.5, where 50 simulations of SuS and MLS-SuS are conducted. The failure probability $\hat{P}_F$,

Table 5.4: *Failure probability estimation results using SuS and MLS-SuS.*

Method	$\hat{P}_F$	$[\hat{c}_{v,\mathrm{lb}}, \hat{c}_{v,\mathrm{ub}}]$	N_{call}
SuS	5.82×10^{-4}	$[0.20, 0.37]$	7933
MLS-SuS	5.17×10^{-4}	$[0.21, 0.38]$	490

Table 5.5: *Failure probability estimation results using SuS and MLS-SuS with 50 runs.*

Method	$\hat{P}_F$	$[\hat{c}_{v,\text{lb}}, \hat{c}_{v,\text{ub}}]$	$\hat{c}_v$	N_{call}
SuS	5.39×10^{-4}	$[0.20, 0.37]$	0.24	7945
MLS-SuS	5.43×10^{-4}	$[0.20, 0.37]$	0.26	446

the c.o.v. bounds $\hat{c}_{v,\text{lb}}$ and $\hat{c}_{v,\text{ub}}$, and the number of calls to the true model N_{call} are the average values of the 50 results. The c.o.v. estimate $\hat{c}_v$ is the sample c.o.v. of the 50 failure probability estimates. Again, the statistical results prove that the MLS-SuS approach is able to provide comparable estimation results at a significantly lower N_{call}.

5.4.2 High-Dimensional Case

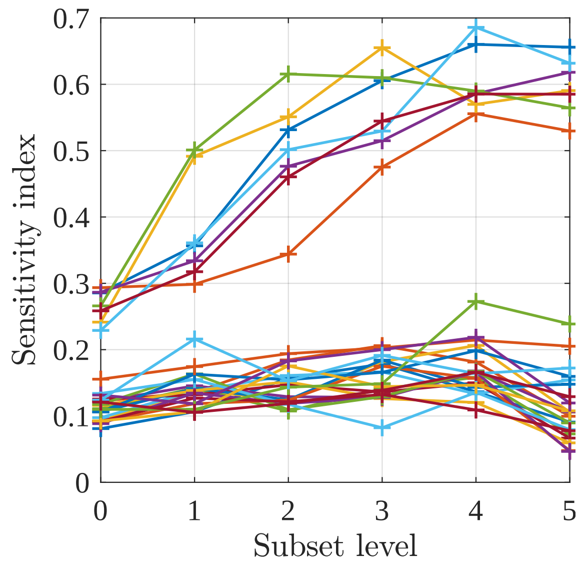
In this subsection, the benefits of the proposed dimensionality reduction strategy is exemplified. Both the direct MLS-SuS without dimensionality reduction and the two-stage MLS-SuS (with dimensionality reduction) are implemented, denoted by "MLS-SuS1" and "MLS-SuS2", respectively.

Consider the reliability analysis problem with the following 28-dimensional smooth limit-state function:

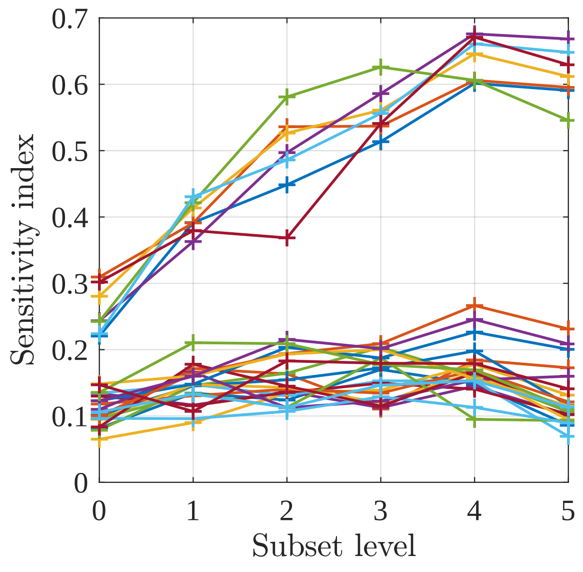
$$g(\boldsymbol{\theta}) = a - \left(\sum_{i=1}^{7} 0.1 e^{0.1\theta_i} + \sum_{i=8}^{14} 0.5 e^{0.1\theta_i} + \sum_{i=15}^{21} 2 e^{0.1\theta_i} + \sum_{i=22}^{28} 8 e^{0.1\theta_i} \right), \tag{5.45}$$

where $a = 85$ and $\theta_i, i = 1, \ldots, 28$, are i.i.d. Gaussian random variables with zero means and unit standard deviations. The variables $\theta_i, i = 22, \ldots, 28$, influence the limit-state value most because the coefficients of $e^{0.1\theta_i}, i = 22, \ldots, 28$, are the largest.

The conventional SuS and MLS-SuS2 are applied to solve this reliability problem. Figure 5.18 compares the sensitivity analysis results using both methods. The SuS results in Figure 5.18a are regarded as reference, in which the number of samples per level $N = 2000$. Figure 5.18b shows the results from the first stage of MLS-SuS2 (see Figure 5.12). Herein, $N_0 = 50$ samples per level are taken for the pure SuS in the first step. This results in 300 model evaluations in this step. In the second step, $N = 2000$ points per level are sampled for MLS-SuS. Since this step does not add additional training points, the first stage calls the true model only 300 times. In both subfigures, each line illustrates how the sensitivity index of θ_i varies with the subset levels. The 7 lines, whose corresponding sensitivity indices are significantly greater than those of the rest of lines, correspond to the last 7 variables. Therefore, these variables are identified as influential parameters by both methods. This result is consistent with our a priori knowledge. Though only 300 true model evaluations are carried out, the proposed strategy is still able to distinguish sensitive components clearly and correctly. At the second stage of MLS-SuS2, only the last



a) *SuS results*



b) *MLS-SuS results*

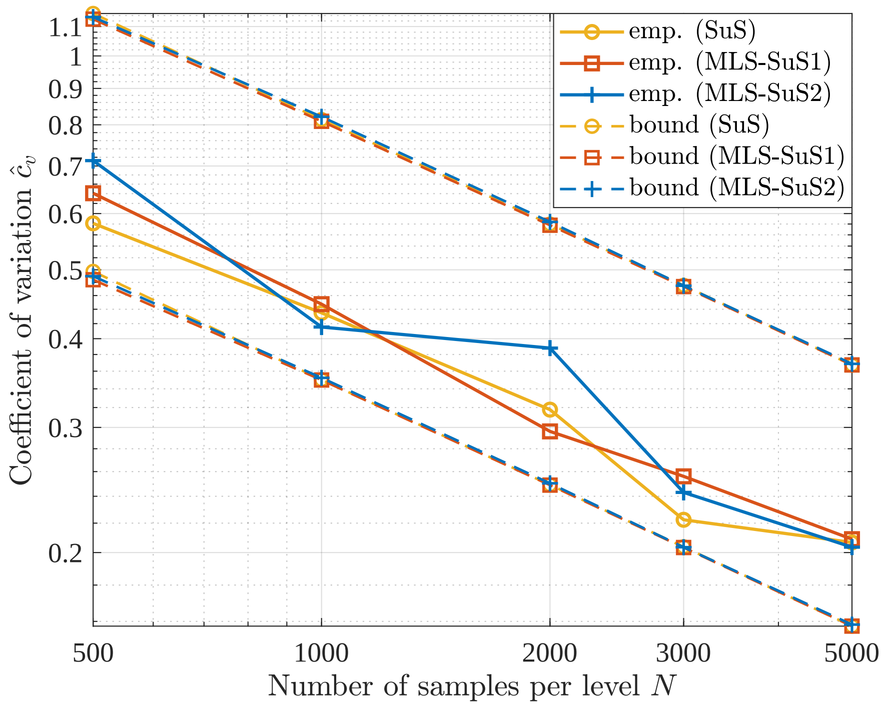
Figure 5.18: *Sensitivity index results at each level of SuS and MLS-SuS.*

7 variables are considered as sensitive components and the hybrid kernel in Equation (5.40) is implemented. This necessitates only 57 basis functions according to Equation (5.41), whereas MLS-SuS1 would need 435 basis functions.

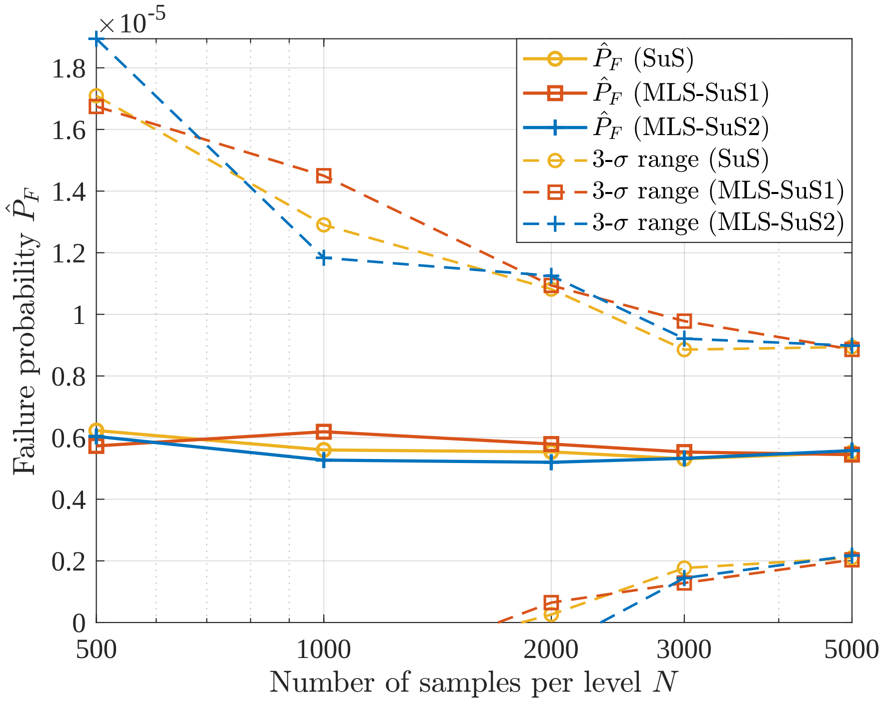
The accuracy and efficiency of the two-stage strategy are illustrated in Figures 5.19, 5.20, and 5.21. In these graphs, three different methods, SuS, MLS-SuS1, and MLS-SuS2, are conducted with different numbers of samples per level: $N = [500, 1000, 2000, 3000, 5000]^\mathsf{T}$. For each N, 50 repeated applications of these methods are performed.

Figure 5.19a shows the c.o.v. results of the failure probability estimates. In each simulation, both upper and lower bounds are evaluated as in Section 2.5.3. In this subfigure, the c.o.v. upper and lower bounds are the average values of the 50 results. The empirical c.o.v., which is represented by "emp." in the figure legend, is the sample c.o.v. of the 50 estimates of failure probability. It is shown that the three methods obtain very close c.o.v. bounds. This is because MLS-SuS1 and MLS-SuS2 employ the same c.o.v. bound estimation strategy as SuS. For empirical c.o.v., all the results are between the lower and upper bounds. Both MLS-SuS1 and MLS-SuS2 achieve comparable empirical c.o.v. results with SuS. Figure 5.19b compares the average failure probability estimates and the corresponding 3-σ ranges. The probability estimates converge as we use more samples at each subset level. The results of MLS-SuS2 are similar with those of SuS and MLS-SuS1, thus demonstrating the accuracy of the two-stage MLS-SuS approach.

Figures 5.20 and 5.21 present the number of true model evaluations and the training time of both MLS-SuS1 and MLS-SuS2. The curves show the average value of the 50 results. In Figure 5.20a, it is shown that fewer calls to the true model are required by MLS-SuS2 when N is small, but more calls are required when N is large. This is caused



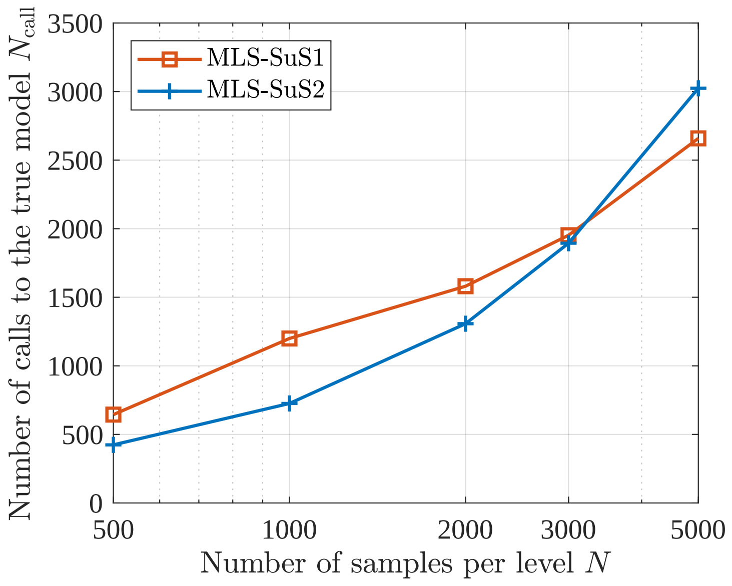
a) *Estimates of coefficients of variation*



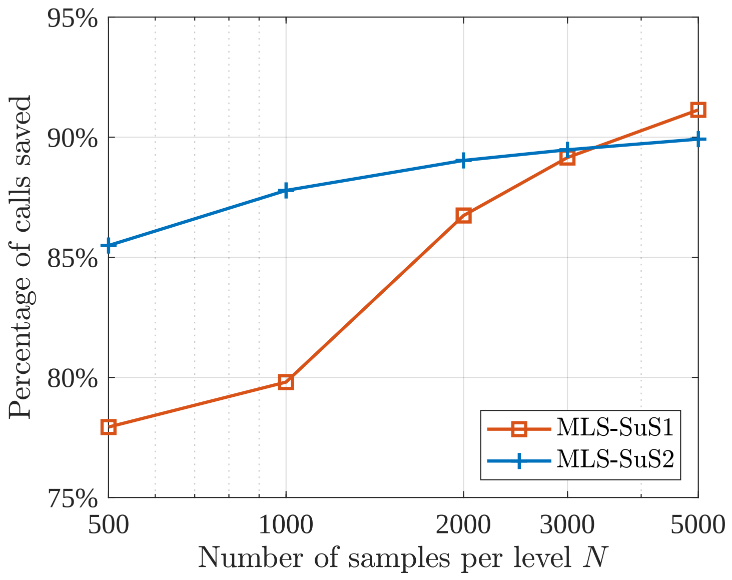
b) *Estimates of failure probabilities*

Figure 5.19: *Failure probability estimation results for SuS and MLS-SuS with different N.*

by the complexity of surrogate model. Compared with MLS-SuS2, MLS-SuS1 needs to approximate far more expansion coefficients, thus necessitates more samples to solve the regression problem. On the contrary, a more complex surrogate results in more accurate estimations, and hence MLS-SuS1 adds fewer additional samples to the training set. The



a) *Number of calls to the true model*



b) *Percentage of calls saved*

Figure 5.20: *Number of true model evaluations for MLS-SuS with different N.*

former factor plays a more important role if only few samples are available, and the latter dominates the results if there are adequate samples. With the dimension increasing, MLS-SuS2 becomes even more efficient. Figure 5.20b demonstrates that the utilization of the MLS surrogate greatly saves the required number of calls to the true model.

Figure 5.21 illustrates that MLS-SuS1 takes much (more than 10 times) longer training time than MLS-SuS2. The complex surrogate model in MLS-SuS1 leads to a high-dimensional matrix inversion problem, which indicates that this method suffers from the curse of dimensionality. However, this is significantly alleviated by the dimensionality reduction strategy of MLS-SuS2.

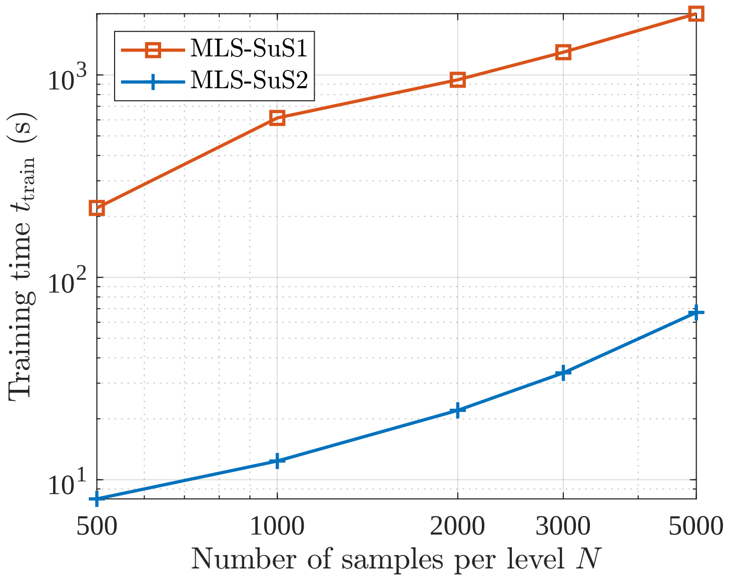


Figure 5.21: *Training time for MLS-SuS with different N.*

To sum up, compared with SuS and MLS-SuS1, the two-stage MLS-SuS is able to provide comparative estimations but with much less computational expense for high-dimensional problems.

5.5 Summary

This chapter first recalled the mathematical basics of a local surrogate model called *moving least-squares* (MLS). Then, the *MLS-accelerated SuS* (MLS-SuS) approach was proposed. Based on the approximation of surrogate errors, an active learning strategy was introduced to classify the SuS samples properly. Subsequently, a dimensionality reduction strategy for MLS-SuS was developed to filter out unnecessary expansion items. Kernel ridge regression was used to calculate the MLS coefficients in a flexible manner and the hybrid kernel tailored to the proposed dimensionality reduction strategy was designed.

Chapter 6

Kriging-Assisted Reliability Analysis

The surrogate models investigated in the previous two chapters are based on the rigid polynomial expression, thus lacking of flexibility. This chapter exploits a more flexible surrogate modeling technique, *kriging*, to accelerate the simulation-based reliability analysis approaches. The mathematical framework of kriging is presented first. The following section introduces a dimensionality reduction technique for kriging. Subsequently, the combination of the kriging surrogate and SuS is proposed. The main elements including active learning strategy, adaptive trend detection, and experimental design strategy are discussed. In the end, kriging and kriging-assisted SuS are compared with the counterparts using RSM/PCE and MLS.

6.1 Kriging

6.1.1 Kriging Model

Kriging [144, 145], also known as *Gaussian process modeling*, is a powerful surrogate model for interpolating noise-free data. It regards the model response $g(\boldsymbol{\theta})$ as a realization of a Gaussian process:

$$\hat{g}(\boldsymbol{\theta}) = \mu(\boldsymbol{\theta}) + z(\boldsymbol{\theta}), \tag{6.1}$$

where $\mu(\boldsymbol{\theta})$ is the *mean function* or the *trend*, whereas $z(\boldsymbol{\theta})$ is a *Gaussian process* with zero mean and *autocovariance* $\mathrm{Cov}[z(\boldsymbol{\theta}), z(\boldsymbol{\theta}')] = \sigma^2 R(\boldsymbol{\theta}, \boldsymbol{\theta}'; \boldsymbol{\eta})$, in which σ^2 denotes the *process variance* and $R(\boldsymbol{\theta}, \boldsymbol{\theta}'; \boldsymbol{\eta})$ denotes the *autocorrelation function* depending on the *hyperparameters* $\boldsymbol{\eta}$.

Different kriging models are distinguished in accordance with the types of the trend. *Simple kriging* assumes the trend as a known constant, i.e., $\mu(\boldsymbol{\theta}) = 0$. *Ordinary kriging* considers the trend to be an unknown constant: $\mu(\boldsymbol{\theta}) = \alpha_0$. More generally, *universal* *kriging* chooses a combination of basis functions as the trend, namely,

$$\mu(\boldsymbol{\theta}) = \sum_{k=1}^{p} \alpha_k \psi_k(\boldsymbol{\theta}) = \boldsymbol{\alpha}^\mathsf{T} \boldsymbol{\psi}(\boldsymbol{\theta}), \tag{6.2}$$

where $\boldsymbol{\psi}(\boldsymbol{\theta}) = [\psi_1(\boldsymbol{\theta}), \dots, \psi_p(\boldsymbol{\theta})]^\mathsf{T}$ and $\boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_p]^\mathsf{T}$ are a vector of *basis functions* and the corresponding *coefficients*, respectively. The linear and quadratic Taylor approximations are commonly used trends [146].

The correlation function determines the influence of the observations on the query samples. A correlation function is symmetric, i.e., $R(\boldsymbol{\theta}, \boldsymbol{\theta}'; \boldsymbol{\eta}) = R(\boldsymbol{\theta}', \boldsymbol{\theta}; \boldsymbol{\eta})$. Additionally, the *correlation matrix*, which is defined as

$$\boldsymbol{R} = \begin{bmatrix} R(\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(1)}; \boldsymbol{\eta}) & R(\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}; \boldsymbol{\eta}) & \cdots & R(\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(N)}; \boldsymbol{\eta}) \\ R(\boldsymbol{\theta}^{(2)}, \boldsymbol{\theta}^{(1)}; \boldsymbol{\eta}) & R(\boldsymbol{\theta}^{(2)}, \boldsymbol{\theta}^{(2)}; \boldsymbol{\eta}) & \cdots & R(\boldsymbol{\theta}^{(2)}, \boldsymbol{\theta}^{(N)}; \boldsymbol{\eta}) \\ \vdots & \vdots & \ddots & \vdots \\ R(\boldsymbol{\theta}^{(N)}, \boldsymbol{\theta}^{(1)}; \boldsymbol{\eta}) & R(\boldsymbol{\theta}^{(N)}, \boldsymbol{\theta}^{(2)}; \boldsymbol{\eta}) & \cdots & R(\boldsymbol{\theta}^{(N)}, \boldsymbol{\theta}^{(N)}; \boldsymbol{\eta}) \end{bmatrix}, \tag{6.3}$$

is positive semi-definite for any choice of samples $\boldsymbol{\theta}^{(i)}, i = 1, \dots, N$. A typical example is the 1-dimensional *Gaussian* or *squared exponential correlation function*:

$$R(\theta, \theta'; \eta) = \exp\left(-\eta(\theta - \theta')^2\right), \quad \eta > 0. \tag{6.4}$$

It is a *stationary* correlation function that only depends on the distance between the two points. A smaller distance corresponds to a larger correlation. Figure 6.1 depicts the Gaussian correlation function with different η and the samples drawn from a zero-mean unit-variance Gaussian process using the corresponding correlation functions. A smaller η results in greater dependence on the observations around and thus a smoother sample path. Other popular 1-dimensional stationary correlation functions are given in Appendix C.

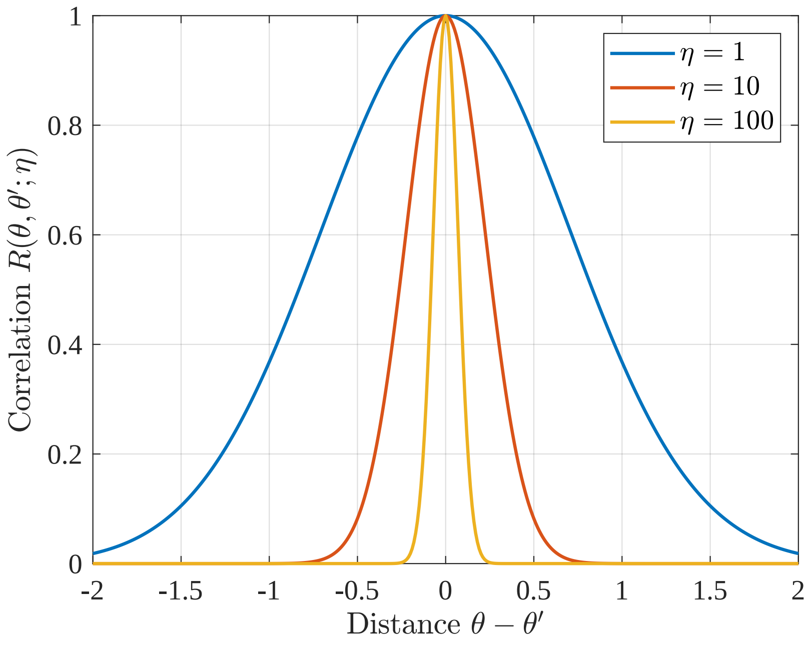
For n-dimensional problems, multi-dimensional correlation functions can be built from 1-dimensional ones. A multi-dimensional correlation function is *isotropic* if a single hyperparameter is utilized for all the dimensions. An isotropic correlation function can be constructed as

$$R(\boldsymbol{\theta}, \boldsymbol{\theta}'; \eta) = \prod_{k=1}^{n} R(\theta_k, \theta_k'; \eta), \tag{6.5}$$

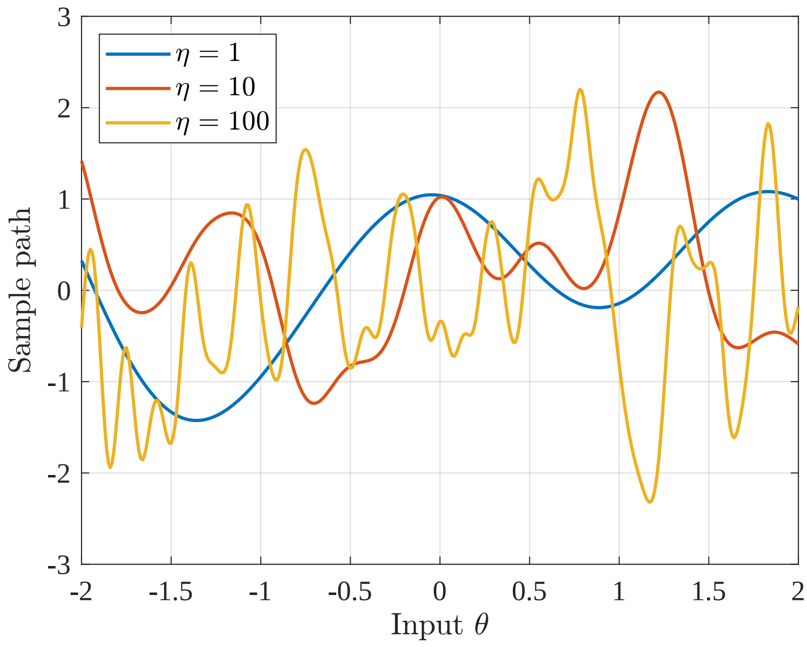
where θ_k and θ_k' are the k-th components of $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$, respectively. By contrast, a multi-dimensional correlation function is *anisotropic* if a different hyperparameter is employed in each dimension. An anisotropic correlation function can be built by

$$R(\boldsymbol{\theta}, \boldsymbol{\theta}'; \boldsymbol{\eta}) = \prod_{k=1}^{n} R(\theta_k, \theta_k'; \eta_k), \tag{6.6}$$

where η_k is the k-th component of $\boldsymbol{\eta}$. Compared with the isotropic choice, the introduction of anisotropy increases the degrees of freedom of the approximation and dramatically improves the accuracy of the surrogate model [147]. The most widely used correlation



a) *Gaussian correlation function*



b) *Sample paths*

Figure 6.1: *Gaussian correlation function and sample paths drawn from the corresponding Gaussian process.*

function is the anisotropic Gaussian correlation function given as follows:

$$R(\boldsymbol{\theta}, \boldsymbol{\theta}'; \boldsymbol{\eta}) = \prod_{k=1}^{n} \exp\left(-\eta_k(\theta_k - \theta_k')^2\right). \tag{6.7}$$

Appendix C lists several other commonly used stationary anisotropic correlation functions. In the following context, the correlation function $R(\boldsymbol{\theta}, \boldsymbol{\theta}'; \boldsymbol{\eta})$ is abbreviated as $R(\boldsymbol{\theta}, \boldsymbol{\theta}')$.

