An evaluation of approximate probabilistic reachability techniques for stochastic parametric hybrid systems

Mariia Vasileva

School of Computing

University of Newcastle

A thesis submitted for the degree of

Doctor of Philosophy

April 2020

Acknowledgements

I would like especially to thank my supervisor Paolo Zuliani, who has always been inspirational and enthusiastic in guiding my research. I would also like to acknowledge the Gaussian process community as a whole and especially Sergey Lisitsyn and Heiko Strathmann of the Shogun toolnet; Filipe Rodrigues from the Technical University of Denmark and Dimitrios Milios from the University of Edinburgh; everyone has been extremely receptive, supportive and helpful to me over the course of my PhD.

I would like to thank my office mate Fedor Shmarov for many hours of productive discussions, development, research and for his friendship. I have also enjoyed the interaction with all members of the ICOS group during my time here. I would like to thank Aisha Blfgeh, Denis Taniguchi, and Rouaa Yassin-Kassab for helpful discussions at various times during my PhD.

For financial support, I thank the SAgE Doctoral Training Scholarships of Newcastle University.

I thank my family for their moral support, and above all, I would like to thank my friend Irina Maksimenko for her endless help and belief in me.

Abstract

Stochastic parametric hybrid systems allow formalising automata with discrete interruptions, continuous nonlinear dynamics and parametric uncertainty (e.g. randomness and/or nondeterminism), and are a useful framework for cyber-physical systems modelling. The problem of designing safe cyber-physical systems is very timely, given that such systems are ubiquitous in modern society, often in safety-critical contexts (*e.g.*, aircraft and cars) with possibly some level of decisional autonomy. Therefore, the verification of cyber-physical systems (and consequently of hybrid systems) is a problem urgently demanding innovative solutions. Unfortunately, this problem is also extremely challenging.

Reachability checking is a crucial element of designing safe systems. Given a system model, we specify a set of "goal" states (indicating (un)wanted behaviour) and ask whether the system evolution can reach these states or not. Probabilistic reachability is the corresponding problem for stochastic systems, and it amounts to computing the probability that the system reaches a goal state.

The main problem researched in this thesis is probabilistic reachability analysis of hybrid systems with random and/or nondeterministic parameters. For nondeterministic systems, this problem amounts to computing a range of reachability probabilities depending on how nondeterminism is resolved.

In this thesis I have investigated and developed three distinct techniques:

- Statistical methods, involving Monte Carlo, Quasi-Monte Carlo

and Randomised Quasi-Monte Carlo sampling with interval estimation techniques which give statistical guarantees;

- An analytical approximation method, utilising Gaussian Processes that offer a statistical approximation for an (unknown) smooth function over its entire domain;
- A promising combination of a formal approach, based on formal reasoning which provides absolute numerical guarantees, and the Gaussian Regression method.

This research offers contributions on two different levels to the verification of stochastic parametric hybrid systems. From a theoretical point of view, it offers a proof that the reachability probability function is a smooth function of the uncertain parameters of the model, and hence Gaussian Processes techniques can be used to obtain an efficient analytical approximation of the function. From a practical point of view, I have implemented all the above described statistical and approximation techniques as part of the publicly available ProbReach tool, including a Gaussian Process Expectation Propagation algorithm that performs Gaussian Process classification and regression for uni-variate and multiple class labels. My empirical evaluation of the presented techniques to a number of case studies has shown a great Gaussian Process approach advantage with respect to standard statistical model checking techniques.

Contents

Chapter 1

Introduction

1.1 Introduction

One of the fundamental problems in verification and model checking is reachability. Given a system model and a set of "goal" states (indicating (un)wanted behaviour), does the system eventually reach these states? Probabilistic reachability is defined as the generalisation of this problem for stochastic parametric hybrid systems (SPHS), and it entails computing the probability that the system reaches a goal state. Hybrid (discrete-continuous) systems [52], which are the main focus of this thesis, are considered to be a very successful framework for modelling cyber-physical systems. The vast use of cyber-physical (and hence of hybrid) systems in our society, often in safety-critical contexts (*e.g.*, aircraft and cars) with possibly some level of decisional autonomy, is the reason why the verification of these systems is a problem needing urgent scalable solutions. At the same time, unfortunately, this problem is extremely challenging.

Checking reachability in hybrid systems is an undecidable problem for all but the simplest systems (e.g. timed automata - a timed automaton is a finite state automaton extended with a set of real-valued variables modelling clocks) [7], [6]. It is a well-known fact that verifying satisfiability of formulas involving real variables