6.1.2 Estimation of Kriging Parameters

The kriging parameters $\boldsymbol{\alpha}$, σ^2, and $\boldsymbol{\eta}$ can be estimated by the *maximum likelihood estimation* (MLE) [148, Ch. 4]. Given the training set $\mathcal{T} = \{(\boldsymbol{\theta}_t^{(i)}, g(\boldsymbol{\theta}_t^{(i)})) : i = 1, \ldots, N_t\}$, the experimental matrix, the correlation matrix, and the output vector are defined, respectively, by

$$\boldsymbol{\Psi} = \begin{bmatrix} \boldsymbol{\psi}(\boldsymbol{\theta}_t^{(1)})^\mathsf{T} \\ \boldsymbol{\psi}(\boldsymbol{\theta}_t^{(2)})^\mathsf{T} \\ \vdots \\ \boldsymbol{\psi}(\boldsymbol{\theta}_t^{(N_t)})^\mathsf{T} \end{bmatrix} = \begin{bmatrix} \psi_1(\boldsymbol{\theta}_t^{(1)}) & \psi_2(\boldsymbol{\theta}_t^{(1)}) & \cdots & \psi_p(\boldsymbol{\theta}_t^{(1)}) \\ \psi_1(\boldsymbol{\theta}_t^{(2)}) & \psi_2(\boldsymbol{\theta}_t^{(2)}) & \cdots & \psi_p(\boldsymbol{\theta}_t^{(2)}) \\ \vdots & \vdots & \ddots & \vdots \\ \psi_1(\boldsymbol{\theta}_t^{(N_t)}) & \psi_2(\boldsymbol{\theta}_t^{(N_t)}) & \cdots & \psi_p(\boldsymbol{\theta}_t^{(N_t)}) \end{bmatrix}, \tag{6.8}$$

$$\boldsymbol{R} = \begin{bmatrix} R(\boldsymbol{\theta}_t^{(1)}, \boldsymbol{\theta}_t^{(1)}) & R(\boldsymbol{\theta}_t^{(1)}, \boldsymbol{\theta}_t^{(2)}) & \cdots & R(\boldsymbol{\theta}_t^{(1)}, \boldsymbol{\theta}_t^{(N_t)}) \\ R(\boldsymbol{\theta}_t^{(2)}, \boldsymbol{\theta}_t^{(1)}) & R(\boldsymbol{\theta}_t^{(2)}, \boldsymbol{\theta}_t^{(2)}) & \cdots & R(\boldsymbol{\theta}_t^{(2)}, \boldsymbol{\theta}_t^{(N_t)}) \\ \vdots & \vdots & \ddots & \vdots \\ R(\boldsymbol{\theta}_t^{(N_t)}, \boldsymbol{\theta}_t^{(1)}) & R(\boldsymbol{\theta}_t^{(N_t)}, \boldsymbol{\theta}_t^{(2)}) & \cdots & R(\boldsymbol{\theta}_t^{(N_t)}, \boldsymbol{\theta}_t^{(N_t)}) \end{bmatrix}, \tag{6.9}$$

and

$$\boldsymbol{y} = [g(\boldsymbol{\theta}_t^{(1)}), g(\boldsymbol{\theta}_t^{(2)}), \ldots, g(\boldsymbol{\theta}_t^{(N_t)})]^\mathsf{T}. \tag{6.10}$$

Since $\boldsymbol{y} - \boldsymbol{\Psi\alpha}$ is assumed to be a correlated zero-mean Gaussian vector, the *marginal likelihood function* is given by

$$\mathcal{L}(\boldsymbol{\alpha}, \sigma^2, \boldsymbol{\eta}; \boldsymbol{y}) = \frac{1}{\sqrt{(2\pi\sigma^2)^{N_t}|\boldsymbol{R}|}} \exp\left(-\frac{1}{2\sigma^2}(\boldsymbol{y} - \boldsymbol{\Psi\alpha})^\mathsf{T}\boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{\Psi\alpha})\right). \tag{6.11}$$

The natural logarithm of the likelihood is

$$\ln \mathcal{L}(\boldsymbol{\alpha}, \sigma^2, \boldsymbol{\eta}; \boldsymbol{y}) = -\frac{N_t}{2}\ln(2\pi) - \frac{N_t}{2}\ln(\sigma^2) - \frac{1}{2}\ln(|\boldsymbol{R}|) - \frac{1}{2\sigma^2}(\boldsymbol{y} - \boldsymbol{\Psi\alpha})^\mathsf{T}\boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{\Psi\alpha}). \tag{6.12}$$

By maximizing Equation (6.12), one gets the optimal estimates of $\boldsymbol{\alpha}$ and σ^2:

$$\hat{\boldsymbol{\alpha}} = \left(\boldsymbol{\Psi}^\mathsf{T}\boldsymbol{R}^{-1}\boldsymbol{\Psi}\right)^{-1}\boldsymbol{\Psi}^\mathsf{T}\boldsymbol{R}^{-1}\boldsymbol{y}, \tag{6.13}$$

$$\hat{\sigma}^2 = \frac{1}{N_t}(\boldsymbol{y} - \boldsymbol{\Psi\alpha})^\mathsf{T}\boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{\Psi\alpha}). \tag{6.14}$$

Substituting Equations (6.13) and (6.14) into Equation (6.12), the log marginal likelihood $\ln \mathcal{L}(\boldsymbol{\alpha}, \sigma^2, \boldsymbol{\eta}; \boldsymbol{y})$ boils down to

$$\ln \mathcal{L}(\boldsymbol{\eta}; \boldsymbol{y}) = -\frac{N_t}{2}(\ln(2\pi) + 1) - \frac{N_t}{2}\ln(\hat{\sigma}^2) - \frac{1}{2}\ln(|\boldsymbol{R}|). \tag{6.15}$$

Here, the first term is a constant, the second term represents the quality of the trend, and the third term is a complexity penalty. Therefore, this log likelihood is a tradeoff between flexibility and accuracy. The hyperparameters $\boldsymbol{\eta}$ are then obtained by maximizing $\ln \mathcal{L}(\boldsymbol{\eta}; \boldsymbol{y})$:

$$\hat{\boldsymbol{\eta}} = \arg\max_{\boldsymbol{\eta}} \ln \mathcal{L}(\boldsymbol{\eta}; \boldsymbol{y}). \tag{6.16}$$

6.1.3 Kriging Predictor

For any query point $\boldsymbol{\theta}$, based on the Gaussian assumption, the response vector $[\boldsymbol{y}^\mathsf{T}, \hat{g}(\boldsymbol{\theta})]^\mathsf{T}$ is Gaussian-distributed:

$$\begin{bmatrix} \boldsymbol{y} \\ \hat{g}(\boldsymbol{\theta}) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\Psi}\boldsymbol{\alpha} \\ \boldsymbol{\psi}(\boldsymbol{\theta})^\mathsf{T}\boldsymbol{\alpha} \end{bmatrix}, \sigma^2 \begin{bmatrix} \boldsymbol{R} & \boldsymbol{r}(\boldsymbol{\theta}) \\ \boldsymbol{r}(\boldsymbol{\theta})^\mathsf{T} & 1 \end{bmatrix} \right), \tag{6.17}$$

where $\boldsymbol{r}(\boldsymbol{\theta}) = [R(\boldsymbol{\theta}, \boldsymbol{\theta}_t^{(1)}), \cdots, R(\boldsymbol{\theta}, \boldsymbol{\theta}_t^{(N_t)})]^\mathsf{T}$ gathering the correlations between the query point and the training samples. The variable $\hat{g}(\boldsymbol{\theta})$ conditional on the training data follows the Gaussian distribution with *mean* $m_{\hat{g}}(\boldsymbol{\theta})$ and *variance* $s^2_{\hat{g}}(\boldsymbol{\theta})$ [144, Ch. 3]:

$$\hat{g}(\boldsymbol{\theta}) \sim \mathcal{N}\left(m_{\hat{g}}(\boldsymbol{\theta}), s^2_{\hat{g}}(\boldsymbol{\theta})\right), \tag{6.18}$$

where

$$m_{\hat{g}}(\boldsymbol{\theta}) = \boldsymbol{\psi}(\boldsymbol{\theta})^\mathsf{T}\hat{\boldsymbol{\alpha}} + \boldsymbol{r}(\boldsymbol{\theta})^\mathsf{T}\boldsymbol{R}^{-1}\left(\boldsymbol{y} - \boldsymbol{\Psi}\hat{\boldsymbol{\alpha}}\right), \tag{6.19}$$

$$s^2_{\hat{g}}(\boldsymbol{\theta}) = \sigma^2\left(1 - \boldsymbol{r}(\boldsymbol{\theta})^\mathsf{T}\boldsymbol{R}^{-1}\boldsymbol{r}(\boldsymbol{\theta}) + \boldsymbol{u}(\boldsymbol{\theta})^\mathsf{T}\left(\boldsymbol{\Psi}^\mathsf{T}\boldsymbol{R}^{-1}\boldsymbol{\Psi}\right)^{-1}\boldsymbol{u}(\boldsymbol{\theta})\right), \tag{6.20}$$

in which

$$\boldsymbol{u}(\boldsymbol{\theta}) = \boldsymbol{\Psi}^\mathsf{T}\boldsymbol{R}^{-1}\boldsymbol{r}(\boldsymbol{\theta}) - \boldsymbol{\psi}(\boldsymbol{\theta}). \tag{6.21}$$

This predictor has been proved to be the *best linear unbiased predictor* (BLUP) in [144, Ch. 3]. The *confidence interval* of the prediction is given by

$$\hat{g}(\boldsymbol{\theta}) \in \left[m_{\hat{g}}(\boldsymbol{\theta}) + \Phi^{-1}\left(\frac{\alpha}{2}\right)s_{\hat{g}}(\boldsymbol{\theta}), m_{\hat{g}}(\boldsymbol{\theta}) + \Phi^{-1}\left(1 - \frac{\alpha}{2}\right)s_{\hat{g}}(\boldsymbol{\theta})\right] \tag{6.22}$$

with a *confidence level* $1 - \alpha$.

Consider the case $\boldsymbol{\theta} = \boldsymbol{\theta}_t^{(i)}, i = 1, \ldots, N_t$, then $\boldsymbol{r}(\boldsymbol{\theta})$ is the i-th column of $\boldsymbol{R}$. Therefore, one has

$$\boldsymbol{R}^{-1}\boldsymbol{r}(\boldsymbol{\theta}_t^{(i)}) = \boldsymbol{e}_i, \tag{6.23}$$

$$\boldsymbol{r}(\boldsymbol{\theta}_t^{(i)})^\mathsf{T}\boldsymbol{R}^{-1} = \boldsymbol{e}_i^\mathsf{T}, \tag{6.24}$$

where $\boldsymbol{e}_i$ is an unit column vector with a 1 at the i-th position. Consequently,

$$\begin{aligned} m_{\hat{g}}(\boldsymbol{\theta}_t^{(i)}) &= \boldsymbol{\psi}(\boldsymbol{\theta}_t^{(i)})^\mathsf{T}\hat{\boldsymbol{\alpha}} + \boldsymbol{r}(\boldsymbol{\theta}_t^{(i)})^\mathsf{T}\boldsymbol{R}^{-1}\left(\boldsymbol{y} - \boldsymbol{\Psi}\hat{\boldsymbol{\alpha}}\right) \\ &= \boldsymbol{\psi}(\boldsymbol{\theta}_t^{(i)})^\mathsf{T}\hat{\boldsymbol{\alpha}} + \boldsymbol{e}_i^\mathsf{T}\left(\boldsymbol{y} - \boldsymbol{\Psi}\hat{\boldsymbol{\alpha}}\right) \\ &= g(\boldsymbol{\theta}_t^{(i)}), \end{aligned} \tag{6.25}$$

$$\boldsymbol{u}(\boldsymbol{\theta}_t^{(i)}) = \boldsymbol{\Psi}^\mathsf{T}\boldsymbol{R}^{-1}\boldsymbol{r}(\boldsymbol{\theta}_t^{(i)}) - \boldsymbol{\psi}(\boldsymbol{\theta}_t^{(i)}) = \boldsymbol{\Psi}^\mathsf{T}\boldsymbol{e}_i - \boldsymbol{\psi}(\boldsymbol{\theta}_t^{(i)}) = \boldsymbol{0}, \tag{6.26}$$

$$\begin{aligned} s^2_{\hat{g}}(\boldsymbol{\theta}_t^{(i)}) &= \sigma^2\left(1 - \boldsymbol{r}(\boldsymbol{\theta}_t^{(i)})^\mathsf{T}\boldsymbol{R}^{-1}\boldsymbol{r}(\boldsymbol{\theta}_t^{(i)}) + \boldsymbol{u}(\boldsymbol{\theta}_t^{(i)})^\mathsf{T}\left(\boldsymbol{\Psi}^\mathsf{T}\boldsymbol{R}^{-1}\boldsymbol{\Psi}\right)^{-1}\boldsymbol{u}(\boldsymbol{\theta}_t^{(i)})\right) \\ &= \sigma^2\left(1 - \boldsymbol{e}_i^\mathsf{T}\boldsymbol{r}(\boldsymbol{\theta}_t^{(i)})\right) \\ &= \sigma^2\left(1 - R(\boldsymbol{\theta}_t^{(i)}, \boldsymbol{\theta}_t^{(i)})\right) \\ &= 0. \end{aligned} \tag{6.27}$$

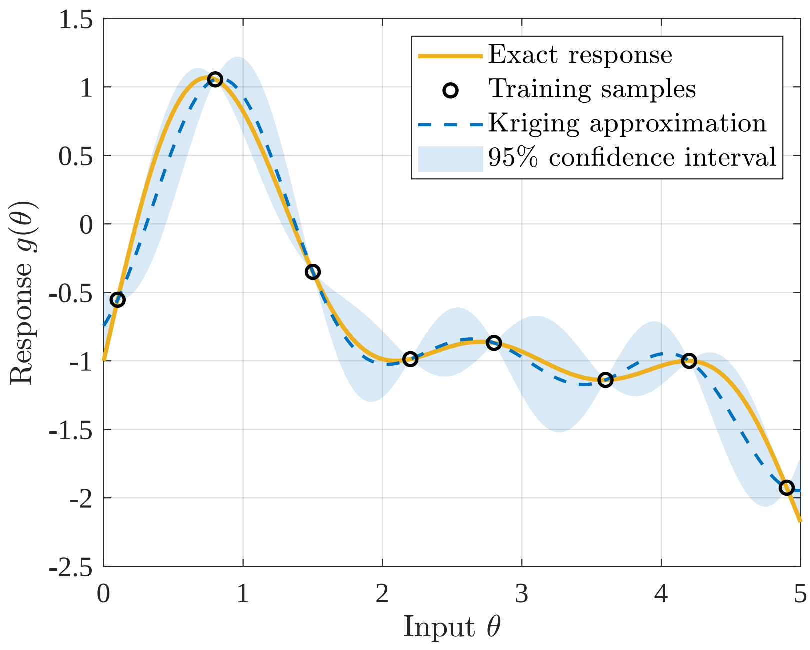


Figure 6.2: *Kriging approximation.*

This means

$$\hat{g}(\boldsymbol{\theta}_t^{(i)}) = g(\boldsymbol{\theta}_t^{(i)}), \tag{6.28}$$

thus proving that kriging is an interpolation method.

A kriging approximation example is presented in Figure 6.2. Here, the 1-dimensional true model is given by

$$g(\theta) = \sin(\theta) + \sin(2\theta) + 0.5\sin(3\theta) - 1. \tag{6.29}$$

Simple kriging is applied with $\mu(\theta) = 0$, $\sigma^2 = 1$, and $\eta = 1.5$. Note that the parameters are prescribed and not the best choices as given in Section 6.1.2. The blue dashed line represents the mean value estimated by Equation (6.19) and the shaded area denotes the 95% confidence interval calculated by Equation (6.22). It is also shown that the predictions are exact at the training samples.

6.1.4 Modeling Steps

The steps for kriging modeling are summarized in Figure 6.3. First, users should choose the basis functions $\boldsymbol{\psi}(\boldsymbol{\theta})$ for the trend and the correlation function $R(\boldsymbol{\theta}, \boldsymbol{\theta}'; \boldsymbol{\eta})$ (see Section 6.1.1). Meanwhile, the training set $\mathcal{T}$ should be determined as well. Next, the hyperparameters $\boldsymbol{\eta}$ are estimated by maximizing the likelihood function in Equation (6.15), after which the remaining parameters $\boldsymbol{\alpha}$ and σ^2 are calculated based on Equations (6.13) and (6.14). In the end, the prediction $m_{\hat{g}}(\boldsymbol{\theta})$ and the estimation variance $s_{\hat{g}}^2(\boldsymbol{\theta})$ are obtained according to Equations (6.19) and (6.20).

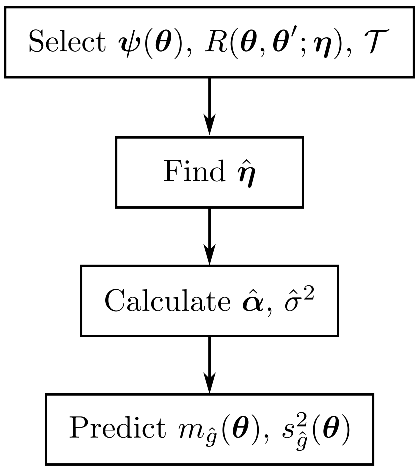

Select $\boldsymbol{\psi}(\boldsymbol{\theta})$, $R(\boldsymbol{\theta}, \boldsymbol{\theta}'; \boldsymbol{\eta})$, $\mathcal{T}$
Find $\hat{\boldsymbol{\eta}}$
Calculate $\hat{\boldsymbol{\alpha}}$, $\hat{\sigma}^2$
Predict $m_{\hat{g}}(\boldsymbol{\theta})$, $s^2_{\hat{g}}(\boldsymbol{\theta})$

Figure 6.3: *Kriging modeling steps.*

6.1.5 Benefits and Drawbacks

The most significant advantage of kriging is its built-in estimate of the prediction variance, which indicates the quality of the prediction. Kriging consists of a trend and a Gaussian process, thus capturing both the global characteristic and the local variability. As a consequence, it is flexible in accurately imitating the behaviors of limit-state functions. Additionally, the best linear unbiased predictor provides interpolation results and is asymptotically consistent when the actual correlation of the Gaussian process is regular [149].

Because of these advantages, kriging has become increasingly popular. However, it suffers from the curse of dimensionality. This is caused by two main reasons. On the one hand, the optimization of hyperparameters requires inverting the correlation matrix of size $N_t \times N_t$ (see Equations (6.14), (6.15), and (6.16)), where N_t is the *number of training samples*. The matrix inversion is of complexity $\mathcal{O}(N_t^3)$. As a result, the computational cost grows dramatically with the increase of training samples. Moreover, a large training set is usually necessary for complex or high-dimensional problems to build a sufficiently accurate surrogate model. Therefore, it is computationally intensive to achieve the matrix inversion task for such cases. On the other hand, the number of hyperparameters to be estimated increases as the dimension grows (e.g., the correlation function in Equation (6.7)). The time complexity for training a kriging model is $\mathcal{O}(N_t^3 N_p N_{\text{iter}})$ [150], where N_p is the *number of hyperparameters* and N_{iter} is the *number of iterations for optimizing the hyperparameters.* A problem with a larger N_p means a greater search space and requires a larger N_{iter}. Thus, the kriging modeling becomes even more computationally expensive in higher dimensions.

To overcome this challenge, one can implement two feasible strategies: reducing the number of hyperparameters and reducing the number of training samples without sacrificing the modeling accuracy. In this thesis, these strategies are accomplished in Section 6.2 and Section 6.3.3, respectively.

6.2 Partial Least-Squares Dimensionality Reduction

To alleviate the problem of the curse of dimensionality, this section employs *partial least-squares* (PLS) to reduce the number of unknown hyperparameters, namely, the dimensionality of the optimization search space.

6.2.1 Partial Least-Squares (PLS)

The PLS approach [151] projects the input variables onto a new space aiming at finding the fundamental relations between the input variables and the responses. The new space is established by selected *latent variables* or *principal components*.

Suppose that the $N \times n$ *sample matrix* $\mathbf{\Phi} = [\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(N)}]^\mathsf{T}$ and the $N \times 1$ *response vector* $\boldsymbol{y} = [g(\boldsymbol{\theta}^{(1)}), \ldots, g(\boldsymbol{\theta}^{(N)})]^\mathsf{T}$ are mean-centered and properly scaled (e.g., normalized). The first principal component is obtained as

$$\boldsymbol{\vartheta}_1 = \mathbf{\Phi}\boldsymbol{w}_1, \tag{6.30}$$

where $\boldsymbol{w}_1$ is the best *weight vector* of size $n \times 1$ maximizing the covariance between $\boldsymbol{\vartheta}_1$ and $\boldsymbol{y}$:

$$\boldsymbol{w}_1 = \arg \max_{\boldsymbol{w}:\|\boldsymbol{w}\|=1} \mathrm{Cov}[\boldsymbol{y}^\mathsf{T}\mathbf{\Phi}\boldsymbol{w}, \boldsymbol{y}^\mathsf{T}\mathbf{\Phi}\boldsymbol{w}]. \tag{6.31}$$

The exact solution is given by

$$\boldsymbol{w}_1 = \frac{\mathbf{\Phi}^\mathsf{T}\boldsymbol{y}}{\|\mathbf{\Phi}^\mathsf{T}\boldsymbol{y}\|}. \tag{6.32}$$

Afterwards, the residuals of sample matrix and response vector are denoted, respectively, by

$$\begin{aligned} \mathbf{\Phi}_1 &= \mathbf{\Phi} - \boldsymbol{\vartheta}_1\boldsymbol{\alpha}_1^\mathsf{T}, \\ \boldsymbol{y}_1 &= \boldsymbol{y} - c_1\boldsymbol{\vartheta}_1, \end{aligned} \tag{6.33}$$

where $\boldsymbol{\alpha}_1$ is an $n \times 1$ vector containing the *projection coefficients* of $\mathbf{\Phi}$ on the first principal component $\boldsymbol{\vartheta}_1$, and c_1 represents the *regression coefficient* of $\boldsymbol{y}$ on $\boldsymbol{\vartheta}_1$. They are evaluated by

$$\begin{aligned} \boldsymbol{\alpha}_1 &= \frac{\mathbf{\Phi}^\mathsf{T}\boldsymbol{\vartheta}_1}{\boldsymbol{\vartheta}_1^\mathsf{T}\boldsymbol{\vartheta}_1}, \\ c_1 &= \frac{\boldsymbol{\vartheta}_1^\mathsf{T}\boldsymbol{y}}{\boldsymbol{\vartheta}_1^\mathsf{T}\boldsymbol{\vartheta}_1}. \end{aligned} \tag{6.34}$$

Similarly, this procedure can be continued to extract more principal components $\boldsymbol{\vartheta}_j$ from the residuals $\mathbf{\Phi}_{j-1}$ and $\boldsymbol{y}_{j-1}$, $j = 2, \ldots, m$:

$$\begin{aligned} \boldsymbol{\vartheta}_j &= \mathbf{\Phi}_{j-1}\boldsymbol{w}_j, \\ \mathbf{\Phi}_j &= \mathbf{\Phi}_{j-1} - \boldsymbol{\vartheta}_j\boldsymbol{\alpha}_j^\mathsf{T}, \\ \boldsymbol{y}_j &= \boldsymbol{y}_{j-1} - c_j\boldsymbol{\vartheta}_j. \end{aligned} \tag{6.35}$$

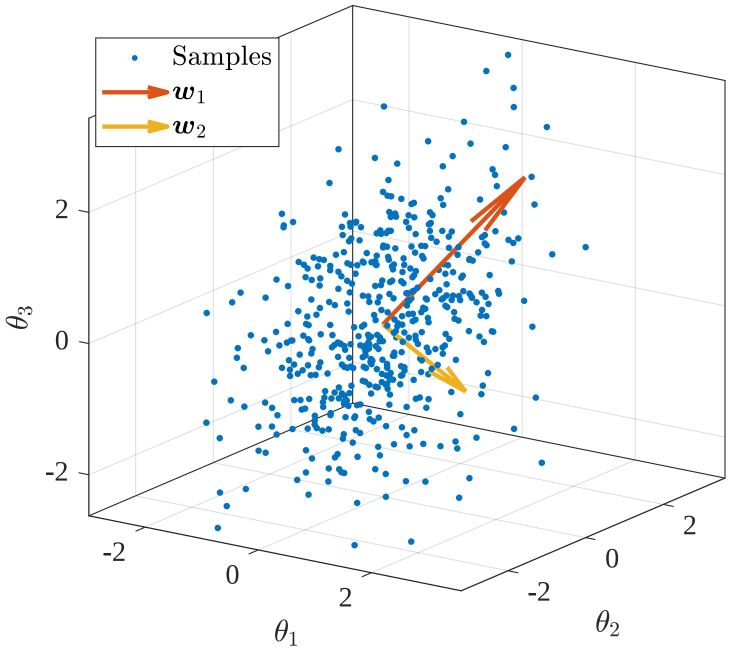
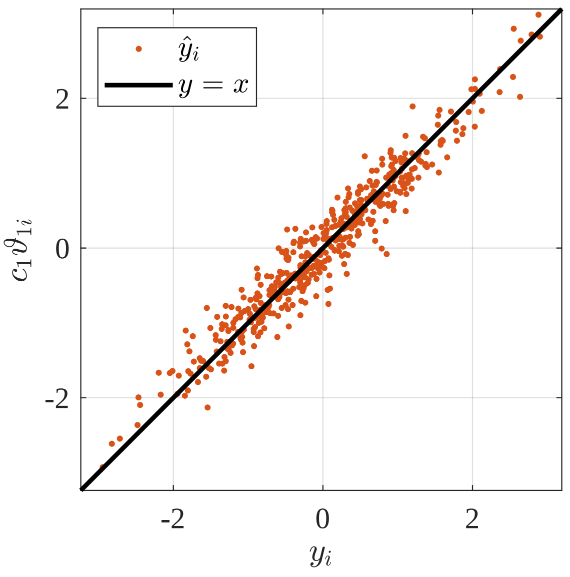
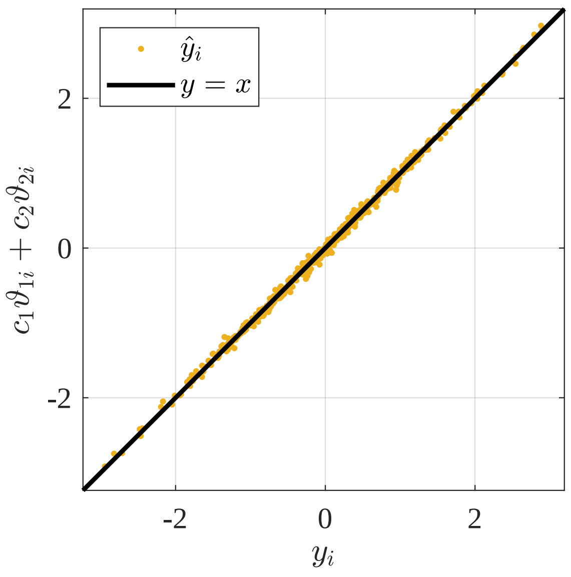


Figure 6.4: *Principal directions detected by PLS.*



a) *Predictions based on* $\boldsymbol{\vartheta}_1$



b) *Predictions based on* $\boldsymbol{\vartheta}_1$ *and* $\boldsymbol{\vartheta}_2$

Figure 6.5: *PLS predictions.*

A 3-dimensional example is given in Figures 6.4 and 6.5 to illustrate the PLS approach. Figure 6.4 depicts the inputs and two principal directions $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$. Figure 6.5 shows the PLS predictions based on the principal components $\boldsymbol{\vartheta}_1$ and $\boldsymbol{\vartheta}_2$, where $y_i = g(\boldsymbol{\theta}^{(i)})$ is the i-th component of $\boldsymbol{y}$, and ϑ_{1i} and ϑ_{2i} are the i-th component of $\boldsymbol{\vartheta}_1$ and $\boldsymbol{\vartheta}_2$, respectively. In the subfigures, the closer these points lie to the line $y = x$, the more accurate the predictions. As is shown in Figure 6.5b, the identified two principal components are able to capture the main input-output relationship.

Algorithm 15 PLS algorithm

Input: Sample matrix $\boldsymbol{\Phi}$; Response vector $\boldsymbol{y}$; The number of principal components m.
Output: Rotation matrix $\boldsymbol{W}^*$.

```
1: Set $\boldsymbol{\Phi}_0 = \boldsymbol{\Phi}$, $\boldsymbol{y}_0 = \boldsymbol{y}$;
2: for $j = 1 : m$ do
3:     Compute the weight vector: $\boldsymbol{w}_j = \boldsymbol{\Phi}_{j-1}^\top \boldsymbol{y}_{j-1} / \|\boldsymbol{\Phi}_{j-1}^\top \boldsymbol{y}_{j-1}\|$;
4:     Obtain the principal component $\boldsymbol{\vartheta}_j = \boldsymbol{\Phi}_{j-1} \boldsymbol{w}_j$;
5:     Calculate the regression coefficients: $\boldsymbol{\alpha}_j = \boldsymbol{\Phi}_{j-1}^\top \boldsymbol{\vartheta}_j / \boldsymbol{\vartheta}_j^\top \boldsymbol{\vartheta}_j$, $c_j = \boldsymbol{\vartheta}_j^\top \boldsymbol{y}_{j-1} / \boldsymbol{\vartheta}_j^\top \boldsymbol{\vartheta}_j$;
6:     Compute the residuals: $\boldsymbol{\Phi}_j = \boldsymbol{\Phi}_{j-1} - \boldsymbol{\vartheta}_j \boldsymbol{\alpha}_j^\top$, $\boldsymbol{y}_j = \boldsymbol{y}_{j-1} - c_j \boldsymbol{\vartheta}_j$;
7: end for
8: Assemble the weight matrix $\boldsymbol{W}$ and the coefficient matrix $\boldsymbol{A}$;
9: Calculate the rotation matrix $\boldsymbol{W}^*$ according to Equation (6.38);
10: return $\boldsymbol{W}^*$.
```

The principal components can be written as

$$\boldsymbol{\vartheta}_j = \boldsymbol{\Phi}_{j-1} \boldsymbol{w}_j = \boldsymbol{\Phi} \boldsymbol{w}_j^*, \quad j = 1, \ldots, m, \tag{6.36}$$

where $\boldsymbol{\Phi}_0 = \boldsymbol{\Phi}$. Then, one has

$$[\boldsymbol{\vartheta}_1, \ldots, \boldsymbol{\vartheta}_m] = \boldsymbol{\Phi} \boldsymbol{W}^*, \tag{6.37}$$

where $\boldsymbol{W}^* = [\boldsymbol{w}_1^*, \ldots, \boldsymbol{w}_m^*]$ is the *rotation matrix* that rotates the original space into the new space with principal directions $\boldsymbol{w}_j, j = 1, \ldots, m$. The rotation matrix can be determined by [152]

$$\boldsymbol{W}^* = \boldsymbol{W} \left(\boldsymbol{A}^\top \boldsymbol{W} \right)^{-1}, \tag{6.38}$$

where $\boldsymbol{W} = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_m]$ is the *weight matrix* and $\boldsymbol{A} = [\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_m]$ is the *coefficient matrix*. The PLS algorithm is summarized in Algorithm 15.

6.2.2 Correlation Functions Constructed by PLS

The PLS approach identifies the j-th principal component by Equation (6.36). For any sample $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_n]^\top$, the corresponding j-th principal component is calculated by

$$\vartheta_j = w_{j1}^* \theta_1 + \cdots + w_{jn}^* \theta_n, \tag{6.39}$$

where $\boldsymbol{w}_j^* = [w_{j1}^*, \ldots, w_{jn}^*]^\top$. Since the items $w_{jk}^* \theta_k, k = 1, \ldots, n$, are equally important for calculating ϑ_j, the correlation function for the linear transformation $h_j : [\theta_1, \ldots, \theta_n]^\top \to [w_{j1}^* \theta_1, \ldots, w_{jn}^* \theta_n]^\top$ can be defined by the following isotropic correlation function:

$$R_j(h_j(\boldsymbol{\theta}), h_j(\boldsymbol{\theta}'); \eta_j) = \prod_{k=1}^{n} R\left(w_{jk}^* \theta_k, w_{jk}^* \theta_k'; \eta_j \right) \tag{6.40}$$

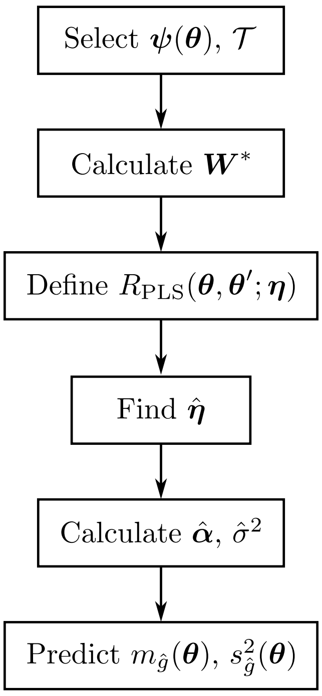

Select $\boldsymbol{\psi}(\boldsymbol{\theta})$, $\mathcal{T}$
Calculate $\boldsymbol{W}^*$
Define $R_{\text{PLS}}(\boldsymbol{\theta}, \boldsymbol{\theta}'; \boldsymbol{\eta})$
Find $\hat{\boldsymbol{\eta}}$
Calculate $\hat{\boldsymbol{\alpha}}$, $\hat{\sigma}^2$
Predict $m_{\hat{g}}(\boldsymbol{\theta})$, $s^2_{\hat{g}}(\boldsymbol{\theta})$

Figure 6.6: *KPLS modeling steps.*

A correlation function incorporating all the information of the first m $(m < n)$ principal components is constructed by the tensor product of $R_j(h_j(\boldsymbol{\theta}), h_j(\boldsymbol{\theta}'); \eta_j), j = 1, \ldots, m$, as follows [153]:

$$R_{\text{PLS}}(\boldsymbol{\theta}, \boldsymbol{\theta}'; \boldsymbol{\eta}) = \prod_{j=1}^{m} R_j(h_j(\boldsymbol{\theta}), h_j(\boldsymbol{\theta}'); \eta_j), \tag{6.41}$$

where $\boldsymbol{\eta} = [\eta_1, \ldots, \eta_m]^\mathsf{T}$. In the anisotropic correlation function given in Equation (6.6), n hyperparameters need to be estimated and they can be regarded as the measures of how strongly the input variables influence the output. By rotating the original space, the PLS correlation function in Equation (6.41) detects the principal directions and neglects the unimportant ones, thus reducing the dimensionality of the optimization search space. Although the PLS approach copes with the linear input-output relationship, the PLS correlation function can also be used in nonlinear problems [153].

Consider the Gaussian correlation function, for example, the corresponding PLS correlation function is given by

$$R_{\text{PLS}}(\boldsymbol{\theta}, \boldsymbol{\theta}'; \boldsymbol{\eta}) = \prod_{j=1}^{m} \prod_{k=1}^{n} \exp\left(-\eta_j (w^*_{jk}\theta_k - w^*_{jk}\theta'_k)^2\right). \tag{6.42}$$

It is demonstrated in [153] that this function is a special case of the anisotropic Gaussian correlation function in Equation (6.7). The other commonly used PLS correlation functions are listed in Appendix C.

The kriging model using the PLS correlation function is called *PLS-accelerated kriging* (KPLS). The modeling steps are summarized in Figure 6.6.

6.2.3 Illustrative Examples

This subsection considers the following Griewank function [153]:

$$g(\boldsymbol{\theta}) = 1 + \sum_{k=1}^{n} \frac{\theta_k^2}{4000} - \prod_{k=1}^{n} \cos\left(\frac{\theta_k}{\sqrt{k}}\right), \tag{6.43}$$

where $n = 10$ and $\theta_k \in [-100, 100]$, $k = 1, \ldots, 10$. Kriging (see Figure 6.3) and KPLS (see Figure 6.6) are implemented to approximate this function and then predict the model responses. They both utilize linear trends but employ the anisotropic Gaussian correlation function in Equation (6.7) and the PLS Gaussian correlation function in Equation (6.42), respectively. Additionally, the KPLS approach is applied with different numbers of principal components: $m = 1, 2, 3$. In both methods, $N_t = 100$ uniformly distributed training samples $\boldsymbol{\theta}_t^{(i)}$ are drawn to construct the kriging surrogate and a test set with $N = 5000$ samples $\boldsymbol{\theta}^{(i)}$ is generated to evaluate the generalization ability of the surrogate. All results are obtained in *Matlab R2020a* based on the *ooDace* toolbox [154]. The computations were done on a desktop PC equipped with a 4-core Intel(R) Core(TM) i7-6700 CPU @3.40GHz.

Table 6.1 presents the training time t_{train} required by both methods and the generalization error $\tilde{\varepsilon}_{\text{gen}}$ of the constructed surrogates. The generalization error is defined as

$$\varepsilon_{\text{gen}} = \frac{1}{N} \sum_{i=1}^{N} \left(g(\boldsymbol{\theta}^{(i)}) - \hat{g}(\boldsymbol{\theta}^{(i)})\right)^2, \tag{6.44}$$

$$\tilde{\varepsilon}_{\text{gen}} = \frac{\varepsilon_{\text{gen}}}{\text{Var}\left[g(\boldsymbol{\theta}^{(i)})\right]}. \tag{6.45}$$

Each approach is conducted 50 times and the results are the average values of the 50 estimates. In this example, the KPLS method with a single principal component requires the shortest training time but results in the largest error. The KPLS with 2 or 3 principal components achieves the same accuracy level as the conventional kriging but needs far fewer training time. The results demonstrate that the PLS dimensionality reduction strategy is able to accelerate the training of the kriging surrogate model by reducing the number of hyperparameters while maintaining the similar modeling accuracy.

Table 6.1: *Kriging and KPLS surrogate modeling results.*

Method	t_{train} (s)	$\tilde{\varepsilon}_{\text{gen}}$
Kriging	3.25	2.72×10^{-4}
KPLS $(m = 1)$	0.05	4.22×10^{-4}
KPLS $(m = 2)$	0.08	2.76×10^{-4}
KPLS $(m = 3)$	0.14	2.42×10^{-4}

6.3 Kriging-Accelerated Simulation Methods

This section incorporates the kriging surrogate model into MCS and SuS to accelerate the process of these simulation methods. Before that, the necessary elements regarding active learning, trend identification, and experimental design are first introduced.

6.3.1 Active Learning Strategy

As described in Figure 5.7, SuS can be interpreted as a series of classification problems and MCS is a special case of SuS. To measure the classification reliability of the constructed surrogate model, the *U-function* is defined as [81]

$$U(\boldsymbol{\theta}) = \frac{|m_{\hat{g}}(\boldsymbol{\theta}) - b|}{s_{\hat{g}}(\boldsymbol{\theta})}, \tag{6.46}$$

where b is the *classification boundary*. If the function output is predicted to be larger than b, i.e., $\hat{g}(\boldsymbol{\theta}) = m_{\hat{g}}(\boldsymbol{\theta}) > b$, then the probability that the prediction is no larger than b is estimated by

$$\mathbb{P}\left[\hat{g}(\boldsymbol{\theta}) \leq b\right] = \Phi\left(\frac{b - m_{\hat{g}}(\boldsymbol{\theta})}{s_{\hat{g}}(\boldsymbol{\theta})}\right) = \Phi\left(-U(\boldsymbol{\theta})\right), \tag{6.47}$$

since $\hat{g}(\boldsymbol{\theta})$ is assumed to be Gaussian-distributed with mean $m_{\hat{g}}(\boldsymbol{\theta})$ and variance $s^2_{\hat{g}}(\boldsymbol{\theta})$. This means that the probability of misclassification is $\Phi(-U(\boldsymbol{\theta}))$. Similarly, one can obtain the same misclassification probability for the case $m_{\hat{g}}(\boldsymbol{\theta}) < b$. Therefore, the U-function values reflect the classification reliability and a smaller U-function value indicates that the kriging surrogate is less confident in classifying the current sample correctly. In general, a small U-function value results from a prediction close to the threshold b or/and a large prediction variance.

To improve the classification accuracy of the kriging surrogate, the work in [81] proposed an active learning strategy which chooses the sample with the smallest U-function value as the best next point:

$$\boldsymbol{\theta}^* = \arg\min U(\boldsymbol{\theta}^{(i)}), \tag{6.48}$$

adds it to the training set, and updates the kriging surrogate. The best next point has the largest risk of misclassification, thus being the most valuable sample for enhancing the classification accuracy. This learning process is repeated until all the samples are classified with sufficient confidence:

$$\min U(\boldsymbol{\theta}^{(i)}) \geq u_b, \tag{6.49}$$

where u_b is the *threshold* for the stopping criterion. As suggested in [81], u_b is chosen to be 2 in this thesis. Because $\Phi(-2) \approx 2.3\%$, this stopping criterion means that any sample is correctly classified with a probability greater than 97.7%.

Algorithm 16 Active learning strategy for the use of kriging in classification problems

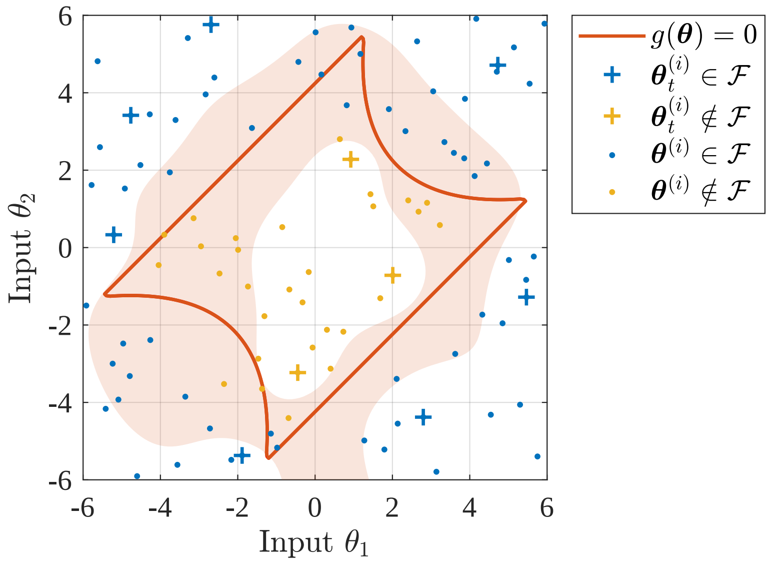
```
Input: Current kriging surrogate model $\hat{g}(\boldsymbol{\theta})$; Samples $\{\boldsymbol{\theta}^{(i)} : i = 1, \ldots, N\}$; Classification
    boundary $b$; Limit-state function $g(\boldsymbol{\theta})$; Training set $\mathcal{T}$.
Output: Updated kriging surrogate model $\hat{g}(\boldsymbol{\theta})$.
 1: Obtain the predictions and the prediction variances $\{(m_{\hat{g}}(\boldsymbol{\theta}^{(i)}), s^2_{\hat{g}}(\boldsymbol{\theta}^{(i)})) : i = 1, \ldots, N\}$;
 2: Calculate the U-function values $\{U(\boldsymbol{\theta}^{(i)}) : i = 1, \ldots, N\}$ as in Equation (6.46);
 3: Find the best next point $\boldsymbol{\theta}^*$ according to Equation (6.48);
 4: while $U(\boldsymbol{\theta}^*) < 2$ do
 5:     Calculate the limit-state value $g(\boldsymbol{\theta}^*)$ and add $\boldsymbol{\theta}^*$ to $\mathcal{T}$;
 6:     Refine $\hat{g}(\boldsymbol{\theta})$ using the updated $\mathcal{T}$;
 7:     Update the predictions and the variances $\{(m_{\hat{g}}(\boldsymbol{\theta}^{(i)}), s^2_{\hat{g}}(\boldsymbol{\theta}^{(i)})) : i = 1, \ldots, N\}$
        employing the refined $\hat{g}(\boldsymbol{\theta})$;
 8:     Update the U-function values $\{U(\boldsymbol{\theta}^{(i)}) : i = 1, \ldots, N\}$ as in Equation (6.46);
 9:     Find the best next point $\boldsymbol{\theta}^*$ according to Equation (6.48);
10: end while
11: return $\hat{g}(\boldsymbol{\theta})$.
```

The active learning strategy is summarized in Algorithm 16. Here, only a single point is added to the training set in each iteration. The user can also add multiple samples every time. The previous choice enriches the training set with the fewest number of samples, whereas the latter one reduces the required number of kriging model refinements.

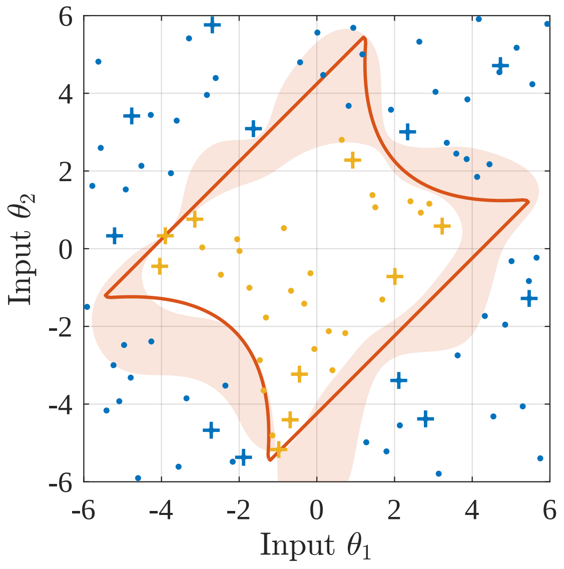
An example is given to showcase the active learning strategy. Consider the four-branch function [87] as follows:

$$g(\boldsymbol{\theta}) = \min \left\{ \begin{array}{c} 3 + 0.1\left(\theta_1 - \theta_2\right)^2 - \frac{\theta_1+\theta_2}{\sqrt{2}} \\ 3 + 0.1\left(\theta_1 - \theta_2\right)^2 + \frac{\theta_1+\theta_2}{\sqrt{2}} \\ (\theta_1 - \theta_2) + \frac{6}{\sqrt{2}} \\ (\theta_2 - \theta_1) + \frac{6}{\sqrt{2}} \end{array} \right\}. \tag{6.50}$$

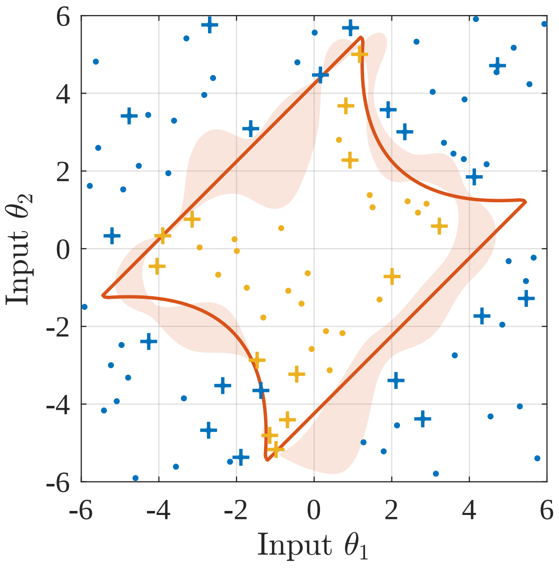
The target is to judge whether the generated $N = 100$ samples $\boldsymbol{\theta}^{(i)}$ fall in the failure domain $\mathcal{F}$, namely, whether $g(\boldsymbol{\theta}^{(i)}) \leq 0$. First of all, 10 training samples $\boldsymbol{\theta}_t^{(i)}$ are selected from $\boldsymbol{\theta}^{(i)}$ to construct the initial kriging surrogate $\hat{g}(\boldsymbol{\theta})$. After that, the active learning strategy is exploited to enrich the training set and refine the surrogate. The learning process is illustrated in Figure 6.7. Figure 6.7a shows the results before the learning process. The red shaded area represents the domain that contains samples of interest for enriching the training set $\mathcal{S}_U = \{\boldsymbol{\theta} \in \boldsymbol{\Omega} : U(\boldsymbol{\theta}) < 2\}$. The non-training points are classified by the surrogate $\hat{g}(\boldsymbol{\theta})$, and a few of them are misclassified. Figure 6.7b depicts the results when 10 samples have been added to the training set, and Figure 6.7c shows the final learning results. With the enrichment of the training set and the refinement of the kriging surrogate, $\mathcal{S}_U$ becomes smaller and fewer samples are misclassified. In the end, there is



a) *Results before active learning* ($N_t = 10$)



b) *Ongoing results* ($N_t = 20$)



c) *Results after learning* ($N_t = 32$)

Figure 6.7: *Active learning process for the use of kriging in classification problems.*

not any sample in $\mathcal{S}_U$ or misclassified. Besides, all the new training samples locate around the classification boundary, thus providing an efficient kriging model refinement process for classification problems.

6.3.2 Trend Identification

Ordinary kriging is typically utilized when there is not any knowledge about the system nonlinearity. However, the detection of non-constant trends enhances the prediction accuracy and thus the modeling efficiency for classification problems [87]. This thesis implements the adaptive PCE (see Algorithm 9) to identify the best polynomial trend in terms of the leave-one-out error. With the enrichment of the training set, the number of training samples N_t may increase greatly. In this case, the maximal polynomial expansion order $d_{\max}$ is chosen as the maximum order such that the number of expansion bases is no larger than N_t:

$$\frac{(n+d_{\max})!}{n!d_{\max}!} \leq N_t < \frac{(n+d_{\max}+1)!}{n!(d_{\max}+1)!}, \tag{6.51}$$

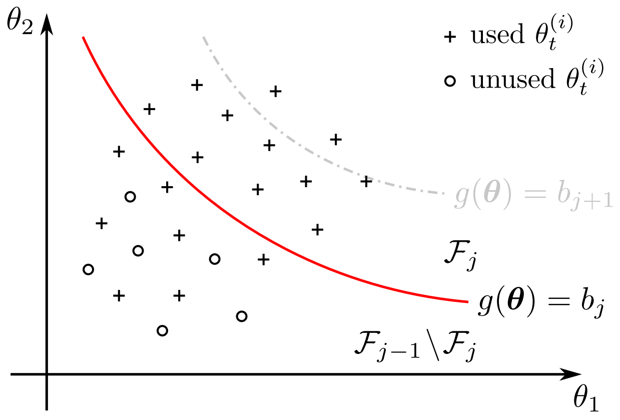
where n is the dimension of uncertain inputs. The minimal expansion order $d_{\min}$ is set to be 1, which corresponds to the linear approximation. Accordingly, the number of initial training samples should not be smaller than n.

6.3.3 Experimental Design Strategy

Section 6.3.1 introduces the learning strategy enriching the training set for the use of kriging in classification problems. However, as discussed in Section 6.1.5, the increasing number of training samples requires more computational effort. To reduce this demand in the joint use of kriging and SuS, this subsection proposes an experimental design strategy that selects influential training points from the full training set to update the kriging surrogate. Here, the full training set consists of all the samples that have been evaluated using true model.

When kriging is applied within the framework of SuS, the goal of the kriging surrogate is to distinguish whether or not the generated samples belong to the next intermediate failure domain. Hence, the focus of the kriging surrogate moves as the samples approach the failure domain. At the j-th level $(j \geq 1)$ of SuS, samples are drawn in the intermediate failure domain $\mathcal{F}_j$. The kriging surrogate needs to be refined locally to accurately find the next intermediate boundary $\{\boldsymbol{\theta} \in \boldsymbol{\Omega} : g(\boldsymbol{\theta}) = b_{j+1}\}$ and classify the generated samples based on the boundary. In this case, the training points in $\mathcal{F}_{j-1} \backslash \mathcal{F}_j$, as shown in Figure 6.8, are far away from the domain of interest, thereby contributing less to the update of the kriging surrogate. This is because the stationary correlation function used in kriging only depends on the distance between two points, and the function value decreases as the distance grows.

Instead of using all the training samples to update the kriging surrogate [83, 84], this thesis seeks to discard the unimportant ones to reduce the computational expense for the kriging model refinement. As illustrated in Figure 6.8, at the j-th subset level $(j \geq 1)$, a training set $\widetilde{\mathcal{T}}$ including all the training samples in $\mathcal{F}_j$ and a proportion of training points in $\mathcal{F}_{j-1} \backslash \mathcal{F}_j$ is determined to refine the surrogate model. This indicates that $\widetilde{\mathcal{T}} \subset \mathcal{T}$. The percentage of training samples that are discarded in $\mathcal{F}_{j-1} \backslash \mathcal{F}_j$ is denoted by p_d. Specifically, we record the latest prediction variances of the training points before they are added to the training set and discard p_d of them with smaller variances. Essentially, a sample with larger variance contributes more to the global trend, whereas one with smaller variance



θ_2
+ used $\theta_t^{(i)}$
○ unused $\theta_t^{(i)}$
$g(\boldsymbol{\theta}) = b_{j+1}$
$\mathcal{F}_j$
$g(\boldsymbol{\theta}) = b_j$
$\mathcal{F}_{j-1} \backslash \mathcal{F}_j$
θ_1

Figure 6.8: *Experimental design for the joint use of kriging and SuS.*

Algorithm 17 Experimental design strategy for kriging at the j-th subset level ($j \geq 1$)

```
Input: Training set $\mathcal{T} = \{(\boldsymbol{\theta}_t^{(i)}, g(\boldsymbol{\theta}_t^{(i)}))\}$; The latest prediction variances of training
    samples before they are added to the training set $s_{\hat{g}}^2(\boldsymbol{\theta}_t^{(i)})$; Intermediate threshold $b_j$;
    The percentage of discarded training samples $p_d$.
Output: The training set for refining the kriging surrogate $\widetilde{\mathcal{T}}$.
1: Divide $\mathcal{T}$ into two sets $\widetilde{\mathcal{T}} = \{(\boldsymbol{\theta}_t^{(i)}, g(\boldsymbol{\theta}_t^{(i)})) : g(\boldsymbol{\theta}_t^{(i)}) \leq b_j\}$ and $\mathcal{T} \backslash \widetilde{\mathcal{T}} = \{(\boldsymbol{\theta}_t^{(i)}, g(\boldsymbol{\theta}_t^{(i)})) :$
    $g(\boldsymbol{\theta}_t^{(i)}) > b_j\}$;
2: Find $1 - p_d$ of the samples in $\mathcal{T} \backslash \widetilde{\mathcal{T}}$ with larger $s_{\hat{g}}^2(\boldsymbol{\theta}_t^{(i)})$ and move them to $\widetilde{\mathcal{T}}$;
3: return $\widetilde{\mathcal{T}}$.
```

focuses more on capturing the local variability. This measure attempts to neglect the local characteristic in the safer domain but still retain the global feature. The experimental design strategy at the j-th subset level is summarized in Algorithm 17. At the initial SuS level, all the training samples are adopted to update the kriging surrogate, i.e., $\widetilde{\mathcal{T}} = \mathcal{T}$.

6.3.4 Kriging-Accelerated MCS/SuS

This subsection integrates the kriging surrogate into MCS and SuS to accelerate the simulation procedure. Since MCS can be regarded as the SuS with only the initial level, only *kriging-accelerated SuS* (kriging-SuS) is discussed in detail.

Figure 6.9 illustrates the main steps of the kriging-SuS approach. At the first subset level, the initial kriging surrogate model $\hat{g}(\boldsymbol{\theta})$ is constructed using a fraction of samples. One can choose to use conventional kriging or KPLS according to the complexity of the problem. After that, the intermediate threshold b_1 is determined. The active learning strategy (see Section 6.3.1) iteratively enriches the training set around the boundary $\{\boldsymbol{\theta} \in \boldsymbol{\Omega} : g(\boldsymbol{\theta}) = b_1\}$ with the most valuable point in terms of the U-function and then refines the surrogate $\hat{g}(\boldsymbol{\theta})$. Meanwhile, b_1 is updated repeatedly as well. The learning

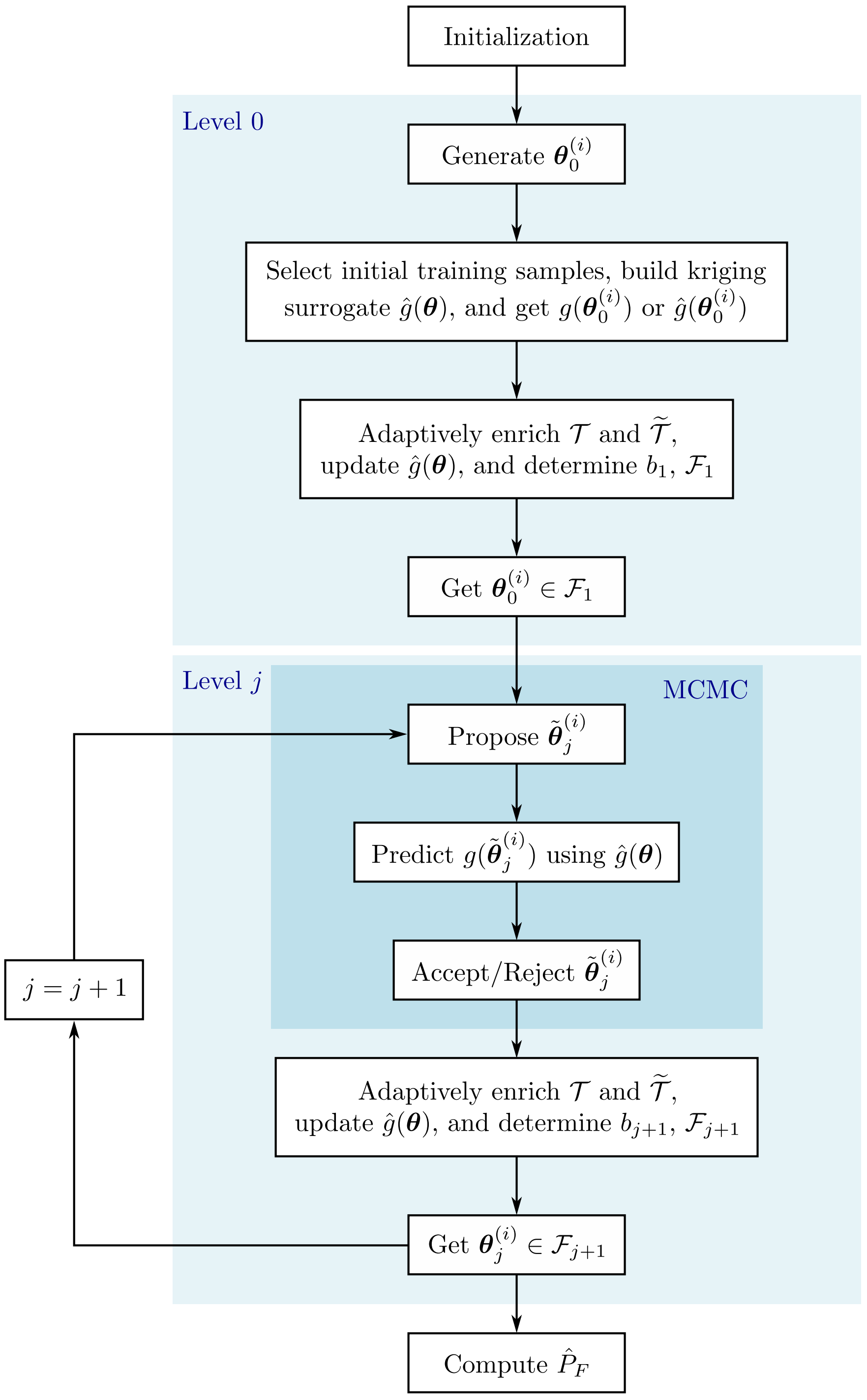

Initialization
Level 0
Generate $\boldsymbol{\theta}_0^{(i)}$
Select initial training samples, build kriging surrogate $\hat{g}(\boldsymbol{\theta})$, and get $g(\boldsymbol{\theta}_0^{(i)})$ or $\hat{g}(\boldsymbol{\theta}_0^{(i)})$
Adaptively enrich $\mathcal{T}$ and $\widetilde{\mathcal{T}}$, update $\hat{g}(\boldsymbol{\theta})$, and determine b_1, $\mathcal{F}_1$
Get $\boldsymbol{\theta}_0^{(i)} \in \mathcal{F}_1$
Level j
MCMC
Propose $\tilde{\boldsymbol{\theta}}_j^{(i)}$
Predict $g(\tilde{\boldsymbol{\theta}}_j^{(i)})$ using $\hat{g}(\boldsymbol{\theta})$
Accept/Reject $\tilde{\boldsymbol{\theta}}_j^{(i)}$
$j = j + 1$
Adaptively enrich $\mathcal{T}$ and $\widetilde{\mathcal{T}}$, update $\hat{g}(\boldsymbol{\theta})$, and determine b_{j+1}, $\mathcal{F}_{j+1}$
Get $\boldsymbol{\theta}_j^{(i)} \in \mathcal{F}_{j+1}$
Compute $\hat{P}_F$

Figure 6.9: *Schema of the kriging-SuS approach.*

process terminates until the surrogate gains sufficient confidence in the classification results. In the end, the samples falling in the intermediate failure domain $\mathcal{F}_1$ are considered as the seeds for the next subset level.

At the j-th subset level ($j = 1, \dots, m-1$), new samples are first generated from the seeds. These points are accepted or rejected based on whether they lie in $\mathcal{F}_j$ from the viewpoint of the latest $\hat{g}(\boldsymbol{\theta})$. Subsequently, the focus is shifted to the next intermediate threshold b_{j+1}. The active learning procedure is performed again to efficiently and accurately solve

Algorithm 18 The kriging-SuS approach

```
Input: The PDF of uncertain parameters $f(\boldsymbol{\theta})$; Limit-state function $g(\boldsymbol{\theta})$; Conditional
    probability $p_0$; The number of samples per subset level $N$; The percentage of points
    that are chosen as the initial training samples $\tilde{p}_0$; Correlation function $R(\boldsymbol{\theta}, \boldsymbol{\theta}'; \boldsymbol{\eta})$; The
    percentage of discarded training samples $p_d$.
Output: The estimate of failure probability $\hat{P}_F$.
 1: Draw $N$ i.i.d. samples $\{\boldsymbol{\theta}_0^{(i)} : i = 1, \ldots, N\}$ in accordance with $f(\boldsymbol{\theta})$ and initialize an
    empty training set $\mathcal{T}$;
 2: Randomly select $\tilde{p}_0 N$ training samples $\{\boldsymbol{\theta}_{0,t}^{(i)} : i = 1, \ldots, \tilde{p}_0 N\}$ from the $N$ samples,
    where $\tilde{p}_0 N$ is an integer, calculate the corresponding limit-state values $\{g(\boldsymbol{\theta}_{0,t}^{(i)}) : i =
    1, \ldots, \tilde{p}_0 N\}$, and add these training samples to $\mathcal{T}$;
 3: Identify the best polynomial trend order (Algorithm 9), build a kriging surrogate
    model $\hat{g}(\boldsymbol{\theta})$, and obtain the predictions and the prediction variances of the $N$ samples
    $\{(m_{\hat{g}}(\boldsymbol{\theta}_0^{(i)}), s_{\hat{g}}^2(\boldsymbol{\theta}_0^{(i)})) : i = 1, \ldots, N\}$;
 4: Find $b_1$ as the $p_0$-percentile of the $N$ predictions $\{m_{\hat{g}}(\boldsymbol{\theta}_0^{(i)})\}$;
 5: Adaptively enrich $\mathcal{T}$ and update $\hat{g}(\boldsymbol{\theta})$ (Algorithm 16); In the meantime, update $b_1$ and
    set $\mathcal{F}_1 = \{\boldsymbol{\theta} \in \boldsymbol{\Omega} : g(\boldsymbol{\theta}) \leq b_1\}$;
 6: Set $j = 1$;
 7: while $b_j > 0$ do
 8:     Consider samples $\boldsymbol{\theta}_{j-1}^{(i)} \in \mathcal{F}_j$ as seeds $\{\boldsymbol{\theta}_{j-1,s}^{(i)} : i = 1, \ldots, N_s\}$, where $N_s = p_0 N$ is
        an integer;
 9:     Propose $N$ candidate samples $\{\tilde{\boldsymbol{\theta}}_j^{(i)} : i = 1, \ldots, N\}$ from the seeds;
10:     Obtain the predictions of these candidate samples $\{m_{\hat{g}}(\tilde{\boldsymbol{\theta}}_j^{(i)}) : i = 1, \ldots, N\}$;
11:     Accept or reject $\tilde{\boldsymbol{\theta}}_j^{(i)}$ based on whether $m_{\hat{g}}(\tilde{\boldsymbol{\theta}}_j^{(i)}) \leq b_j$; After rejection, the samples
        are denoted by $\{\boldsymbol{\theta}_j^{(i)} : i = 1, \ldots, N\}$;
12:     Find $b_{j+1}$ as the $p_0$-percentile of the $N$ predictions $\{m_{\hat{g}}(\boldsymbol{\theta}_j^{(i)})\}$;
13:     Identify the best trend order (Algorithm 9), adaptively enrich $\mathcal{T}$ and $\tilde{\mathcal{T}}$, and
        update $\hat{g}(\boldsymbol{\theta})$ using $\tilde{\mathcal{T}}$ (Algorithms 16 and 17); In the meantime, update $b_{j+1}$ and
        set $\mathcal{F}_{j+1} = \{\boldsymbol{\theta} \in \boldsymbol{\Omega} : g(\boldsymbol{\theta}) \leq b_{j+1}\}$;
14:     Set $j = j + 1$;
15: end while
16: Set the total number of subsets $m = j$, set $b_m = 0$, denote the number of samples
    $\boldsymbol{\theta}_{m-1}^{(i)} \in \mathcal{F}$ by $N_F$, and estimate the failure probability as in Equation (2.54);
17: return $\hat{P}_F$.
```

the new classification problem with the boundary $\{\boldsymbol{\theta} \in \boldsymbol{\Omega} : g(\boldsymbol{\theta}) = b_{j+1}\}$. Additionally, the experimental design strategy (see Section 6.3.3) is applied to reduce the cardinality of the training set for refining the kriging surrogate. Finally, the samples in $\mathcal{F}_{j+1}$ are identified as the seeds for the next subset level. The detailed steps are presented in Algorithm 18. Here, the adaptive trend order selection (see Section 6.3.2) is only performed at the beginning of each subset. Since the active learning strategy seeks to accurately classify the generated samples, the c.o.v. approximation for the conventional SuS (see Section 2.5.3) is still exploited to measure the estimation accuracy of the failure probability.

6.3.5 Illustrative Examples

Consider the 28-dimensional limit-state function in Section 5.4.2:

$$g(\boldsymbol{\theta}) = a - \left(\sum_{i=1}^{7} 0.1 e^{0.1\theta_i} + \sum_{i=8}^{14} 0.5 e^{0.1\theta_i} + \sum_{i=15}^{21} 2 e^{0.1\theta_i} + \sum_{i=22}^{28} 8 e^{0.1\theta_i} \right), \tag{6.52}$$

where $a = 85$ and $\theta_i, i = 1, \ldots, 28$, are i.i.d. Gaussian random variables with zero means and unit standard deviations. Kriging-SuS methods with different settings, as listed in Table 6.2, are implemented to evaluate the failure probability $\mathbb{P}\left[g(\boldsymbol{\theta}) \leq 0\right]$. Here, m represents the number of principal components for PLS and p_d represents the percentage of discarded training samples for experimental design.

Table 6.3 shows the evaluation results of 50 independent implementations of three different methods, SuS, kriging-SuS, and KPLS-SuS1, with $N = 2000$ samples per subset level. The failure probability $\hat{P}_F$, the c.o.v. bounds $\hat{c}_{v,\mathrm{lb}}$ and $\hat{c}_{v,\mathrm{ub}}$, the number of calls to the true model N_{call}, and the training time t_{train} are the average values of the 50 results, whereas the c.o.v. estimate $\hat{c}_v$ is the sample c.o.v. of the 50 failure probability estimates. Kriging-SuS and KPLS-SuS1 are able to achieve similar probability estimates and the same accuracy level as SuS. This is due to the fact that the active learning strategy for kriging aims to provide sufficiently accurate classification results. Nonetheless, the kriging-accelerated methods only need to evaluate the true model a few hundred times,

Table 6.2: *Kriging-SuS methods with different settings.*

Method	Use PLS dimensionality reduction	Use experimental design
kriging-SuS	No	No
KPLS-SuS1	Yes ($m = 3$)	No
KPLS-SuS2	Yes ($m = 3$)	Yes ($p_d = 0.5$)

Table 6.3: *Failure probability estimation results using SuS and kriging-SuS.*

Method	$\hat{P}_F$	$[\hat{c}_{v,\mathrm{lb}}, \hat{c}_{v,\mathrm{ub}}]$	$\hat{c}_v$	N_{call}	t_{train} (s)
SuS	5.54×10^{-6}	$[0.25, 0.58]$	0.32	11920	— —
kriging-SuS	5.79×10^{-6}	$[0.25, 0.58]$	0.33	176	1346
KPLS-SuS1	5.82×10^{-6}	$[0.25, 0.58]$	0.29	184	35.3

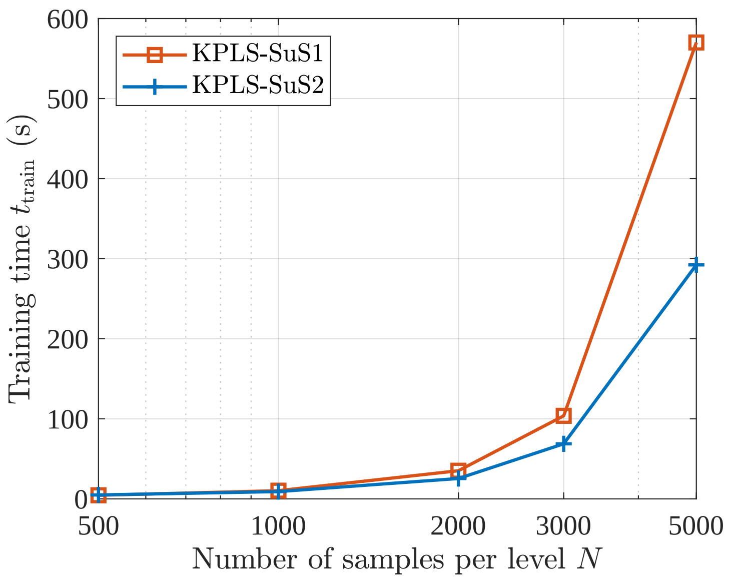


Figure 6.10: *Training time for KPLS-SuS with different N.*

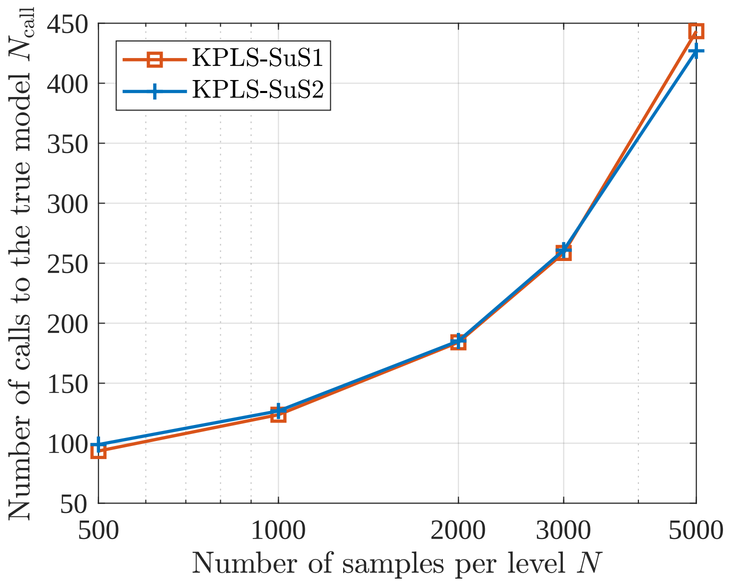
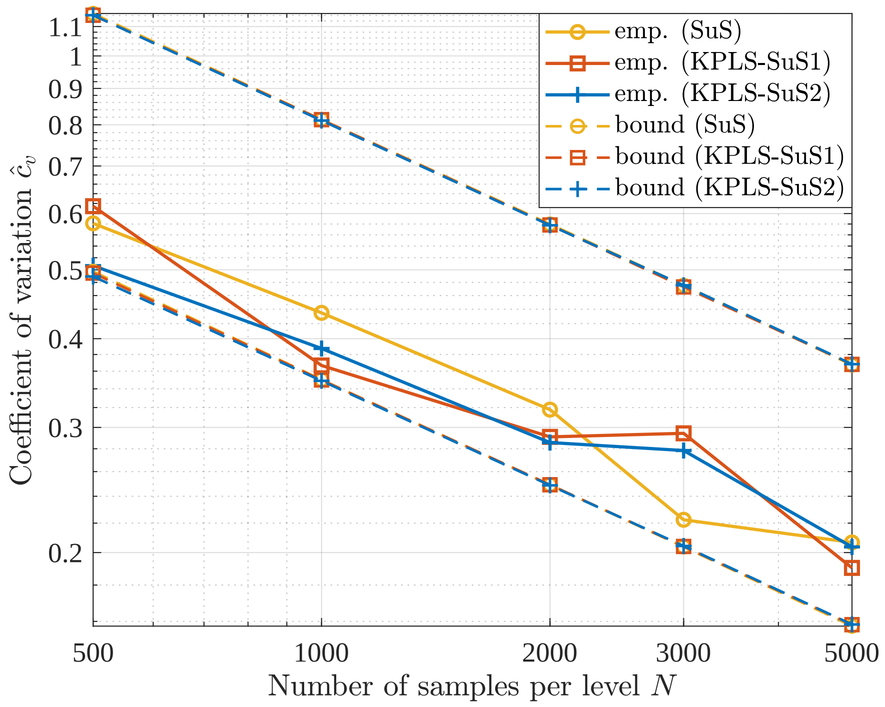


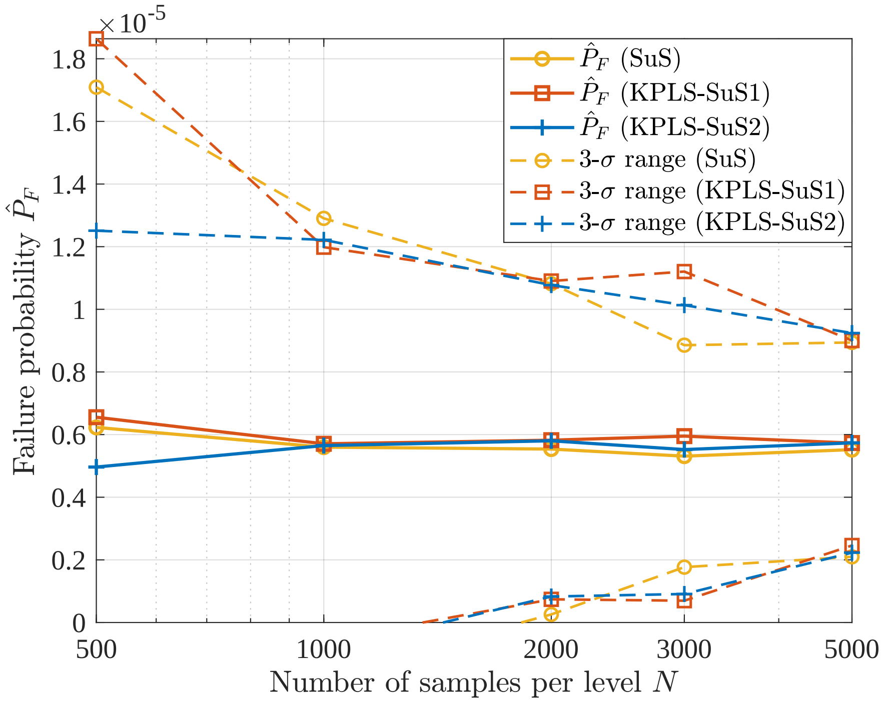
Figure 6.11: *Number of true model evaluations for KPLS-SuS with different N.*

which saves a considerable proportion of true model evaluations. The high efficiency is accomplished at the cost of the training time. The basic kriging-SuS requires 1346 seconds to iteratively update the kriging as the SuS samples approach the failure domain. However, this demand can be greatly mitigated by the PLS dimensionality reduction strategy.

In the following, three methods, SuS, KPLS-SuS1, and KPLS-SuS2, are applied with different numbers of samples per level: $N = [500, 1000, 2000, 3000, 5000]^{\mathsf{T}}$. For each N, 50 independent applications of these methods are conducted. The simulation results are shown in Figures 6.10, 6.11, and 6.12. Same as the results in Table 6.3, the training time, the number of calls to the true model, the c.o.v. bounds, and the failure probability are the average values of the 50 results. The empirical c.o.v., denoted by "emp." in the figure



a) *Estimates of coefficients of variation*



b) *Estimates of failure probabilities*

Figure 6.12: *Failure probability estimation results for SuS and KPLS-SuS with different N.*

legend, is the sample c.o.v. of the 50 failure probability estimates. Figures 6.10 and 6.11 compare the training time and the number of true model evaluations required by the KPLS-SuS methods. It is shown that the proposed experimental design strategy further reduces the training time. The improvement gets more evident when the cardinality of the training set increases. In general, using a subset of the full training set to update the kriging surrogate would add more samples to the training set than using the full set. However, as depicted in Figure 6.11, the experimental design strategy does not necessitate a larger full training set. The results demonstrate that the proposed strategy enhances the training efficiency without requiring more true model evaluations. Figure 6.12 presents the failure probability estimates and the corresponding c.o.v. results. All the results obtained by the KPLS-SuS approaches are similar with those by the conventional SuS method, thus proving the accuracy of the KPLS-SuS approaches.

6.4 Comparisons with Other Surrogates

6.4.1 Surrogate Modeling Techniques

The features of three types of surrogate modeling techniques, namely, RSM/PCE (see Chapter 4), MLS (see Chapter 5), and kriging, are compared in Table 6.4. RSM/PCE exploits the polynomial expansion to construct a global surrogate and then predicts the true model responses. It is easy to implement but limited to weakly nonlinear and low-dimensional problems.

Table 6.4: *Feature comparisons of surrogate modeling techniques.*

Method	RSM/PCE	MLS	Kriging
Model assumption	Polynomial	Polynomial	Polynomial + Gaussian process
Model type	Global	Local	Global + local
Estimation of model parameters	Ordinary least-squares (OLS)	Weighted least-squares (WLS)	Maximum likelihood estimation (MLE)
Advantages	Easy to implement	- More flexible than global polynomial models - Applicable to highly nonlinear problems	- Provides prediction variances - Interpolates known samples - More flexible than polynomial models - Applicable to highly nonlinear problems
Disadvantages	Unsuitable for highly nonlinear or high-dimensional problems	Requires many computations of expansion coefficients	Requires much computational effort

Table 6.5: *Performance comparisons of surrogate modeling techniques.*

Flexibility	Kriging > MLS > RSM/PCE
Accuracy	Kriging > MLS > RSM/PCE
Required number of true model evaluations	Kriging < MLS, RSM/PCE
Computational demand	Kriging > MLS > RSM/PCE

By contrast, MLS builds a low-order polynomial surrogate locally for each unknown point and assigns larger weights to the training points that are closer to the query point. The local surrogate can be used to approximate any nonlinearity given enough training points, thus being more flexible and accurate than RSM/PCE. A typical example is given in Figure 5.4. MLS gains more flexibility and applicability at the expense of computational cost. Since the local expansion must be conducted at every query point, MLS requires more computational effort than RSM/PCE.

Kriging not only utilizes polynomials to detect the global trend but also employs Gaussian process to capture the local variability. Therefore, it is more flexible than the aforementioned polynomial-based surrogates and applicable to highly nonlinear problems. In addition, it provides interpolation results as well as variance estimates for predictions. An example is presented in Figure 6.2. Optimization techniques are generally applied to find the best kriging parameters, thereby resulting in the optimal surrogate. Compared with RSM/PCE and MLS, the optimization step enables kriging to accurately mimic the true model behavior with a smaller number of training samples, but necessitates longer training time.

Similar as RSM/PCE, the conventional MLS and kriging also suffer from the curse of dimensionality, but respective strategies have been introduced to alleviate this problem. The performance comparisons of these surrogate models are summarized in Table 6.5.

6.4.2 Surrogate-Based SuS Approaches

The main elements for the combination of SuS with each surrogate modeling method are compared in Table 6.6. *Polynomial surrogate-based* SuS (PS-SuS) utilizes an adaptive algorithm to select the optimal expansion order minimizing the *leave-one-out* (LOO) error. Similarly, kriging-SuS applies this algorithm to detect the polynomial trend. By contrast, MLS-SuS does not need such step since the second-order approximation is always adopted in the proposed approach.

Aiming at accurately achieving the classification task within the SuS procedure, active learning strategies are introduced for MLS-SuS and kriging-SuS. The MLS prediction error is approximated and the samples that are more likely to be misclassified are added to

Table 6.6: *Main elements of each type of surrogate-accelerated SuS.*

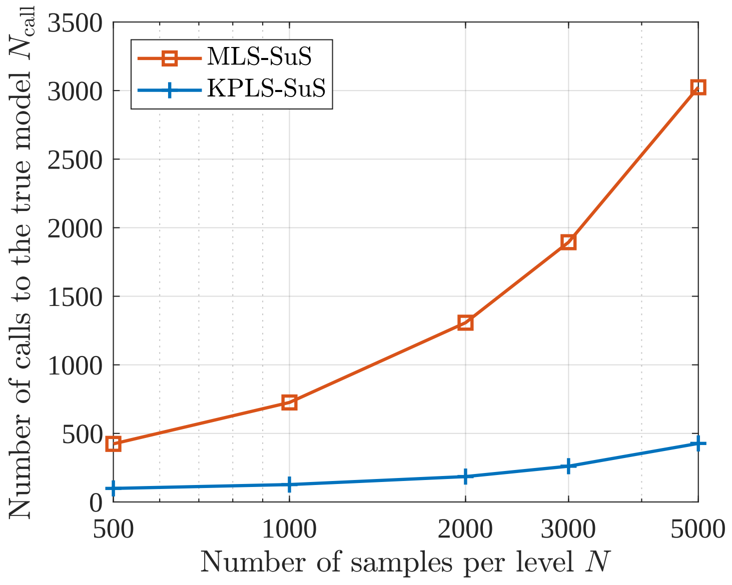
Method	PS-SuS	MLS-SuS	Kriging-SuS
Adaptive expansion order	$d^* = \arg\min \tilde{\varepsilon}_{\text{LOO}}(d)$	— —	$d^* = \arg\min \tilde{\varepsilon}_{\text{LOO}}(d)$
Active learning strategy	— —	$\boldsymbol{\theta}^* = \{\boldsymbol{\theta}^{(i)} : \lvert\hat{g}(\boldsymbol{\theta}^{(i)}) - b_j\rvert < \epsilon^{(i)}\}$	$\boldsymbol{\theta}^* = \arg\min U(\boldsymbol{\theta}^{(i)})$
Experimental design strategy	Uses fewer training samples outside $\mathcal{F}_j$	— —	Uses fewer training samples outside $\mathcal{F}_j$
Dimensionality reduction	— —	Performs sensitivity analysis to detect the influential variables	Applies PLS to reduce the number of hyperparameters

the training set. Likewise, samples with the least confidence for the kriging classification results are chosen as the next training samples, wherein the confidence is represented by the U-function value.

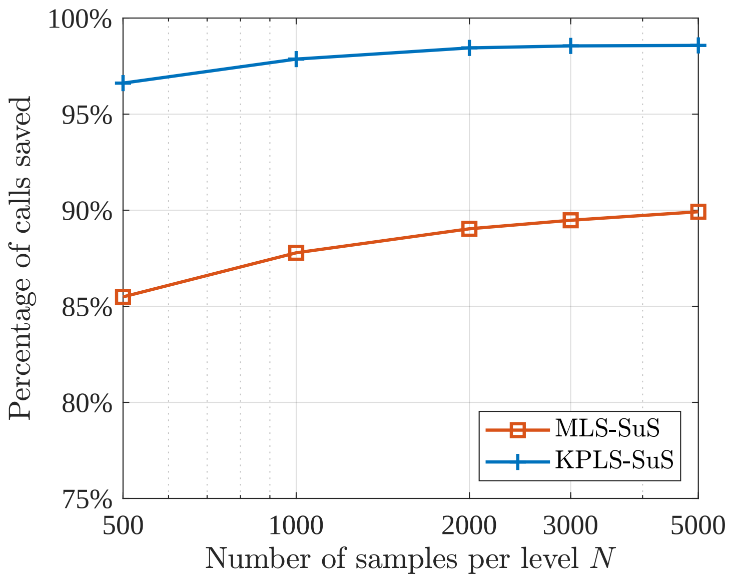
When updating the surrogate model in the intermediate failure domain $\mathcal{F}_j$, experimental design strategies are proposed for PS-SuS and kriging-SuS to neglect some unimportant training samples outside $\mathcal{F}_j$. However, these two methods have different purposes. PS-SuS seeks to focus more on the domain of interest, whereas kriging-SuS attempts to reduce the computational expense. Additionally, MLS-SuS does not need an experimental design strategy because MLS automatically selects the training samples of interest from the full training set for any query point.

For high-dimensional problems, dimensionality reduction techniques are developed to save the training time. MLS-SuS conducts sensitivity analysis to detect the influential variables and then reduces the dimensionality of the feature space. Kriging-SuS implements *partial least-squares* (PLS) to reduce the number of unknown hyperparameters, i.e., the dimensionality of the optimization search space.

As PS-SuS is limited to weakly nonlinear and low-dimensional problems, only the implementations of MLS-SuS and kriging-SuS are discussed in the sequel. In Section 5.4.2 and Section 6.3.5, the MLS-SuS approach with dimensionality reduction and the KPLS-SuS approach are implemented to solve the reliability analysis problem with a 28-dimensional limit-state function. The estimation accuracy of both methods is quite similar with that of SuS (see Figures 5.19 and 6.12). Figure 6.13 shows the required number of calls to the true model and the percentage of saved calls in contrast to the conventional SuS. When achieving sufficient accuracy level, both approaches are able to save a large proportion of true model evaluations. Furthermore, KPLS-SuS outperforms MLS-SuS with two main reasons. For one thing, MLS employs a fixed low-order polynomial to construct surrogates, thus requiring more training samples for more complicated nonlinearity. By contrast,



a) *Number of calls to the true model*



b) *Percentage of calls saved*

Figure 6.13: *Number of true model evaluations for MLS-SuS and KPLS-SuS.*

kriging is more flexible since both polynomial trend and Gaussian process are utilized and the hyperparameters introduce additional degrees of freedom for the kriging model. For another, the built-in error estimation enables kriging-SuS to actively choose the best next training sample in a more efficient way. Though using fewer true model evaluations, as depicted in Figure 6.14, KPLS-SuS necessitates more training time than MLS-SuS when a large training set is necessary. This is because kriging needs to optimize the hyperparameters, and the execution time increases dramatically as the number of training samples grows.

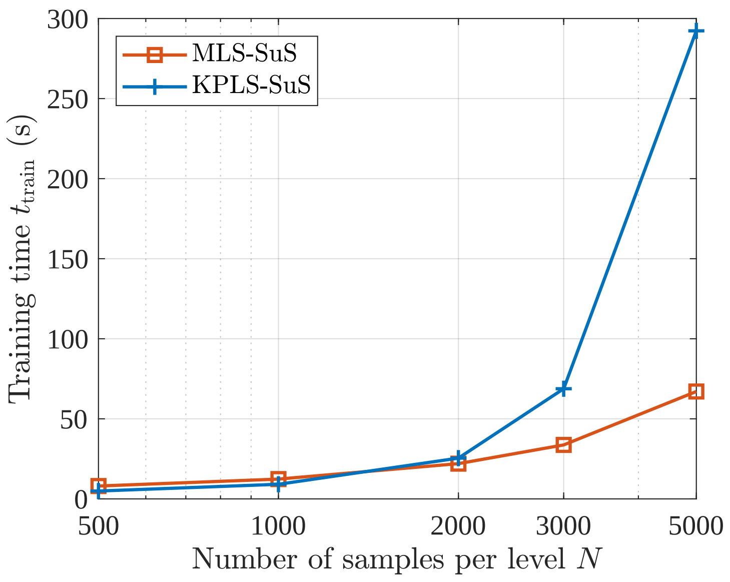


Figure 6.14: *Training time for MLS-SuS and KPLS-SuS.*

Table 6.7: *Recommended application contexts for different types of surrogate-accelerated SuS.*

Method	PS-SuS	MLS-SuS	Kriging-SuS
Nonlinearity	Weak	Weak/strong	Weak/strong
Dimension	Low	Low/high	Low/high
Computational demand of the true model evaluation	Low	Low	High

Based on the previous analysis and simulation results, PS-SuS is only suitable for problems with weak nonlinearity, low dimensions, and relatively cheap-to-evaluate models. However, MLS-SuS and kriging-SuS are not limited to weakly nonlinear or low-dimensional problems. It is recommended to use kriging-SuS if evaluating the true model is much more computationally expensive than solving the optimization problem, otherwise MLS-SuS is suggested. The recommended application contexts are summarized in Table 6.7.

6.5 Summary

This chapter first recalled the mathematical basics of the *kriging* surrogate modeling technique, as well as a dimensionality reduction strategy for kriging based on *partial least-squares* (PLS). Then, the *kriging-accelerated SuS* (kriging-SuS) approach was presented, for which an experimental design strategy was proposed to further improve the kriging modeling efficiency within the SuS framework. Finally, this chapter summarized the applicability of three types of surrogate models and the corresponding surrogate-based SuS approaches.

Chapter 7

Application to Flight Control Systems

In this chapter, the proposed surrogate-accelerated reliability analysis methods and the novel *reliability-based control optimization* (RBCO) framework are implemented to flight control systems for evaluating the failure probabilities and tuning the control parameters. The simulation results illustrate the accuracy and efficiency of the surrogate-assisted reliability assessment techniques as well as the ability of RBCO to directly guarantee the fulfillment of probabilistic requirements.

7.1 Low-Dimensional Problem

7.1.1 DA42 Longitudinal Model

7.1.1.1 Enhanced Plant

The control plant is an aircraft model (Diamond DA42) comprising longitudinal plant dynamics, actuator dynamics, structural mode, and notch filter, as depicted in Figure 7.1.

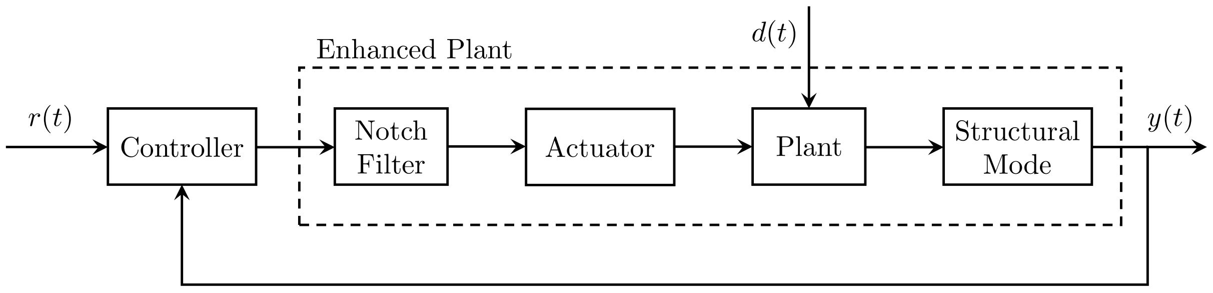


Figure 7.1: *The closed-loop system for DA42 longitudinal model.*

The short-period approximation of the longitudinal plant model is represented as

$$\begin{bmatrix} \dot{\alpha} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} Z_\alpha & Z_q + 1 \\ M_\alpha & M_q \end{bmatrix} \begin{bmatrix} \alpha \\ q \end{bmatrix} + \begin{bmatrix} Z_\eta \\ M_\eta \end{bmatrix} \eta, \tag{7.1}$$

where α, q, and η denote the angle of attack, the pitch rate, and the elevator deflection, respectively. The reference values of corresponding aerodynamic derivatives are: $Z_\alpha = -1.1933$, $Z_q = 0.0055$, $Z_\eta = 0.0291$, $M_\alpha = -12.4139$, $M_q = -2.8390$, and $M_\eta = 6.4818$.

The actuator dynamics is modeled as a second-order system with transfer function:

$$G_a(s) = \frac{\omega_a^2}{s^2 + 2\zeta_a \omega_a + \omega_a^2}, \tag{7.2}$$

with the natural frequency $\omega_a = 62.83$ rad/s and the damping ratio $\zeta_a = 0.71$.

The structural mode is simplified as a second-order, high-frequency, low-damped dynamics superimposed on the feedback signals:

$$G_s(s) = \frac{\omega_s^2}{s^2 + 2\zeta_s \omega_s + \omega_s^2}, \tag{7.3}$$

with the natural frequency $\omega_s = 25.07$ rad/s and the damping ratio $\zeta_s = 0.024$. The structural coupling can lead to a significant phase loss and an amplification around the mode frequency. The notch filter is designed to attenuate these effects as follows:

$$G_n(s) = \frac{s^2 + 2\zeta_n \omega_n + \omega_n^2}{s^2 + 2\zeta_d \omega_d + \omega_d^2}, \tag{7.4}$$

where $\omega_d = \omega_n = \omega_s$, $\zeta_n = \zeta_s$, and $\zeta_d = 0.30$.

7.1.1.2 Controller

A *proportional-integral-derivative* (PID) controller with feedforward control is applied as follows:

$$\dot{q}_{\text{cmd}} = k_H n_{z,\text{cmd}} + k_{n_z} n_z + k_I \int (n_{z,\text{cmd}} - n_z)\, \mathrm{d}t + k_q \omega_y, \tag{7.5}$$

where $\boldsymbol{k} = [k_H, k_{n_z}, k_I, k_q]^\mathsf{T}$ denotes the control parameters, $n_{z,\text{cmd}}$ is the command of the vertical load factor, $\dot{q}_{\text{cmd}}$ is the pitch acceleration command, and n_z and ω_y are the feedback signals of the vertical load factor and pitch rate, respectively.

7.1.1.3 Discrete Gust

To assess the system's gust-rejection performance, the standard "1−cosine" gust (represented by $d(t)$ in Figure 7.1) is introduced [155, p. 48]:

$$w_z = \begin{cases} 0, & x_g < 0, \\ \dfrac{v_g}{2}\left[1 - \cos\left(\dfrac{\pi x_g}{d_g}\right)\right], & 0 \le x_g < d_g, \\ v_g, & x_g \ge d_g, \end{cases} \tag{7.6}$$

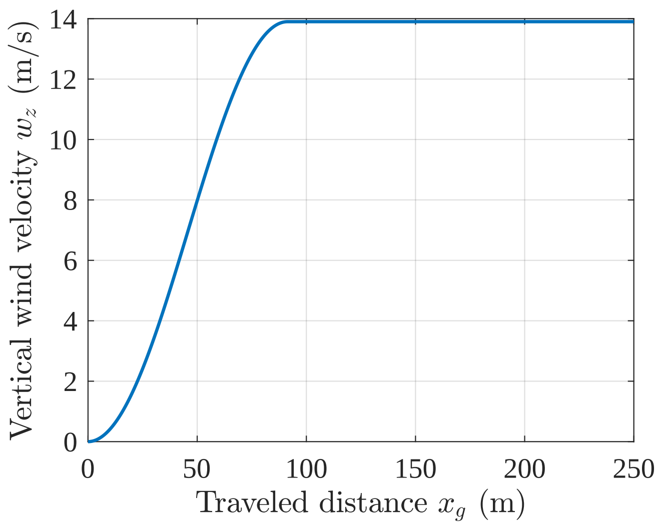


Figure 7.2: *Discrete gust with shape "1−cosine".*

where w_z is the vertical wind velocity of the gust, x_g is the horizontal distance traveled, d_g is the gust length, and v_g is the gust amplitude. In this application, $d_g = 91.4$ m and $v_g = 13.9$ m/s. The discrete gust is shown in Figure 7.2.

7.1.1.4 Parameter Uncertainties

Uncertainties in aerodynamic derivatives are considered and the relative values with respect to the reference values are assumed to be normally distributed:

$$\begin{aligned}\lambda_{M_\alpha} &= M_\alpha / M_{\alpha,\text{ref}} \sim \mathcal{N}(1, 0.2^2),\\ \lambda_{M_q} &= M_q / M_{q,\text{ref}} \sim \mathcal{N}(1, 0.2^2),\\ \lambda_{M_\eta} &= M_\eta / M_{\eta,\text{ref}} \sim \mathcal{N}(1, 0.2^2),\end{aligned} \tag{7.7}$$

and the vector of parameter uncertainties $\boldsymbol{\theta} = [\lambda_{M_\alpha}, \lambda_{M_q}, \lambda_{M_\eta}]^\mathsf{T}$.

7.1.1.5 Closed-Loop System

Finally, the closed-loop system in Figure 7.1 can be expressed as:

$$\begin{aligned}\dot{\boldsymbol{x}} &= \boldsymbol{A}_c \boldsymbol{x} + \boldsymbol{B}_c \boldsymbol{u},\\ \boldsymbol{y} &= \boldsymbol{C}_c \boldsymbol{x},\end{aligned} \tag{7.8}$$

where the system state $\boldsymbol{x}$ consists of the notch filter states, the states of actuator dynamics, the states of the short-period mode, the states due to the simplified structural mode on feedback signals, and the state introduced by the integration element of the PID controller. The input $\boldsymbol{u} = [r(t), d(t)]^\mathsf{T} = [n_{z,\text{cmd}}, w_z]^\mathsf{T}$ and the output $\boldsymbol{y} = n_z$. The matrices $\boldsymbol{A}_c$, $\boldsymbol{B}_c$ and $\boldsymbol{C}_c$ are functions of $\boldsymbol{\theta}$ and $\boldsymbol{k}$.

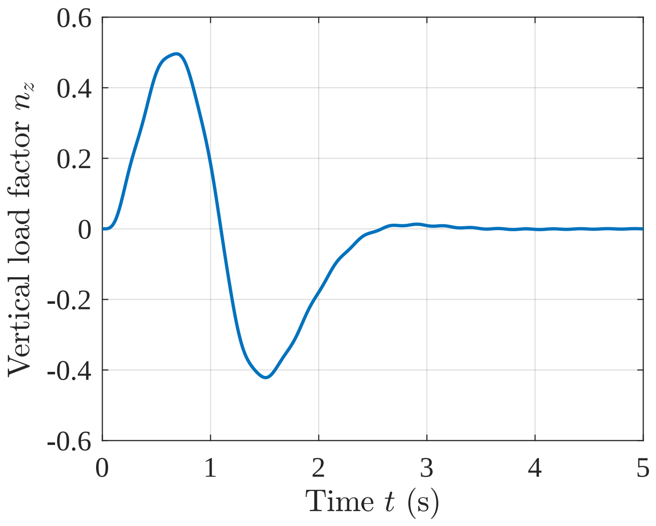


Figure 7.3: *Gust reaction given the nominal parameters.*

7.1.2 Failure Probability Estimation

7.1.2.1 Large Failure Probability Estimation

Given the nominal control parameters $\boldsymbol{k}_0 = [-1.2877, -0.9703, 3.2381, -5.6720]^\mathsf{T}$, the discrete gust w_z depicted in Figure 7.2, and the reference values of the aerodynamic derivatives, the gust reaction is shown in Figure 7.3. Considering the Gaussian-distributed uncertainties $\boldsymbol{\theta}$, we assume the probability that the *maximum negative deviation of the gust reaction* $y_{g,\min}$ is lower than -0.5 is of interest:

$$\mathbb{P}[g_0(\boldsymbol{\theta}, \boldsymbol{k}_0) < 0] = \mathbb{P}[y_{g,\min} < -0.5], \tag{7.9}$$

where the performance function

$$g_0(\boldsymbol{\theta}, \boldsymbol{k}_0) = y_{g,\min} + 0.5. \tag{7.10}$$

MCS and surrogate-based MCS methods are applied to assess the failure probability in Equation (7.9). Each method is performed through a number of 50 simulation runs. The simulation parameters for each approach are listed in Table 7.1, where N is the number of MCS samples, N_0 is the number of initial training samples, d is the polynomial expansion order, and $R(\boldsymbol{\theta}, \boldsymbol{\theta}'; \boldsymbol{\eta})$ is the correlation function. Adaptive PCE is implemented in PCE-MCS with the maximal order $d_{\max} = 6$, whereas a linear trend is utilized for kriging-MCS. In addition, the anisotropic Gaussian correlation function (denoted by "ani. Gaussian") in Equation (6.7) is adopted in kriging-MCS for the low-dimensional problem.

The simulation results are shown in Table 7.2, where the failure probability $\hat{P}_F$, the number of calls to the true model N_{call}, the total execution time t_{total}, and the classification error ε_c are the average values of the 50 results, whereas the c.o.v. estimate $\hat{c}_v$ is the

Table 7.1: *Simulation parameters for MCS and surrogate-based MCS methods.*

Method	N	N_0	d	$R(\boldsymbol{\theta}, \boldsymbol{\theta}'; \boldsymbol{\eta})$
MCS	2000	— —	— —	— —
PCE-MCS	2000	200	$1:6$	— —
MLS-MCS	2000	50	2	— —
kriging-MCS	2000	10	1	ani. Gaussian

Table 7.2: *Simulation results for the estimation of* $\mathbb{P}[y_{g,\min} < -0.5]$ *using MCS and surrogate-based MCS methods.*

Method	$\hat{P}_F$	$\hat{c}_v$	N_{call}	t_{total} (s)	ε_c
MCS	0.3042	0.0093	2000	12.2	— —
PCE-MCS	0.3046	0.0093	200	1.3	1×10^{-4}
MLS-MCS	0.3036	0.0088	71	0.9	— —
kriging-MCS	0.3045	0.0110	38	1.8	— —

sample c.o.v. of the 50 failure probability estimates. The total execution time t_{total} mainly consists of the time evaluating the true model and that training the surrogate model. The classification error ε_c is defined as the ratio of misclassified training samples (see Equation (4.68)).

Table 7.2 shows that these approaches result in similar probability estimates and estimation accuracy, but require different numbers of true model evaluations and execution time. PCE-MCS only employs the initial training points to build a global surrogate. By contrast, based on the initial training samples, the active learning strategies enable MLS-MCS and kriging-MCS to adaptively enrich the training set with valuable candidate samples until the surrogates are sufficiently accurate for the classification problem. Without ensuring the classification accuracy, PCE-MCS needs the error ε_c to assess the classification performance. The very small ε_c demonstrates that the nonlinearity of the true model is relatively weak such that a polynomial approximation is able to solve the classification problem properly. Because of the weak nonlinearity and the active learning strategies, MLS-MCS and kriging-MCS only need to add few additional training samples to the training set. In this example, MLS-MCS and kriging-MCS require far fewer true model evaluations than PCE-MCS. Though kriging-MCS has the lowest demand for calling the true model, it takes the longest training time. Considering both the model evaluation time and the training time, MLS-MCS is the most efficient method for this problem. Furthermore, all the surrogate-accelerated methods are much more efficient than the conventional MCS.

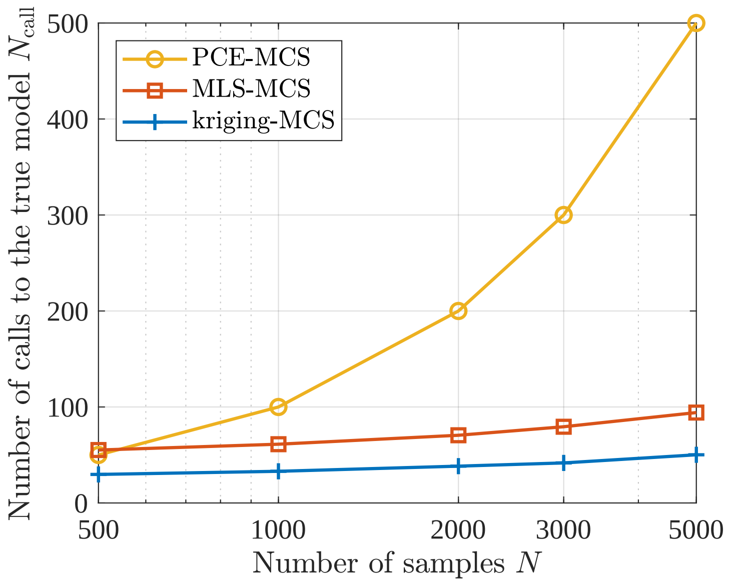


Figure 7.4: *Number of true model evaluations for estimating* $\mathbb{P}[y_{g,\min} < -0.5]$ *using surrogate-based MCS methods.*

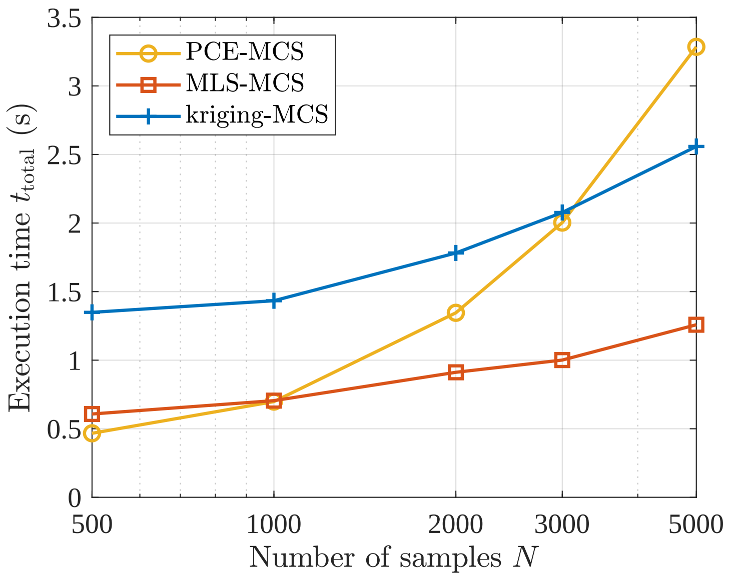


Figure 7.5: *Execution time for estimating* $\mathbb{P}[y_{g,\min} < -0.5]$ *using surrogate-based MCS methods.*

Given different N, the number of true model evaluations N_{call} and the execution time t_{total} required by these surrogate-based MCS techniques are compared in Figures 7.4 and 7.5, respectively. In this example, PCE-MCS always employs 10% of the MCS samples to construct the global surrogate. By contrast, the required N_{call} for MLS-SuS and kriging-SuS grows slowly with the increase of N due to the active learning strategies. Kriging-SuS saves more calls than MLS-SuS, but necessitates longer training time. In the end, kriging-SuS is generally the least efficient choice, whereas MLS-SuS finds the best balance between the training time and the model evaluation time.

7.1.2.2 Rare Failure Probability Estimation

In the following, the probabilities that the stability margins are lower than the corresponding limits are considered:

$$\begin{aligned} \mathbb{P}[g_1(\boldsymbol{\theta}, \boldsymbol{k}_0) < 0] &= \mathbb{P}[k_M < 6\,\mathrm{dB}], \\ \mathbb{P}[g_2(\boldsymbol{\theta}, \boldsymbol{k}_0) < 0] &= \mathbb{P}[\phi_M < 45^\circ], \end{aligned} \tag{7.11}$$

with performance functions

$$\begin{aligned} g_1(\boldsymbol{\theta}, \boldsymbol{k}_0) &= k_M - 6\,\mathrm{dB}, \\ g_2(\boldsymbol{\theta}, \boldsymbol{k}_0) &= \phi_M - 45^\circ, \end{aligned} \tag{7.12}$$

where k_M and ϕ_M are the *gain margin* and the *phase margin*, respectively.

SuS and surrogate-based SuS techniques are implemented to evaluate the failure probabilities in Equation (7.11). As in Section 7.1.2.1, each approach is conducted through a number of 50 simulation runs. The simulation parameters for each method are listed in Table 7.3. Here, N denotes the number of samples at each subset level, p_0 represents the conditional probability for SuS, and p_d is the percentage of discarded training samples for kriging-SuS (see Section 6.3.3). The adaptive polynomial trend with the maximal expansion order $d_{\max} = 4$ is used for kriging-SuS.

The simulation results for the estimation of $\mathbb{P}[k_M < 6\,\mathrm{dB}]$ are shown in Table 7.4. The c.o.v. bounds $\hat{c}_{v,\mathrm{lb}}$ and $\hat{c}_{v,\mathrm{ub}}$ are the average values of the 50 approximations. The probability estimates and estimation accuracy obtained by surrogate-assisted methods

Table 7.3: *Simulation parameters for SuS and surrogate-based SuS methods.*

Method	N	p_0	N_0	d	$R(\boldsymbol{\theta}, \boldsymbol{\theta}'; \boldsymbol{\eta})$	p_d
SuS	2000	0.1	——	——	——	——
PS-SuS	2000	0.1	240	$1:6$	——	——
MLS-SuS	2000	0.1	50	2	——	——
kriging-SuS	2000	0.1	10	$1:4$	ani. Gaussian	0.5

Table 7.4: *Simulation results for the estimation of* $\mathbb{P}[k_M < 6\,\mathrm{dB}]$ *using SuS and surrogate-based SuS methods.*

Method	$\hat{P}_F$	$[\hat{c}_{v,\mathrm{lb}}, \hat{c}_{v,\mathrm{ub}}]$	$\hat{c}_v$	N_{call}	t_{total} (s)	ε_c
SuS	9.55×10^{-7}	$[0.32, 0.78]$	0.52	13232	94.3	——
PS-SuS	9.75×10^{-7}	$[0.31, 0.77]$	0.63	1331	10.7	4.25×10^{-5}
MLS-SuS	8.51×10^{-7}	$[0.32, 0.78]$	0.53	722	13.8	——
kriging-SuS	9.77×10^{-7}	$[0.32, 0.77]$	0.52	87	5.5	——

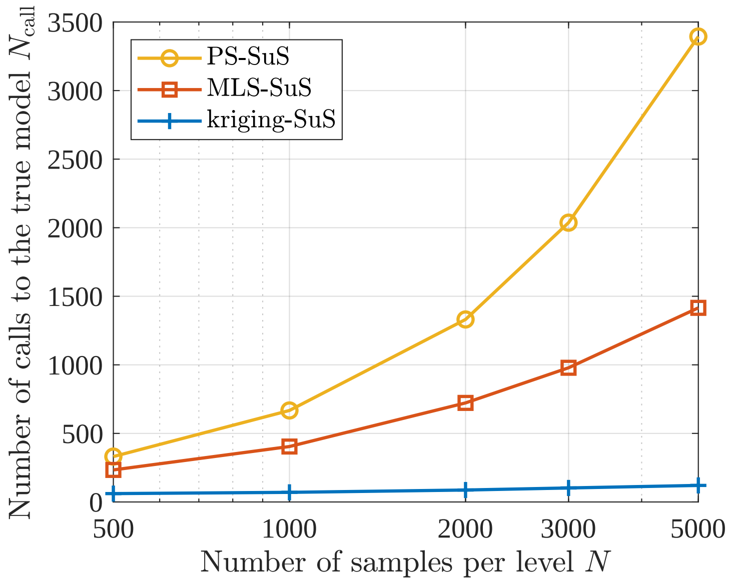


Figure 7.6: *Number of true model evaluations for estimating* $\mathbb{P}[k_M < 6\,\mathrm{dB}]$ *using surrogate-based SuS methods.*

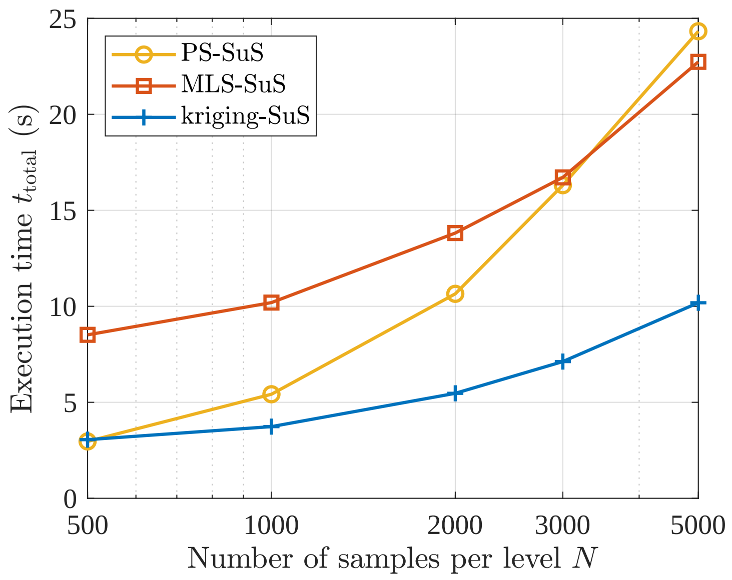


Figure 7.7: *Execution time for estimating* $\mathbb{P}[k_M < 6\,\mathrm{dB}]$ *using surrogate-based SuS methods.*

are similar with those obtained by SuS. The number of true model evaluations N_{call} and the execution time t_{total} required by these surrogate-assisted SuS techniques are further compared in Figures 7.6 and 7.7, where different N are exploited. Compared with SuS, PS-SuS saves about 90% true model evaluations. More calls to the true model are saved by MLS-SuS and kriging-SuS because of the active learning strategies. In addition, kriging-SuS outperforms MLS-SuS since kriging attempts to construct the surrogate model in an optimal way. All the surrogate-accelerated SuS methods are far more efficient than the conventional SuS, and kriging-SuS is the best choice among them.

Table 7.5: *Simulation results for the estimation of $\mathbb{P}[\phi_M < 45°]$ using SuS and surrogate-based SuS methods.*

Method	$\hat{P}_F$	$[\hat{c}_{v,\mathrm{lb}}, \hat{c}_{v,\mathrm{ub}}]$	$\hat{c}_v$	N_{call}	t_{total} (s)	ε_c
SuS	3.33×10^{-5}	$[0.25, 0.55]$	0.46	9940	67.9	— —
PS-SuS	3.29×10^{-5}	$[0.26, 0.55]$	0.40	1014	8.1	3.53×10^{-3}
MLS-SuS	3.72×10^{-5}	$[0.26, 0.54]$	0.39	727	12.3	— —
kriging-SuS	3.39×10^{-5}	$[0.26, 0.55]$	0.42	226	25.9	— —

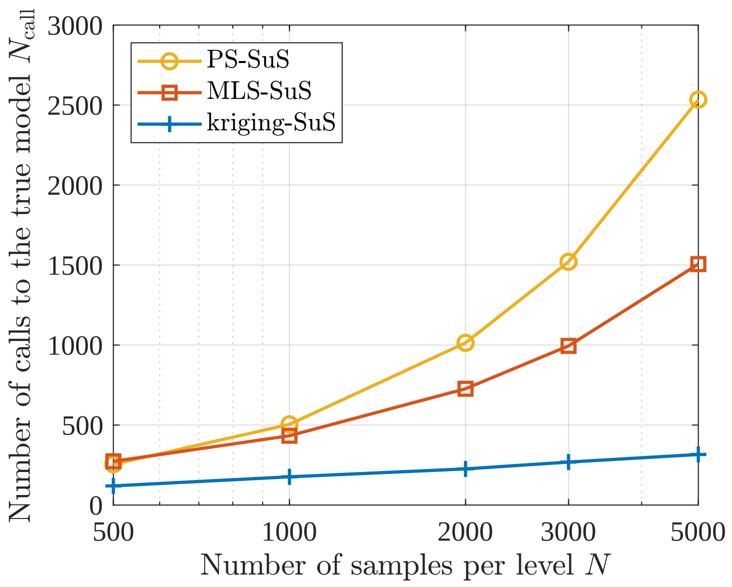


Figure 7.8: *Number of true model evaluations for estimating $\mathbb{P}[\phi_M < 45°]$ using surrogate-based SuS methods.*

Similarly, given the parameters in Table 7.3, the simulation results for the estimation of $\mathbb{P}[\phi_M < 45°]$ are listed in Table 7.5. Again, the proposed surrogate-assisted approaches obtain similar probability estimates and estimation accuracy as the conventional SuS. The classification error for PS-SuS in Table 7.5 is rather larger than that in Table 7.4, indicating that the nonlinearity of $g_2(\boldsymbol{\theta}, \boldsymbol{k}_0)$ is more difficult to deal with. This is verified by the fact that MLS-SuS and kriging-SuS need more additional training samples for estimating $\mathbb{P}[\phi_M < 45°]$ although fewer subset levels are required. For different N, the number of true model evaluations N_{call} and the execution time t_{total} required by these surrogate-assisted SuS techniques are compared in Figures 7.8 and 7.9. In this example, among these methods, kriging-SuS requires the fewest calls to the true model. However, using a smaller training set does not necessarily mean it takes less execution time. Kriging-SuS needs far more training time than MLS-SuS. In the end, it is less efficient than MLS-SuS. Nonetheless, in the case where evaluating the true model is much more time-consuming than training the kriging model, kriging-SuS would be the most attractive choice.

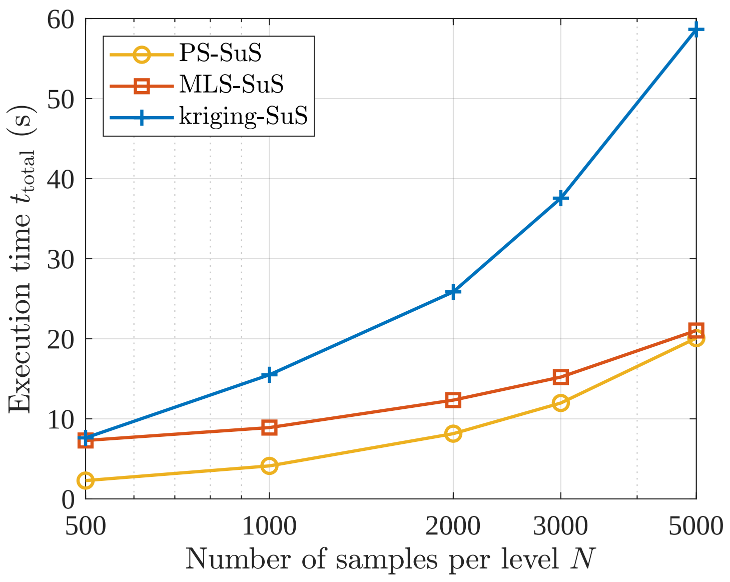


Figure 7.9: *Execution time for estimating* $\mathbb{P}[\phi_M < 45°]$ *using surrogate-based SuS methods.*

To sum up, all the surrogate-based SuS approaches are able to provide sufficiently accurate probability estimates and are more efficient than the conventional SuS. Users may choose the most appropriate method according to the complexity of the reliability analysis problem.

7.1.3 Control Parameter Optimization

The proposed surrogate-accelerated reliability assessment techniques have been proved to be accurate and efficient. In this subsection, these methods and the RBCO approach designed in Chapter 3 are implemented to evaluate the failure probability and search for control parameters satisfying rare-event probabilistic requirements.

Assume that the goal of the control design optimization is to improve the disturbance rejection behavior while guaranteeing the tracking performance and satisfying the stability requirements. The objective function and probabilistic constraints are defined in Table 7.6.

Table 7.6: *Objective function and probabilistic constraints for the RBCO problem.*

Specification	Performance function	Objective function or constraint
$y_{g,\min}$	$g_0(\boldsymbol{\theta}, \boldsymbol{k}) = y_{g,\min} + 0.5$	$\mathbb{P}[y_{g,\min} < -0.5]$
k_M	$g_1(\boldsymbol{\theta}, \boldsymbol{k}) = k_M - 6\,\text{dB}$	$(1 + 3c_{v1})\,\mathbb{P}[k_M < 6\,\text{dB}] \leq 10^{-6}$
ϕ_M	$g_2(\boldsymbol{\theta}, \boldsymbol{k}) = \phi_M - 45°$	$(1 + 3c_{v2})\,\mathbb{P}[\phi_M < 45°] \leq 10^{-6}$
$\sigma\%$	$g_3(\boldsymbol{\theta}, \boldsymbol{k}) = 20\% - \sigma\%$	$\mathbb{P}[\sigma\% > 20\%] \leq 0.1$
t_r	$g_4(\boldsymbol{\theta}, \boldsymbol{k}) = 1\,\text{s} - t_r$	$\mathbb{P}[t_r > 1\,\text{s}] \leq 0.1$

Table 7.7: *Simulation results for the estimation of failure probabilities using SuS.*

Failure probability	$\hat{P}_F$	$[\hat{c}_{v,\text{lb}}, \hat{c}_{v,\text{ub}}]$	N_{call}	t_{total} (s)
$\mathbb{P}[y_{g,\min} < -0.5]$	0.119	——	2000	11.8
$\mathbb{P}[k_M < 6\,\text{dB}]$	5.63×10^{-8}	$[0.34, 0.93]$	15882	108.7
$\mathbb{P}[\phi_M < 45°]$	2.60×10^{-8}	$[0.33, 0.91]$	15846	97.6
$\mathbb{P}[\sigma\% > 20\%]$	6.10×10^{-2}	$[0.08, 0.10]$	3986	23.1
$\mathbb{P}[t_r > 1\,\text{s}]$	8.34×10^{-2}	$[0.07, 0.09]$	3986	23.0

Table 7.8: *Simulation results for the estimation of failure probabilities using surrogate-based SuS methods.*

Failure probability	Method	$\hat{P}_F$	$[\hat{c}_{v,\text{lb}}, \hat{c}_{v,\text{ub}}]$	N_{call}	t_{total} (s)
$\mathbb{P}[y_{g,\min} < -0.5]$	MLS-SuS	0.119	——	64	1.0
$\mathbb{P}[k_M < 6\,\text{dB}]$	kriging-SuS	8.78×10^{-8}	$[0.36, 0.94]$	86	10.7
$\mathbb{P}[\phi_M < 45°]$	MLS-SuS	4.91×10^{-8}	$[0.34, 0.92]$	1459	24.7
$\mathbb{P}[\sigma\% > 20\%]$	MLS-SuS	6.18×10^{-2}	$[0.08, 0.10]$	188	2.9
$\mathbb{P}[t_r > 1\,\text{s}]$	MLS-SuS	8.57×10^{-2}	$[0.07, 0.08]$	92	2.5

Here, $\sigma\%$ and t_r are the *overshoot* and the *rise time* of the step response. The rise time is defined as the time required for the response to rise from 0% to 80% of its final value. Besides, c_{v1} and c_{v2} represent the coefficients of variation of the probability estimates $\mathbb{P}[k_M < 6\,\text{dB}]$ and $\mathbb{P}[\phi_M < 45°]$, respectively. To ensure the fulfillment of the stability requirements, the 3-σ upper bounds of the probability estimates are considered to satisfy the rare-event chance constraints.

Based on the efficiency comparisons in previous subsection, kriging-SuS is employed to estimate $\mathbb{P}[k_M < 6\,\text{dB}]$ and the rest failure probabilities are assessed by MLS-SuS. $N = 2000$ samples are generated at each subset level. Solving the RBCO problem in Equation (3.2) with the *Matlab* global solver *surrogateopt* [156] leads to the optimal design $\boldsymbol{k}^* = [-2.8397, -1.0703, 2.9698, -3.3762]^{\mathsf{T}}$ for the controller in Equation (7.5). Note that the use of surrogate-assisted SuS methods within the RBCO problem greatly reduces the computational expense. Given the designed control parameters, the failure probability estimation results obtained by SuS and surrogate-accelerated SuS are listed in Tables 7.7 and 7.8, respectively. The SuS results are regarded as references. The results of the surrogate-based SuS approaches, i.e., MLS-SuS and kriging-SuS, match well with the reference results, but a large proportion of true model evaluations and the execution time are saved. Moreover, the simulation results show that the probabilistic constraints given in Table 7.6 are fulfilled directly by the RBCO framework.

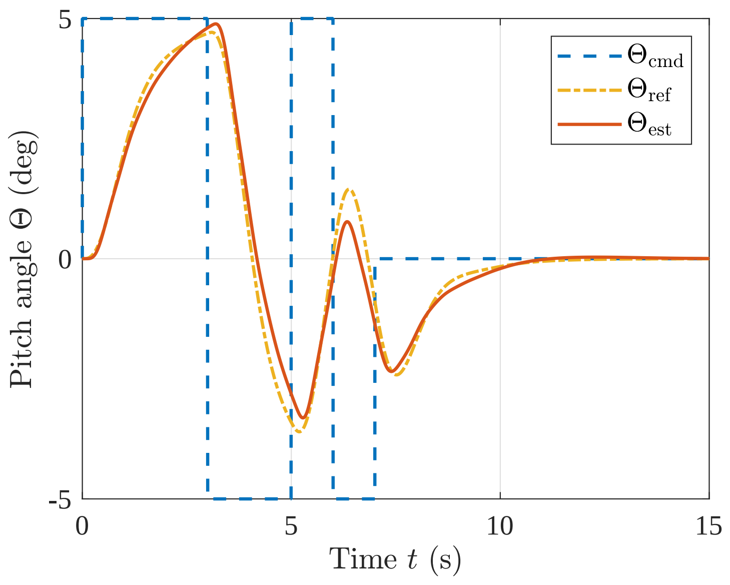


Figure 7.10: *Pitch angle tracking results.*

7.2 High-Dimensional Problem

7.2.1 High-Fidelity eVTOL Aircraft Model

An unmanned *electric vertical take-off and landing* (eVTOL) demonstrator aircraft has been developed and numerically modeled at the *Institute of Flight System Dynamics* (FSD) of the *Technical University of Munich* (TUM). In this section, the target is to assess the attitude tracking performance of the *critical flight control law* (CFCL) for the high-fidelity eVTOL model developed based on the work in [157, 158].

Figure 7.10 shows the reference of pitch angle Θ_{ref} and the estimated pitch angle response Θ_{est} given the pitch angle command Θ_{cmd}. In this application, the pitch angle tracking performance is evaluated by the *root mean square* (RMS) error between Θ_{ref} and Θ_{est}

$$\mathrm{RMS}(\Theta_{\text{ref}} - \Theta_{\text{est}}). \tag{7.13}$$

The failure event is defined as

$$\mathrm{RMS}(\Theta_{\text{ref}} - \Theta_{\text{est}}) > 0.45°. \tag{7.14}$$

Considering a number of 42 Gaussian-distributed uncertain parameters $\boldsymbol{\theta}$, it is assumed that the following failure probability is of interest:

$$\mathbb{P}[g(\boldsymbol{\theta}) < 0] = \mathbb{P}[\mathrm{RMS}(\Theta_{\text{ref}} - \Theta_{\text{est}}) > 0.45°], \tag{7.15}$$

with performance function

$$g(\boldsymbol{\theta}) = 0.45° - \mathrm{RMS}(\Theta_{\text{ref}} - \Theta_{\text{est}}). \tag{7.16}$$

7.2.2 Rare Failure Probability Estimation

In practice, the simulation of the high-fidelity eVTOL model is accelerated by the *Simulation Accelerator* toolbox [159]. As a consequence, one simulation run takes about 2.8 seconds. According to the recommendations in Table 6.7, due to the unknown nonlinearity of the performance function and its relatively low computational demand, the MLS-SuS approach is implemented to estimate the failure probability in Equation (7.15).

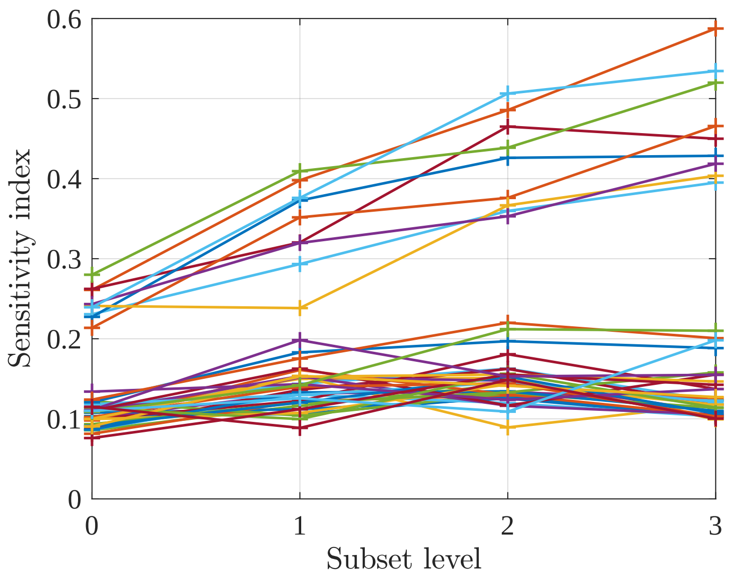
Both the direct MLS-SuS without dimensionality reduction and the two-stage MLS-SuS (with dimensionality reduction) are implemented, denoted by “MLS-SuS1” and “MLS-SuS2”, respectively. Besides, the conventional SuS technique is applied for reference. The simulation parameters for each method are listed in Table 7.9. Here, N denotes the number of samples at each subset level, p_0 represents the conditional probability for SuS, d is the polynomial expansion order, and N_0 is the number of samples at each subset level for the pure SuS at the first stage of MLS-SuS2.

The simulation results are presented in the following. Figure 7.11a shows the sensitivity analysis results of SuS, whereas Figure 7.11b shows those of the first stage of MLS-SuS2. For SuS, $N = 3000$ samples are employed at each subset level. By contrast, only $N_0 = 100$ samples per level are used for the pure SuS at the first stage of MLS-SuS2. In both subfigures, each line illustrates the relative importance of each uncertain parameter. Similar as SuS, the first stage of MLS-SuS2 is able to clearly detect the most influential 9 variables. However, the latter approach only needs 400 samples in total to achieve that. At the second stage of MLS-SuS2, only the detected 9 variables are regarded as sensitive parameters and the hybrid kernel in Equation (5.40) is applied. This results in a feature reduction from 946 to 88.

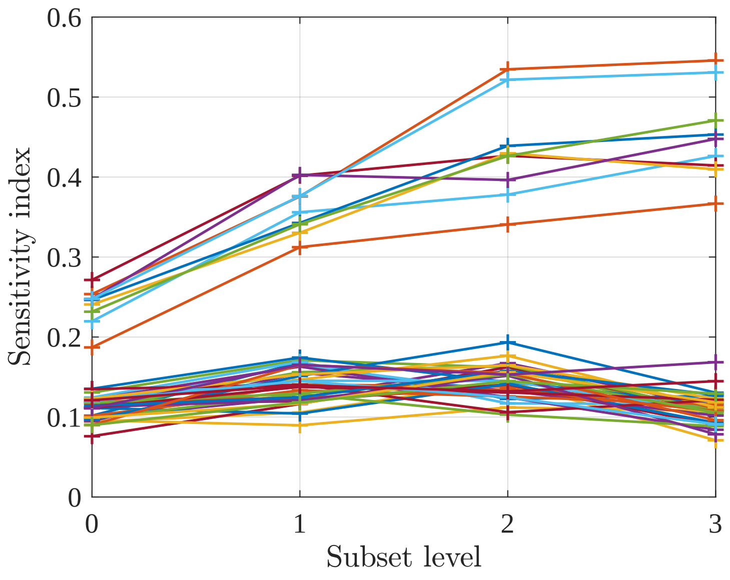
Table 7.10 lists the simulation results of these three methods. Both MLS-SuS1 and MLS-SuS2 save a large number of calls to the true model N_{call}, and the latter outperforms the former. Compared with MLS-SuS2, the number of basis functions p for MLS-SuS1 is much larger, which generally requires more samples to build the surrogate model. In this example, it is time-consuming to evaluate the system performance. The elapsed time for evaluating the performance function t_{call} is proportional to the number of calls N_{call}. Additionally, MLS-SuS1 takes 1.11×10^4 seconds (about 3 hours) to iteratively update

Table 7.9: *Simulation parameters for SuS and MLS-SuS methods.*

Method	N	p_0	d	N_0
SuS	3000	0.1	— —	— —
MLS-SuS1	3000	0.1	2	— —
MLS-SuS2	3000	0.1	1 : 2	100



a) *SuS results*



b) *MLS-SuS (Stage 1) results*

Figure 7.11: *Simulation results for the sensitivity analysis using SuS and MLS-SuS methods.*

Table 7.10: *Simulation results for the failure probability estimation using SuS and MLS-SuS methods.*

Method	$\hat{P}_F$	$[\hat{c}_{v,\mathrm{lb}}, \hat{c}_{v,\mathrm{ub}}]$	p	N_{call}	t_{call} (s)	t_{train} (s)	t_{total} (s)
SuS	1.63×10^{-4}	$[0.17, 0.33]$	——	12000	3.35×10^4	——	3.35×10^4
MLS-SuS1	2.35×10^{-4}	$[0.16, 0.31]$	946	4932	1.38×10^4	1.11×10^4	2.49×10^4
MLS-SuS2	1.93×10^{-4}	$[0.16, 0.32]$	88	3066	8.57×10^3	64	8.63×10^3

the surrogate, whereas MLS-SuS2 only needs 64 seconds (about 1 minute). The huge gap is due to the difference of surrogate complexities. With simplified expansion (the number of basis functions is reduced from 946 to 88), MLS-SuS2 is far more efficient than MLS-SuS1 without losing the accuracy of failure probability estimation. The total simulation time t_{total} is generally a sum of the time evaluating the true model and the training time. Although MLS-SuS1 is still more efficient than the conventional SuS, it suffers from the curse of dimensionality. The two-stage MLS-SuS provides a successful way to alleviate this problem.

7.3 Summary

This chapter implemented the proposed surrogate-accelerated reliability analysis techniques and *reliability-based control optimization* (RBCO) framework to flight control systems. The failure probabilities were estimated with a sufficient accuracy in an efficient way and the control parameters were tuned to meet the probabilistic requirements with a formal guarantee.

Chapter 8

Conclusions and Perspectives

8.1 Conclusions

In this thesis, we investigated the surrogate-assisted reliability analysis and design problems for flight control systems. Here we conclude the results of this thesis by chapters.

Aiming at satisfying the probabilistic requirements specified in the certification specifications in aviation, Chapter 3 proposed a novel control design optimization framework called *reliability-based control optimization* (RBCO). The presented framework consists of two loops: the inner loop evaluates the failure probabilities whereas the outer loop explores the design space using optimization techniques. It is a verification-driven approach that ensures the probabilistic requirements are directly fulfilled. In comparison with conventional control design methods which treat chance constraints in a conservative manner, this framework is able to enlarge the design space and explore further performance.

To accurately assess the probabilities of interest, simulation methods including *Monte Carlo simulation* (MCS) and *subset simulation* (SuS) were employed without making any hypothesis on the nonlinearity of the performance function. However, obtaining sufficiently precise estimations requires a large number of performance function evaluations. To reduce this demand, this thesis incorporated surrogate modeling techniques into the simulation approaches by replacing some true model evaluations with the predictions of the surrogate model.

Chapter 4 exploited two polynomial-based global surrogates, namely, *response surface method* (RSM) and *polynomial chaos expansion* (PCE), to achieve the uncertainty quantification task. Firstly, the uncertainty propagation schemes estimating the statistical characteristics (including moments, failure probability, and PDF) of system outputs were summarized. The differences between these two surrogates stem from different requirements for system inputs. Compared with RSM, PCE requires the knowledge of the input distribution to find orthogonal polynomial bases, thus guaranteeing the convergence of the approximation and being able to estimate statistical moments analytically. However, the isoprobabilistic transformation required by PCE may increase the complexity of the transformed system. Subsequently, we introduced the *polynomial surrogate-based SuS* (PS-SuS) approach that combines global surrogates with SuS. Adaptive PCE or RSM searching for the optimal expansion order was applied to progressively refine the surrogate at each subset level. To detect both the local variability and the global trend, an experimental design strategy was designed to choose the most valuable training samples from the training set to update the surrogate model. Moreover, the classification error was defined to monitor the quality of the estimation results for the PS-SuS method. The accuracy and efficiency of the proposed approach were demonstrated by low-dimensional illustrative examples.

Though the global surrogates are easy to implement, they may suffer from large estimation errors for highly nonlinear applications. To alleviate this problem, Chapter 5 employed a local surrogate model called *moving least-squares* (MLS) to accelerate the simulation-based reliability analysis methods. First, the *MLS-accelerated SuS* (MLS-SuS) strategy was introduced. After that, surrogate errors were approximated based on *leave-one-out* (LOO) cross validation. With the error approximation, an active learning strategy which enriches the training set with potentially misclassified samples was proposed to properly classify the generated samples. The numerical examples showed that, in comparison with SuS, the proposed strategy saves a large proportion of calls to the true model while providing comparative estimation accuracy. Furthermore, a dimensionality reduction strategy was developed for MLS-SuS. Sensitivity analysis was first conducted to rank the variables according to their importance level with respect to the failure event. According to the ranking, unnecessary expansion items were then filtered out, thus reducing the dimensionality of the feature space. For the flexible calculation of the expansion coefficients, *kernel ridge regression* (KRR) was applied, and the hybrid kernel tailored to the dimensionality reduction strategy was introduced. The simulation results suggested that this strategy reduces the required training effort without sacrificing the estimation accuracy.

Instead of leveraging surrogates that are based on the rigid polynomial expression, Chapter 6 utilized a more flexible surrogate, called *kriging*, to mimic the true model behavior during the SuS procedure. This chapter integrated the commonly implemented active learning kriging into SuS, resulting the *kriging-accelerated SuS* (kriging-SuS) approach. However, the conventional kriging suffers from the curse of dimensionality. Aiming at overcoming this challenge, *partial least-squares-based kriging* (KPLS) was implemented to reduce the number of unknown hyperparameters, which consequently saves the computational cost for optimizing hyperparameters. In order to enhance the modeling efficiency, the adaptive PCE was implemented to identify the best polynomial trend. Considering that the kriging training time grows dramatically with the enrichment of the training set, an experimental design strategy was proposed to mitigate this issue by selecting influential training samples from the full training set for the refinement of the kriging surrogate. Illustrative examples demonstrated the accuracy and efficiency of the proposed kriging-SuS method. Finally, this chapter compared the performance of three types of surrogate modeling techniques and the corresponding surrogate-based SuS approaches. The applicability of RSM or PCE is limited to weakly nonlinear and low-dimensional problems. Compared with MLS, kriging is more accurate and requires fewer number of true model evaluations, but necessitates longer training time. Therefore, it is suggested to apply kriging-SuS if the evaluation of the true model is more computationally expensive than the optimization problem, otherwise MLS-SuS is recommended.

Chapter 7 is focused on the implementation of the proposed surrogate-based reliability analysis methods and RBCO framework to flight control systems. The simulation results demonstrated that the surrogate-assisted SuS approaches are able to provide sufficiently accurate failure probability estimates and are more efficient than the conventional SuS. The computational expense for solving the RBCO problem is thus drastically reduced by these surrogate-based methods. Furthermore, the probabilistic requirements are satisfied directly by the RBCO framework.

8.2 Perspectives

This section discusses the potential future work that may enhance the proposed methods.

It was shown in the thesis that the proposed RBCO framework can be used to search for control parameters that satisfy the probabilistic requirements with a formal guarantee. However, the reliability analysis procedure needs to be conducted repeatedly in the optimization loop, thus requiring high computational expense. To reduce this demand, one can consider adopting variable accuracy levels for probability estimation. Specifically, if the probability to be estimated differs from the target probability by orders of magnitude, an estimation with a large coefficient of variation (c.o.v.) would be sufficient. When the probability to be estimated gets close to the target probability, it is necessary to increase the estimation accuracy level accordingly.

The proposed RBCO framework is inspired by the two-level *reliability-based design optimization* (RBDO) techniques. In comparison with the two-level methods for RBDO, there exist various mono-level and decoupled approaches in the literature [106, 107] which can be used to achieve higher efficiency by reformulating and decoupling the two-level formulation. However, they both suffer from multiple failure domains and strong nonlinearities in the limit-state functions. For future work, one can further develop a scheme combining these three types of approaches appropriately by making a tradeoff between accuracy and efficiency for the RBCO problem.

Since the active learning strategies in the MLS-SuS and kriging-SuS approaches seek to classify samples with high confidence, the c.o.v. approximation for the conventional SuS method is still exploited for these proposed methods. Simulation results demonstrate these approaches obtains similar accuracy level for probability estimation as the conventional SuS method. In order to achieve a more reliable accuracy estimation, the classification error can be integrated into the c.o.v. approximation for the surrogate-based SuS techniques.

During the SuS procedure, the domain of interest progressively converges to the targeting failure domain. At a certain subset level, the constructed surrogate model needs to first identify whether or not the samples are lying in the current intermediate failure domain, and then predict the model response for those samples that are within this domain. An alternative approach would be constructing two surrogates: one for distinguishing samples whether they are located in the domain of interest, and the other for mimicking the model behaviors at the current level. This framework builds separated surrogates for different tasks, which can be helpful to improve the efficiency of surrogate-assisted SuS techniques.

Appendix A

Commonly Used Distributions

Table A.1 lists several widely used continuous distributions.

Table A.1: *Commonly used continuous distributions.*

Distribution	PDF	Support	Parameters
Uniform $\mathcal{U}(a,b)$	$f(\theta) = \frac{1}{b-a}$	$a \leq \theta \leq b$	$-\infty < a < \infty$ $a < b < \infty$
Gaussian or Normal $\mathcal{N}(\mu,\sigma)$	$f(\theta) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(\theta-\mu)^2}{2\sigma^2}}$	$-\infty < \theta < \infty$	$-\infty < \mu < \infty$ $\sigma > 0$
Lognormal $\mathcal{LN}(\mu,\sigma)$	$f(\theta) = \frac{1}{\theta\sigma\sqrt{2\pi}} e^{-\frac{(\ln\theta-\mu)^2}{2\sigma^2}}$	$0 < \theta < \infty$	$-\infty < \mu < \infty$ $\sigma > 0$
Exponential $\text{Exp}(\lambda)$	$f(\theta) = \lambda e^{-\lambda\theta}$	$0 \leq \theta < \infty$	$\lambda > 0$
Gamma $\mathcal{G}(\alpha,\beta)$	$f(\theta) = \frac{\theta^{\alpha} e^{-\theta/\beta}}{\beta^{\alpha+1}\Gamma(\alpha+1)}$	$0 \leq \theta < \infty$	$\alpha > -1$ $\beta > 0$
Beta $\mathcal{B}(\alpha,\beta)$	$f(\theta) = \frac{\Gamma(\alpha+\beta+2)(1-\theta)^{\alpha}(1+\theta)^{\beta}}{2^{\alpha+\beta+1}\Gamma(\alpha+1)\Gamma(\beta+1)}$	$-1 \leq \theta \leq 1$	$\alpha > -1$ $\beta > -1$

Appendix B

Orthogonal Polynomials

B.1 Legendre Polynomials

Legendre polynomials $P_i(\xi)$ [49] are defined over $[-1, 1]$ and orthogonal with respect to the uniform distribution:

$$f(\xi) = \frac{1}{2}, \quad \xi \in [-1, 1]. \tag{B.1}$$

They satisfy the recurrence relation:

$$\begin{aligned} P_{-1}(\xi) &= 0, \\ P_0(\xi) &= 1, \\ (i+1)P_{i+1}(\xi) &= (2i+1)\xi P_i(\xi) - iP_{i-1}(\xi), \quad i \in \mathbb{N}_0, \end{aligned} \tag{B.2}$$

and

$$\langle P_i(\xi), P_j(\xi) \rangle = \int_{-1}^{1} P_i(\xi) P_j(\xi) f(\xi) \mathrm{d}\xi = \frac{1}{2i+1} \delta_{ij}. \tag{B.3}$$

This indicates

$$\gamma_i = \mathbb{E}\left[P_i(\xi)^2\right] = \frac{1}{2i+1}. \tag{B.4}$$

The first few Legendre polynomials are given in Table B.1 and plotted in Figure B.1.

Table B.1: *First few Legendre polynomials.*

i	$P_i(\xi)$	$\mathbb{E}[P_i(\xi)^2]$
0	1	1
1	ξ	1/3
2	$\frac{1}{2}(3\xi^2 - 1)$	1/5
3	$\frac{1}{2}(5\xi^3 - 3\xi)$	1/7
4	$\frac{1}{8}(35\xi^4 - 30\xi^2 + 3)$	1/9

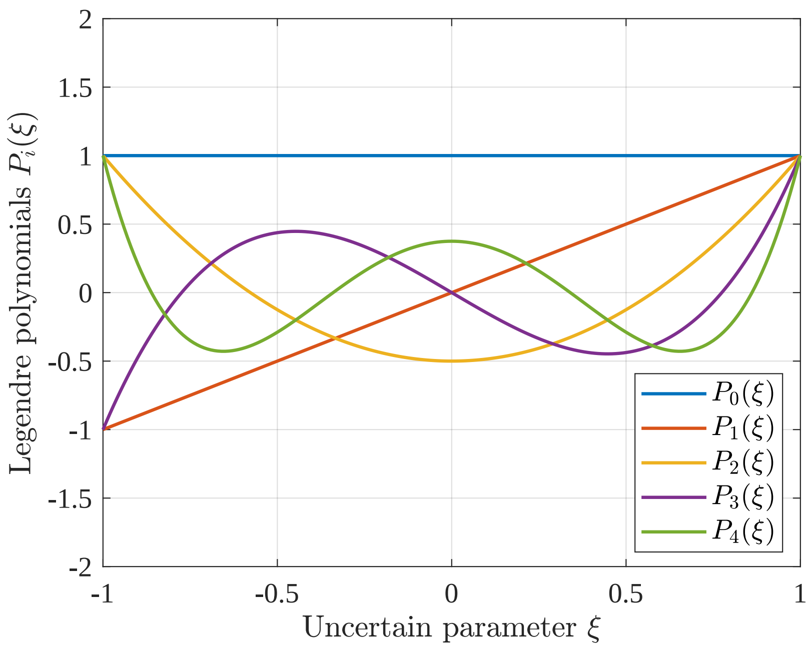


Figure B.1: *First few Legendre polynomials.*

B.2 Laguerre Polynomials

Laguerre polynomials $L_i^\alpha(\xi)$ [49] are defined over $[0,\infty)$ and orthogonal with respect to the gamma distribution (see Appendix A) with $\beta = 1$:

$$f(\xi) = \frac{\xi^\alpha e^{-\xi}}{\Gamma(\alpha+1)}, \quad \xi \in [0,\infty),\ \alpha > -1. \tag{B.5}$$

They satisfy the recurrence relation:

$$\begin{aligned} L_{-1}^\alpha(\xi) &= 0, \\ L_0^\alpha(\xi) &= 1, \\ (i+1)L_{i+1}^\alpha(\xi) &= (2i+\alpha+1-\xi)L_i^\alpha(\xi) - (i+\alpha)L_{i-1}^\alpha(\xi), \quad i \in \mathbb{N}_0, \end{aligned} \tag{B.6}$$

and

$$\left\langle L_i^\alpha(\xi), L_j^\alpha(\xi) \right\rangle = \int_0^\infty L_i^\alpha(\xi)L_j^\alpha(\xi)f(\xi)\mathrm{d}\xi = \frac{(\alpha+1)_i}{i!}\delta_{ij}, \tag{B.7}$$

where

$$(\alpha)_i = \frac{\Gamma(\alpha+i)}{\Gamma(\alpha)} \tag{B.8}$$

is the *Pochhammer symbol.* This indicates

$$\gamma_i = \mathbb{E}\left[L_i^\alpha(\xi)^2\right] = \frac{(\alpha+1)_i}{i!}. \tag{B.9}$$

Given $\alpha = 0$, the first few Laguerre polynomials are listed in Table B.2 and plotted in Figure B.2.

Table B.2: *First few Laguerre polynomials given $\alpha = 0$.*

i	$L_i^\alpha(\xi)$	$\mathbb{E}\left[L_i^\alpha(\xi)^2\right]$
0	1	1
1	$-\xi + 1$	1
2	$\frac{1}{2}(\xi^2 - 4\xi + 2)$	1
3	$\frac{1}{6}(-\xi^3 + 9\xi^2 - 18\xi + 6)$	1
4	$\frac{1}{24}(\xi^4 - 16\xi^3 + 72\xi^2 - 96\xi + 24)$	1

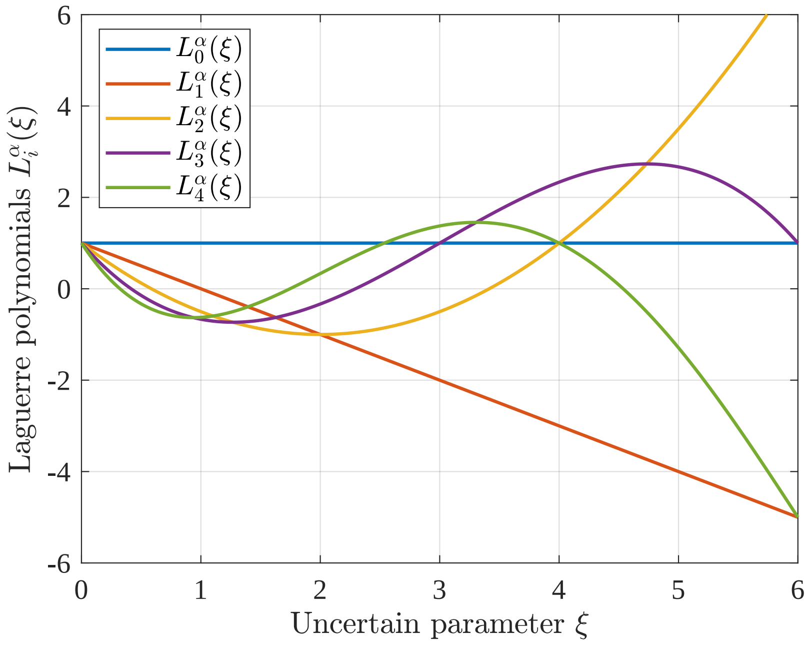


Figure B.2: *First few Laguerre polynomials given $\alpha = 0$.*

B.3 Jacobi Polynomials

Jacobi polynomials $J_i^{\alpha,\beta}(\xi)$ [49] are defined over $[-1, 1]$ and orthogonal with respect to the beta distribution:

$$f(\xi) = \frac{\Gamma(\alpha + \beta + 2)}{2^{\alpha+\beta+1}\Gamma(\alpha + 1)\Gamma(\beta + 1)}(1 - \xi)^\alpha(1 + \xi)^\beta, \quad \xi \in [-1, 1],\ \alpha, \beta > -1. \tag{B.10}$$

They satisfy the recurrence relation:

$$\begin{aligned} J_{-1}^{\alpha,\beta}(\xi) &= 0, \\ J_0^{\alpha,\beta}(\xi) &= 1, \\ \frac{2(i+1)(i+\alpha+\beta+1)}{(a+1)(a+2)}J_{i+1}^{\alpha,\beta}(\xi) &= \left(\xi + \frac{\alpha^2 - \beta^2}{a(a+2)}\right)J_i^{\alpha,\beta}(\xi) - \frac{2(i+\alpha)(i+\beta)}{a(a+1)}J_{i-1}^{\alpha,\beta}(\xi),\ i \in \mathbb{N}_0, \end{aligned} \tag{B.11}$$

where

$$a = 2i + \alpha + \beta. \tag{B.12}$$

Jacobi polynomials satisfy the following orthogonality condition:

$$\begin{aligned}\left\langle J_i^{\alpha,\beta}(\xi), J_j^{\alpha,\beta}(\xi)\right\rangle &= \int_{-1}^{1} J_i^{\alpha,\beta}(\xi) J_j^{\alpha,\beta}(\xi) f(\xi)\mathrm{d}\xi \\ &= \frac{(\alpha+1)_i(\beta+1)_i}{i!(2i+\alpha+\beta+1)(\alpha+\beta+2)_{i-1}}\delta_{ij}.\end{aligned} \tag{B.13}$$

This indicates

$$\gamma_i = \mathbb{E}\left[J_i^{\alpha,\beta}(\xi)^2\right] = \frac{(\alpha+1)_i(\beta+1)_i}{i!(2i+\alpha+\beta+1)(\alpha+\beta+2)_{i-1}}. \tag{B.14}$$

Given $\alpha = 0$ and $\beta = 0$, Equations (B.10), (B.11), and (B.13) reduce to Equations (B.1), (B.2), and (B.3), respectively. This means that Legendre polynomials are a special case of Jacobi polynomials.

Appendix C

Stationary Correlation Functions

Tables C.1 and C.2 list several popular 1-dimensional and multi-dimensional stationary correlation functions, respectively. Here, $\eta \geq 0$ and $\eta_k \geq 0$.

Table C.1: *Commonly used 1-dimensional stationary correlation functions.*

Correlation function	Expression
Exponential	$\exp\left(-\eta\lvert\theta-\theta'\rvert\right)$
Gaussian	$\exp\left(-\eta(\theta-\theta')^2\right)$
Matérn 3/2	$\left(1+\sqrt{3}\eta\lvert\theta-\theta'\rvert\right)\exp\left(-\sqrt{3}\eta\lvert\theta-\theta'\rvert\right)$
Matérn 5/2	$\left(1+\sqrt{5}\eta\lvert\theta-\theta'\rvert+\frac{5}{3}\eta^2(\theta-\theta')^2\right)\exp\left(-\sqrt{5}\eta\lvert\theta-\theta'\rvert\right)$

Table C.2: *Commonly used stationary anisotropic correlation functions.*

Correlation function	Expression
Exponential	$\prod_{k=1}^{n}\exp\left(-\eta_k\lvert\theta_k-\theta'_k\rvert\right)$
Gaussian	$\prod_{k=1}^{n}\exp\left(-\eta_k(\theta_k-\theta'_k)^2\right)$
Matérn 3/2	$\prod_{k=1}^{n}\left(1+\sqrt{3}\eta_k\lvert\theta_k-\theta'_k\rvert\right)\exp\left(-\sqrt{3}\eta_k\lvert\theta_k-\theta'_k\rvert\right)$
Matérn 5/2	$\prod_{k=1}^{n}\left(1+\sqrt{5}\eta_k\lvert\theta_k-\theta'_k\rvert+\dfrac{5}{3}\eta_k^2(\theta_k-\theta'_k)^2\right)\exp\left(-\sqrt{5}\eta_k\lvert\theta_k-\theta'_k\rvert\right)$

Table C.3 presents PLS correlation functions [153] which are based on the anisotropic correlation functions shown in Table C.2. Here, $\eta_j \geq 0$, $\vartheta_{jk} = w^*_{jk}\theta_k$ and $\vartheta'_{jk} = w^*_{jk}\theta'_k$.

Table C.3: *PLS correlation functions.*

Correlation function	Expression
Exponential	$\prod_{j=1}^{m}\prod_{k=1}^{n} \exp\left(-\eta_j\lvert\vartheta_{jk} - \vartheta'_{jk}\rvert\right)$
Gaussian	$\prod_{j=1}^{m}\prod_{k=1}^{n} \exp\left(-\eta_j(\vartheta_{jk} - \vartheta'_{jk})^2\right)$
Matérn 3/2	$\prod_{j=1}^{m}\prod_{k=1}^{n} \left(1+\sqrt{3}\eta_j\lvert\vartheta_{jk} - \vartheta'_{jk}\rvert\right)\exp\left(-\sqrt{3}\eta_j\lvert\vartheta_{jk} - \vartheta'_{jk}\rvert\right)$
Matérn 5/2	$\prod_{j=1}^{m}\prod_{k=1}^{n} \left(1+\sqrt{5}\eta_j\lvert\vartheta_{jk} - \vartheta'_{jk}\rvert + \frac{5}{3}\eta_j^2(\vartheta_{jk} - \vartheta'_{jk})^2\right)\exp\left(-\sqrt{5}\eta_j\lvert\vartheta_{jk} - \vartheta'_{jk}\rvert\right)$

Bibliography

[1] International Civil Aviation Organization, Montreal, Canada, Convention on International Civil Aviation, 9th Edition (2006).
URL `https://www.icao.int/publications/pages/doc7300.aspx`

[2] European Aviation Safety Agency, Easy Access Rules for All Weather Operations (CS-AWO) (2018).
URL `https://www.easa.europa.eu/document-library/easy-access-rules/easy-access-rules-all-weather-operations-cs-awo-initial-issue`

[3] European Aviation Safety Agency, Certification Specifications and Acceptable Means of Compliance for Large Aeroplanes, CS-25, Amendment 25 (2020).
URL `https://www.easa.europa.eu/document-library/certification-specifications/cs-25-amendment-25`

[4] S. Marelli, R. Schöbi, B. Sudret, UQLab User Manual – Structural Reliability (Rare Event Estimation), Tech. rep., Chair of Risk, Safety and Uncertainty Quantification, ETH Zurich, Switzerland (2021).
URL `https://www.uqlab.com/reliability-user-manual`

[5] A. M. Hasofer, N. C. Lind, Exact and invariant second-moment code format, Journal of the Engineering Mechanics Division 100 (1) (1974) 111–121. `doi:10.1061/JMCEA3.0001848`.

[6] M. Lemaire, A. Chateauneuf, J.-C. Mitteau, Structural Reliability, John Wiley & Sons, 2009. `doi:10.1002/9780470611708`.

[7] A. Der Kiureghian, H.-Z. Lin, S.-J. Hwang, Second-order reliability approximations, Journal of Engineering Mechanics 113 (8) (1987) 1208–1225. `doi:10.1061/(ASCE)0733-9399(1987)113:8(1208)`.

[8] M. Tichý, First-order third-moment reliability method, Structural Safety 16 (3) (1994) 189–200. `doi:10.1016/0167-4730(94)00021-H`.

[9] Y.-G. Zhao, T. Ono, M. Kato, Second-order third-moment reliability method, Journal of Structural Engineering 128 (8) (2002) 1087–1090. doi:10.1061/(ASCE)0733-9445(2002)128:8(1087).

[10] Y.-G. Zhao, Z.-H. Lu, Fourth-moment standardization for structural reliability assessment, Journal of Structural Engineering 133 (7) (2007) 916–924. doi:10.1061/(ASCE)0733-9445(2007)133:7(916).

[11] Z.-H. Lu, D.-Z. Hu, Y.-G. Zhao, Second-order fourth-moment method for structural reliability, Journal of Engineering Mechanics 143 (4) (2017) 06016010. doi:10.1061/(ASCE)EM.1943-7889.0001199.

[12] J. Zhang, X. Du, A second-order reliability method with first-order efficiency, Journal of Mechanical Design 132 (10) (2010) 101006. doi:10.1115/1.4002459.

[13] X. Huang, Y. Li, Y. Zhang, X. Zhang, A new direct second-order reliability analysis method, Applied Mathematical Modelling 55 (2018) 68–80. doi:10.1016/j.apm.2017.10.026.

[14] C. Huang, A. El Hami, B. Radi, Overview of structural reliability analysis methods—part I: local reliability methods, Incert. Fiabilité des Systèmes Multiphysiques 17 (2017) 1–10. doi:10.21494/ISTE.OP.2017.0115.

[15] N. Metropolis, S. Ulam, The Monte Carlo method, Journal of the American Statistical Association 44 (247) (1949) 335–341. doi:10.1080/01621459.1949.10483310.

[16] K. M. Beck, James L.and Zuev, Rare-event simulation, in: R. Ghanem, D. Higdon, H. Owhadi (Eds.), Handbook of Uncertainty Quantification, Springer, 2016, pp. 1–26. doi:10.1007/978-3-319-11259-6_24-1.

[17] S. Asmussen, P. W. Glynn, Stochastic Simulation: Algorithms and Analysis, Springer Science & Business Media, 2007. doi:10.1007/978-0-387-69033-9.

[18] R. Melchers, Importance sampling in structural systems, Structural Safety 6 (1) (1989) 3–10. doi:10.1016/0167-4730(89)90003-9.

[19] S.-K. Au, J. Beck, Important sampling in high dimensions, Structural Safety 25 (2) (2003) 139–163. doi:10.1016/S0167-4730(02)00047-4.

[20] L. S. Katafygiotis, K. M. Zuev, Geometric insight into the challenges of solving high-dimensional reliability problems, Probabilistic Engineering Mechanics 23 (2-3) (2008) 208–218. doi:10.1016/j.probengmech.2007.12.026.

[21] A. Owen, Y. Zhou, Safe and effective importance sampling, Journal of the American Statistical Association 95 (449) (2000) 135–143. doi:10.1080/01621459.2000.10473909.

[22] V. Elvira, L. Martino, D. Luengo, M. F. Bugallo, Generalized multiple importance sampling, Statistical Science 34 (1) (2019) 129–155. doi:10.1214/18-STS668.

[23] O. Cappé, R. Douc, A. Guillin, J.-M. Marin, C. P. Robert, Adaptive importance sampling in general mixture classes, Statistics and Computing 18 (4) (2008) 447–459. doi:10.1007/s11222-008-9059-x.

[24] I. Papaioannou, C. Papadimitriou, D. Straub, Sequential importance sampling for structural reliability analysis, Structural Safety 62 (2016) 66–75. doi:10.1016/j.strusafe.2016.06.002.

[25] L. Martino, V. Elvira, D. Luengo, J. Corander, Layered adaptive importance sampling, Statistics and Computing 27 (3) (2017) 599–623. doi:10.1007/s11222-016-9642-5.

[26] S.-K. Au, J. L. Beck, Estimation of small failure probabilities in high dimensions by subset simulation, Probabilistic Engineering Mechanics 16 (4) (2001) 263–277. doi:10.1016/S0266-8920(01)00019-4.

[27] S.-K. Au, Y. Wang, Engineering Risk Assessment with Subset Simulation, John Wiley & Sons, 2014. doi:10.1002/9781118398050.

[28] C. P. Robert, G. Casella, Monte Carlo Statistical Methods, 2nd Edition, Springer, 2004. doi:10.1007/978-1-4757-4145-2.

[29] A. Santoso, K. Phoon, S. Quek, Modified Metropolis–Hastings algorithm with reduced chain correlation for efficient subset simulation, Probabilistic Engineering Mechanics 26 (2) (2011) 331–341. doi:10.1016/j.probengmech.2010.08.007.

[30] K. M. Zuev, L. S. Katafygiotis, Modified Metropolis–Hastings algorithm with delayed rejection, Probabilistic Engineering Mechanics 26 (3) (2011) 405–412. doi:10.1016/j.probengmech.2010.11.008.

[31] L. Tierney, A. Mira, Some adaptive Monte Carlo methods for Bayesian inference, Statistics in Medicine 18 (17-18) (1999) 2507–2515. doi:10.1002/(SICI)1097-0258(19990915/30)18:17/18<2507::AID-SIM272>3.0.CO;2-J.

[32] I. Papaioannou, W. Betz, K. Zwirglmaier, D. Straub, MCMC algorithms for subset simulation, Probabilistic Engineering Mechanics 41 (2015) 89–103. doi:10.1016/j.probengmech.2015.06.006.

[33] S. K. Au, Z.-H. Wang, S.-M. Lo, Compartment fire risk analysis by advanced Monte Carlo simulation, Engineering Structures 29 (9) (2007) 2381–2390. doi:10.1016/j.engstruct.2006.11.024.

[34] S. K. Au, J. Ching, J. Beck, Application of subset simulation methods to reliability benchmark problems, Structural Safety 29 (3) (2007) 183–193. doi:10.1016/j.strusafe.2006.07.008.

[35] M. T. Sichani, S. R. Nielsen, First passage probability estimation of wind turbines by Markov chain Monte Carlo, Structure and Infrastructure Engineering 9 (10) (2013) 1067–1079. doi:10.1080/15732479.2012.659739.

[36] F. Cadini, D. Avram, N. Pedroni, E. Zio, Subset simulation of a reliability model for radioactive waste repository performance assessment, Reliability Engineering & System Safety 100 (2012) 75–83. doi:10.1016/j.ress.2011.12.012.

[37] D. Löbl, F. Holzapfel, Subset simulation for estimating small failure probabilities of an aerial system subject to atmospheric turbulences, in: AIAA Atmospheric Flight Mechanics Conference, 2015, p. 0236. doi:10.2514/6.2015-0236.

[38] C. Mishra, S. Maskell, S.-K. Au, J. F. Ralph, Efficient estimation of probability of conflict between air traffic using subset simulation, IEEE Transactions on Aerospace and Electronic Systems 55 (6) (2019) 2719–2742. doi:10.1109/TAES.2019.2899714.

[39] M. Wang, S. Zhang, F. Holzapfel, D. Löbl, F. Hellmundt, Probabilistic assessment of a safety-critical backup controller by subset simulation, Journal of Guidance, Control, and Dynamics 42 (5) (2019) 1146–1156. doi:10.2514/1.G003901.

[40] C. Mishra, F. Schwaiger, N. M. Bähr, F. Sax, M. A. Kleser, P. Nagarajan, F. Holzapfel, Efficient verification and validation of performance-based safety requirements using subset simulation, in: AIAA Scitech 2021 Forum, 2021, p. 0072. doi:10.2514/6.2021-0072.

[41] A. I. Forrester, A. J. Keane, Recent advances in surrogate-based optimization, Progress in Aerospace Sciences 45 (1-3) (2009) 50–79. doi:10.1016/j.paerosci.2008.11.001.

[42] B. Sudret, Meta-models for structural reliability and uncertainty quantification, in: The 5th Asian-Pacific Symposium on Structure Reliability and its Applications, 2012, pp. 53–76. doi:10.3850/978-981-07-2219-7_P321.

[43] A. I. Khuri, S. Mukhopadhyay, Response surface methodology, Wiley Interdisciplinary Reviews: Computational Statistics 2 (2) (2010) 128–149. doi:10.1002/wics.73.

[44] J. E. Hurtado, An examination of methods for approximating implicit limit state functions from the viewpoint of statistical learning theory, Structural Safety 26 (3) (2004) 271–293. doi:10.1016/j.strusafe.2003.05.002.

[45] P. Breitkopf, H. Naceur, A. Rassineux, P. Villon, Moving least squares response surface approximation: Formulation and metal forming applications, Computers & Structures 83 (17-18) (2005) 1411–1428. doi:10.1016/j.compstruc.2004.07.011.

[46] R. G. Ghanem, P. D. Spanos, Stochastic Finite Elements: A Spectral Approach, Springer, 1991. doi:10.1007/978-1-4612-3094-6.

[47] D. Xiu, G. E. Karniadakis, The Wiener–Askey polynomial chaos for stochastic differential equations, SIAM Journal on Scientific Computing 24 (2) (2002) 619–644. doi:10.1137/S1064827501387826.

[48] S. S. Isukapalli, Uncertainty analysis of transport-transformation models, Ph.D. thesis, Rutgers The State University of New Jersey, New Brunswick, New Jersey (1999).
URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.473.402&rep=rep1&type=pdf

[49] D. Xiu, Numerical Methods for Stochastic Computations: A Spectral Method Approach, Princeton University Press, 2010. doi:10.1515/9781400835348.

[50] B. Sudret, Polynomial chaos expansions and stochastic finite element methods, in: K.-K. Phoon, J. Ching (Eds.), Risk and Reliability in Geotechnical Engineering, CRC Press, 2015, pp. 265–300. doi:10.1201/b17970.

[51] J. Sacks, W. J. Welch, T. J. Mitchell, H. P. Wynn, Design and analysis of computer experiments, Statistical science 4 (4) (1989) 409–423. doi:10.1214/ss/1177012413.

[52] V. Vapnik, The Nature of Statistical Learning Theory, Springer, 2013. doi:10.1007/978-1-4757-3264-1.

[53] K. Gurney, An Introduction to Neural Networks, CRC Press, 1997. doi:10.1201/9781315273570.

[54] L. Faravelli, Response-surface approach for reliability analysis, Journal of Engineering Mechanics 115 (12) (1989) 2763–2781. doi:10.1061/(ASCE)0733-9399(1989)115:12(2763).

[55] C. G. Bucher, U. Bourgund, A fast and efficient response surface approach for structural reliability problems, Structural Safety 7 (1) (1990) 57–66. doi:10.1016/0167-4730(90)90012-E.

[56] M. R. Rajashekhar, B. R. Ellingwood, A new look at the response surface approach for reliability analysis, Structural Safety 12 (3) (1993) 205–220. doi:10.1016/0167-4730(93)90003-J.

[57] S.-H. Kim, S.-W. Na, Response surface method using vector projected sampling points, Structural Safety 19 (1) (1997) 3–19. doi:10.1016/S0167-4730(96)00037-9.

[58] P. K. Das, Y. Zheng, Cumulative formation of response surface and its use in reliability analysis, Probabilistic Engineering Mechanics 15 (4) (2000) 309–315. doi:10.1016/S0266-8920(99)00030-2.

[59] I. Kaymaz, C. A. McMahon, A response surface method based on weighted regression for structural reliability analysis, Probabilistic Engineering Mechanics 20 (1) (2005) 11–17. doi:10.1016/j.probengmech.2004.05.005.

[60] X. S. Nguyen, A. Sellier, F. Duprat, G. Pons, Adaptive response surface method based on a double weighted regression technique, Probabilistic Engineering Mechanics 24 (2) (2009) 135–143. doi:10.1016/j.probengmech.2008.04.001.

[61] N. Roussouly, F. Petitjean, M. Salaun, A new adaptive response surface method for reliability analysis, Probabilistic Engineering Mechanics 32 (2013) 103–115. doi:10.1016/j.probengmech.2012.10.001.

[62] H. Guimarães, J. C. Matos, A. A. Henriques, An innovative adaptive sparse response surface method for structural reliability analysis, Structural Safety 73 (2018) 12–28. doi:10.1016/j.strusafe.2018.02.001.

[63] C. Bucher, T. Most, A comparison of approximate response functions in structural reliability analysis, Probabilistic Engineering Mechanics 23 (2-3) (2008) 154–163. doi:10.1016/j.probengmech.2007.12.022.

[64] C. Proppe, Estimation of failure probabilities by local approximation of the limit state function, Structural Safety 30 (4) (2008) 277–290. doi:10.1016/j.strusafe.2007.04.001.

[65] S.-C. Kang, H.-M. Koh, J. F. Choo, An efficient response surface method using moving least squares approximation for structural reliability analysis, Probabilistic Engineering Mechanics 25 (4) (2010) 365–371. doi:10.1016/j.probengmech.2010.04.002.

[66] J. Li, H. Wang, N. H. Kim, Doubly weighted moving least squares and its application to structural reliability analysis, Structural and Multidisciplinary Optimization 46 (1) (2012) 69–82. doi:10.1007/s00158-011-0748-2.

[67] S. Goswami, S. Ghosh, S. Chakraborty, Reliability analysis of structures by iterative improved response surface method, Structural Safety 60 (2016) 56–66. doi:10.1016/j.strusafe.2016.02.002.

[68] S. Kabasi, A. Roy, S. Chakraborty, A generalized moving least square-based response surface method for efficient reliability analysis of structure, Structural and Multidisciplinary Optimization 63 (3) (2021) 1085–1097. doi:10.1007/s00158-020-02743-9.

[69] B. Sudret, A. Der Kiureghian, Stochastic finite elements and reliability: A state-of-the-art report, Tech. Rep. UCB/SEMM-2000/08, University of California, Berkeley (2000).
URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.453.7433&rep=rep1&type=pdf

[70] B. Sudret, A. Der Kiureghian, Comparison of finite element reliability methods, Probabilistic Engineering Mechanics 17 (4) (2002) 337–348. doi:10.1016/S0266-8920(02)00031-0.

[71] S.-K. Choi, R. V. Grandhi, R. A. Canfield, Structural reliability under non-gaussian stochastic behavior, Computers & Structures 82 (13-14) (2004) 1113–1121. doi:10.1016/j.compstruc.2004.03.015.

[72] M. Berveiller, B. Sudret, M. Lemaire, Stochastic finite element: a non intrusive approach by regression, European Journal of Computational Mechanics/Revue Européenne de Mécanique Numérique 15 (1-3) (2006) 81–92. doi:10.3166/remn.15.81-92.

[73] M. Paffrath, U. Wever, Adapted polynomial chaos expansion for failure detection, Journal of Computational Physics 226 (1) (2007) 263–281. doi:10.1016/j.jcp.2007.04.011.

[74] S. Marelli, B. Sudret, An active-learning algorithm that combines sparse polynomial chaos expansions and bootstrap for structural reliability analysis, Structural Safety 75 (2018) 67–74. doi:10.1016/j.strusafe.2018.06.003.

[75] G. Blatman, B. Sudret, An adaptive algorithm to build up sparse polynomial chaos expansions for stochastic finite element analysis, Probabilistic Engineering Mechanics 25 (2) (2010) 183–197. doi:10.1016/j.probengmech.2009.10.003.

[76] C. Hu, B. D. Youn, Adaptive-sparse polynomial chaos expansion for reliability analysis and design of complex engineering systems, Structural and Multidisciplinary Optimization 43 (3) (2011) 419–442. doi:10.1007/s00158-010-0568-9.

[77] G. Blatman, B. Sudret, Adaptive sparse polynomial chaos expansion based on least angle regression, Journal of Computational Physics 230 (6) (2011) 2345–2367. doi:10.1016/j.jcp.2010.12.021.

[78] V. Romero, L. Swiler, A. Giunta, Construction of response surfaces based on progressive-lattice-sampling experimental designs with application to uncertainty propagation, Structural Safety 26 (2) (2004) 201–219. doi:10.1016/j.strusafe.2003.03.001.

[79] I. Kaymaz, Application of kriging method to structural reliability problems, Structural Safety 27 (2) (2005) 133–151. doi:10.1016/j.strusafe.2004.09.001.

[80] B. J. Bichon, M. S. Eldred, L. P. Swiler, S. Mahadevan, J. M. McFarland, Efficient global reliability analysis for nonlinear implicit performance functions, AIAA journal 46 (10) (2008) 2459–2468. doi:10.2514/1.34321.

[81] B. Echard, N. Gayton, M. Lemaire, AK-MCS: An active learning reliability method combining kriging and Monte Carlo simulation, Structural Safety 33 (2) (2011) 145–154. doi:10.1016/j.strusafe.2011.01.002.

[82] B. Echard, N. Gayton, M. Lemaire, N. Relun, A combined importance sampling and kriging reliability method for small failure probabilities with time-demanding numerical models, Reliability Engineering & System Safety 111 (2013) 232–240. doi:10.1016/j.ress.2012.10.008.

[83] X. Huang, J. Chen, H. Zhu, Assessing small failure probabilities by AK-SS: An active learning method combining kriging and subset simulation, Structural Safety 59 (2016) 86–95. doi:10.1016/j.strusafe.2015.12.003.

[84] C. Ling, Z. Lu, K. Feng, X. Zhang, A coupled subset simulation and active learning kriging reliability analysis method for rare failure events, Structural and Multidisciplinary Optimization 60 (6) (2019) 2325–2341. doi:10.1007/s00158-019-02326-3.

[85] M. Xiao, J. Zhang, L. Gao, S. Lee, A. T. Eshghi, An efficient kriging-based subset simulation method for hybrid reliability analysis under random and interval variables with small failure probability, Structural and Multidisciplinary Optimization 59 (6) (2019) 2077–2092. doi:10.1007/s00158-018-2176-z.

[86] F. Cui, M. Ghosn, Implementation of machine learning techniques into the subset simulation method, Structural Safety 79 (2019) 12–25. doi:10.1016/j.strusafe.2019.02.002.

[87] R. Schöbi, B. Sudret, S. Marelli, Rare event estimation using polynomial-chaos kriging, ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A: Civil Engineering 3 (2) (2017) D4016002. doi:10.1061/AJRUA6.0000870.

[88] C. M. Rocco, J. A. Moreno, Fast Monte Carlo reliability evaluation using support vector machine, Reliability Engineering & System Safety 76 (3) (2002) 237–243. doi:10.1016/S0951-8320(02)00015-7.

[89] J. E. Hurtado, D. A. Alvarez, Classification approach for reliability analysis with stochastic finite-element modeling, Journal of Structural Engineering 129 (8) (2003) 1141–1149. doi:10.1061/(ASCE)0733-9445(2003)129:8(1141).

[90] H.-S. Li, Z.-Z. Lü, Z.-F. Yue, Support vector machine for structural reliability analysis, Applied Mathematics and Mechanics 27 (10) (2006) 1295–1303. doi:10.1007/s10483-006-1001-z.

[91] J. E. Hurtado, Filtered importance sampling with support vector margin: a powerful method for structural reliability analysis, Structural Safety 29 (1) (2007) 2–15. doi:10.1016/j.strusafe.2005.12.002.

[92] A. Basudhar, S. Missoum, A. H. Sanchez, Limit state function identification using support vector machines for discontinuous responses and disjoint failure domains, Probabilistic Engineering Mechanics 23 (1) (2008) 1–11. doi:10.1016/j.probengmech.2007.08.004.

[93] J.-M. Bourinet, F. Deheeger, M. Lemaire, Assessing small failure probabilities by combined subset simulation and support vector machines, Structural Safety 33 (6) (2011) 343–353. doi:10.1016/j.strusafe.2011.06.001.

[94] H. Dai, H. Zhang, W. Wang, G. Xue, Structural reliability assessment by local approximation of limit state functions using adaptive Markov chain simulation and support vector regression, Computer-Aided Civil and Infrastructure Engineering 27 (9) (2012) 676–686. doi:10.1111/j.1467-8667.2012.00767.x.

[95] J.-M. Bourinet, Rare-event probability estimation with adaptive support vector regression surrogates, Reliability Engineering & System Safety 150 (2016) 210–221. doi:10.1016/j.ress.2016.01.023.

[96] J.-M. Bourinet, Reliability assessment by adaptive kernel-based surrogate models-approximation of non-smooth limit-state functions, in: 13th International Conference on Applications of Statistics and Probability in Civil Engineering (ICASP13), 2019. doi:10.22725/ICASP13.177.

[97] M. Papadrakakis, V. Papadopoulos, N. D. Lagaros, Structural reliability analyis of elastic-plastic structures using neural networks and Monte Carlo simulation, Computer Methods in Applied Mechanics and Engineering 136 (1-2) (1996) 145–163. doi:10.1016/0045-7825(96)01011-0.

[98] J. E. Hurtado, D. A. Alvarez, Neural-network-based reliability analysis: a comparative study, Computer Methods in Applied Mechanics and Engineering 191 (1-2) (2001) 113–132. doi:10.1016/S0045-7825(01)00248-1.

[99] J. Deng, D. Gu, X. Li, Z. Q. Yue, Structural reliability analysis for implicit performance functions using artificial neural network, Structural Safety 27 (1) (2005) 25–48. doi:10.1016/j.strusafe.2004.03.004.

[100] J. Cheng, Hybrid genetic algorithms for structural reliability analysis, Computers & Structures 85 (19-20) (2007) 1524–1533. doi:10.1016/j.compstruc.2007.01.018.

[101] J. Cheng, Q. Li, Reliability analysis of structures using artificial neural network based genetic algorithms, Computer Methods in Applied Mechanics and Engineering 197 (45-48) (2008) 3742–3750. doi:10.1016/j.cma.2008.02.026.

[102] V. Papadopoulos, D. G. Giovanis, N. D. Lagaros, M. Papadrakakis, Accelerated subset simulation with neural networks for reliability analysis, Computer Methods in Applied Mechanics and Engineering 223 (2012) 70–80. doi:10.1016/j.cma.2012.02.013.

[103] N.-C. Xiao, M.-J. Zuo, C.-N. Zhou, A new adaptive sequential sampling method to construct surrogate models for efficient reliability analysis, Reliability Engineering & System Safety 169 (2018) 330–338. doi:10.1016/j.ress.2017.09.008.

[104] N. Lelièvre, P. Beaurepaire, C. Mattrand, N. Gayton, A. Otsmane, On the consideration of uncertainty in design: optimization-reliability-robustness, Structural and Multidisciplinary Optimization 54 (6) (2016) 1423–1437. doi:10.1007/s00158-016-1556-5.

[105] T. Chatterjee, S. Chakraborty, R. Chowdhury, A critical review of surrogate assisted robust design optimization, Archives of Computational Methods in Engineering 26 (1) (2019) 245–274. doi:10.1007/s11831-017-9240-5.

[106] Y. Aoues, A. Chateauneuf, Benchmark study of numerical methods for reliability-based design optimization, Structural and Multidisciplinary Optimization 41 (2) (2010) 277–294. doi:10.1007/s00158-009-0412-2.

[107] M. Moustapha, B. Sudret, Surrogate-assisted reliability-based design optimization: A survey and a unified modular framework, Structural and Multidisciplinary Optimization 60 (2019) 2157–2176. doi:10.1007/s00158-019-02290-y.

[108] K. Zhou, J. C. Doyle, Essentials of Robust Control, Upper Saddle River, NJ: Prentice-Hall, 1998.
URL https://www.ece.lsu.edu/kemin/essentials.htm

[109] G. Stein, J. C. Doyle, Beyond singular values and loop shapes, Journal of Guidance, Control, and Dynamics 14 (1) (1991) 5–16. doi:10.2514/3.20598.

[110] S. Bennani, G. Looye, Design of flight control laws for a civil aircraft using μ-synthesis, in: Guidance, Navigation, and Control Conference and Exhibit, 1998, p. 4133. doi:10.2514/6.1998-4133.

[111] F. Amato, U. Ciniglio, F. Corraro, R. Iervolino, μ synthesis for a small commercial aircraft: Design and simulator validation, Journal of Guidance, Control, and Dynamics 27 (3) (2004) 479–490. doi:10.2514/1.10334.

[112] Z. K. Nagy, R. D. Braatz, Open-loop and closed-loop robust optimal control of batch processes using distributional and worst-case analysis, Journal of Process Control 14 (4) (2004) 411–422. doi:10.1016/j.jprocont.2003.07.004.

[113] K. K. K. Kim, R. D. Braatz, Probabilistic analysis and control of uncertain dynamic systems: Generalized polynomial chaos expansion approaches, in: 2012 American Control Conference, IEEE, 2012, pp. 44–49. doi:10.1109/ACC.2012.6314853.

[114] G. C. Calafiore, M. C. Campi, The scenario approach to robust control design, IEEE Transactions on Automatic Control 51 (5) (2006) 742–753. doi:10.1109/TAC.2006.875041.

[115] A. Mesbah, S. Streif, A probabilistic approach to robust optimal experiment design with chance constraints, IFAC-PapersOnLine 48 (8) (2015) 100–105. doi:10.1016/j.ifacol.2015.08.164.

[116] Z. Zhao, M. Kumar, Split-bernstein approach to chance-constrained optimal control, Journal of Guidance, Control, and Dynamics 40 (11) (2017) 2782–2795. doi:10.2514/1.G002551.

[117] J.-B. Caillau, M. Cerf, A. Sassi, E. Trélat, H. Zidani, Solving chance constrained optimal control problems in aerospace via kernel density estimation, Optimal Control Applications and Methods 39 (5) (2018) 1833–1858. doi:10.1002/oca.2445.

[118] J. A. Paulson, A. Mesbah, An efficient method for stochastic optimal control with joint chance constraints for nonlinear systems, International Journal of Robust and Nonlinear Control 29 (15) (2019) 5017–5037. doi:10.1002/rnc.3999.

[119] M. Diehl, J. Bjornberg, Robust dynamic programming for min-max model predictive control of constrained uncertain systems, IEEE Transactions on Automatic Control 49 (12) (2004) 2253–2257. doi:10.1109/TAC.2004.838489.

[120] W. Langson, I. Chryssochoos, S. Raković, D. Q. Mayne, Robust model predictive control using tubes, Automatica 40 (1) (2004) 125–133. doi:10.1016/j.automatica.2003.08.009.

[121] A. Mesbah, S. Streif, R. Findeisen, R. D. Braatz, Stochastic nonlinear model predictive control with probabilistic constraints, in: 2014 American Control Conference, IEEE, 2014, pp. 2413–2419. doi:10.1109/ACC.2014.6858851.

[122] M. Farina, R. Scattolini, Model predictive control of linear systems with multiplicative unbounded uncertainty and chance constraints, Automatica 70 (2016) 258–265. doi:10.1016/j.automatica.2016.04.008.

[123] J. A. Paulson, E. A. Buehler, R. D. Braatz, A. Mesbah, Stochastic model predictive control with joint chance constraints, International Journal of Control 93 (1) (2020) 126–139. doi:10.1080/00207179.2017.1323351.

[124] G. C. Calafiore, L. Fagiano, Stochastic model predictive control of LPV systems via scenario optimization, Automatica 49 (6) (2013) 1861–1866. doi:10.1016/j.automatica.2013.02.060.

[125] D. Shi, F. Holzapfel, Coupled subset simulation and moving least-squares method for reliability-based control optimization, Journal of Guidance, Control, and Dynamics 44 (8) (2021) 1550–1558. doi:10.2514/1.G005786.

[126] D. Shi, X. Fang, F. Holzapfel, Polynomial chaos-based flight control optimization with guaranteed probabilistic performance, in: 21st IFAC World Congress, 2020, pp. 7274–7279. doi:10.1016/j.ifacol.2020.12.565.

[127] D. Shi, F. Holzapfel, Rare-event chance-constrained flight control optimization using surrogate-based subset simulation, in: AIAA Scitech 2021 Forum, 2021, p. 1856. doi:10.2514/6.2021-1856.

[128] L. Brevault, M. Balesdent, J. Morio, et al., Aerospace System Analysis and Optimization in Uncertainty, Springer, 2020. doi:10.1007/978-3-030-39126-3.

[129] M. Hohenbichler, R. Rackwitz, Non-normal dependent vectors in structural safety, Journal of the Engineering Mechanics Division 107 (6) (1981) 1227–1238. doi:10.1061/JMCEA3.0002777.

[130] A. Der Kiureghian, P.-L. Liu, Structural reliability under incomplete probability information, Journal of Engineering Mechanics 112 (1) (1986) 85–104. doi:10.1061/(ASCE)0733-9399(1986)112:1(85).

[131] S. M. Ross, A First Course in Probability, 8th Edition, Pearson Prentice Hall, 2009.

[132] K. M. Zuev, J. L. Beck, S.-K. Au, L. S. Katafygiotis, Bayesian post-processor and other enhancements of subset simulation for estimating failure probabilities in high dimensions, Computers & Structures 92 (2012) 283–296. doi:10.1016/j.compstruc.2011.10.017.

[133] F. Cérou, P. Del Moral, T. Furon, A. Guyader, Sequential Monte Carlo for rare event estimation, Statistics and Computing 22 (3) (2012) 795–808. doi:10.1007/s11222-011-9231-6.

[134] F. Schwaiger, D. Shi, C. Mishra, L. Höhndorf, F. Holzapfel, A modular subset simulation toolbox for Matlab, in: AIAA Scitech 2022 Forum, 2022, p. 1893. doi:10.2514/6.2022-1893.

[135] SAE International, Aerospace - Flight Control Systems - Design, Installation and Test of Piloted Military Aircraft, General Specification For, AS94900 (2007).
URL https://www.sae.org/standards/content/as94900/

[136] A. M. E. Saleh, M. Arashi, B. G. Kibria, Theory of Ridge Regression Estimation with Applications, John Wiley & Sons, 2019. doi:10.1002/9781118644478.

[137] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Springer, 2009. doi:10.1007/978-0-387-84858-7.

[138] Y.-G. Zhao, T. Ono, Moment methods for structural reliability, Structural Safety 23 (1) (2001) 47–75. doi:10.1016/S0167-4730(00)00027-8.

[139] P. Lancaster, K. Salkauskas, Surfaces generated by moving least squares methods, Mathematics of Computation 37 (155) (1981) 141–158. doi:10.1090/S0025-5718-1981-0616367-1.

[140] W. Y. Tey, N. A. C. Sidik, Y. Asako, M. W. Muhieldeen, O. Afshar, Moving least squares method and its improvement: a concise review, Journal of Applied and Computational Mechanics 7 (2) (2021) 883–889. doi:10.22055/JACM.2021.35435.2652.

[141] K. P. Murphy, Machine Learning: A Probabilistic Perspective, MIT Press, 2012.
URL https://mitpress.mit.edu/books/machine-learning-1

[142] T. Hofmann, B. Schölkopf, A. J. Smola, Kernel methods in machine learning, The Annals of Statistics 36 (3) (2008) 1171–1220. doi:10.1214/009053607000000677.

[143] S. Xiao, Z. Lu, Structural reliability sensitivity analysis based on classification of model output, Aerospace Science and Technology 71 (2017) 52–61. doi:10.1016/j.ast.2017.09.009.

[144] T. J. Santner, B. J. Williams, W. I. Notz, The Design and Analysis of Computer Experiments, Springer, 2003. doi:10.1007/978-1-4757-3799-8.

[145] C. E. Rasmussen, C. K. I. Williams, Gaussian Processes for Machine Learning, MIT Press, 2005. doi:10.7551/mitpress/3206.001.0001.

[146] C. Lataniotis, D. Wicaksono, S. Marelli, B. Sudret, UQLab User Manual – Kriging (Gaussian Process Modeling), Tech. rep., Chair of Risk, Safety and Uncertainty Quantification, ETH Zurich, Switzerland (2022).
URL https://www.uqlab.com/kriging-user-manual

[147] M. Moustapha, J.-M. Bourinet, B. Guillaume, B. Sudret, Comparative study of kriging and support vector regression for structural engineering applications, ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A: Civil Engineering 4 (2) (2018) 04018005. doi:10.1061/AJRUA6.0000950.

[148] R. V. Jategaonkar, Flight Vehicle System Identification: A Time Domain Methodology, American Institute of Aeronautics and Astronautics, 2006. doi:10.2514/4.866852.

[149] V. Dubourg, Adaptive surrogate models for reliability analysis and reliability-based design optimization, Ph.D. thesis, Blaise Pascal University, Clermont-Ferrand, France (2011).
URL https://sudret.ibk.ethz.ch/research/publications/doctoralTheses/v--dubourg.html

[150] M. T. Emmerich, K. C. Giannakoglou, B. Naujoks, Single-and multiobjective evolutionary optimization assisted by Gaussian random field metamodels, IEEE Transactions on Evolutionary Computation 10 (4) (2006) 421–439. doi:10.1109/TEVC.2005.859463.

[151] P. Geladi, B. R. Kowalski, Partial least-squares regression: a tutorial, Analytica chimica acta 185 (1986) 1–17. doi:10.1016/0003-2670(86)80028-9.

[152] R. Manne, Analysis of two partial-least-squares algorithms for multivariate calibration, Chemometrics and Intelligent Laboratory Systems 2 (1-3) (1987) 187–197. doi:10.1016/0169-7439(87)80096-5.

[153] M. A. Bouhlel, N. Bartoli, A. Otsmane, J. Morlier, Improving kriging surrogates of high-dimensional design models by Partial Least Squares dimension reduction, Structural and Multidisciplinary Optimization 53 (5) (2016) 935–952. doi:10.1007/s00158-015-1395-9.

[154] I. Couckuyt, T. Dhaene, P. Demeester, ooDACE toolbox: a flexible object-oriented Kriging implementation, Journal of Machine Learning Research 15 (2014) 3183–3186.
URL http://jmlr.org/papers/v15/couckuyt14a.html

[155] D. Moorhouse, R. Woodcock (Eds.), US Military Specification MIL-F-8785C, US Department of Defense, 1980.

[156] MathWorks, Surrogateopt: Surrogate optimization for global minimization of time-consuming objective functions, `https://www.mathworks.com/help/gads/surrogateopt.html`, accessed September 8, 2020 (2018).

[157] S. A. Raab, J. Zhang, P. Bhardwaj, F. Holzapfel, Proposal of a unified control strategy for vertical take-off and landing transition aircraft configurations, in: 2018 Applied Aerodynamics Conference, 2018, p. 3478. `doi:10.2514/6.2018-3478`.

[158] P. Bhardwaj, S. A. Raab, J. Zhang, F. Holzapfel, Integrated reference model for a tilt-rotor vertical take-off and landing transition UAV, in: 2018 Applied Aerodynamics Conference, 2018, p. 3479. `doi:10.2514/6.2018-3479`.

[159] F. Schwaiger, C. Mishra, F. Holzapfel, Acceleration framework to quantify the influence of uncertain parameters on simulation models using Matlab and Simulink, in: AIAA Scitech 2021 Forum, 2021, p. 1980. `doi:10.2514/6.2021-1980`.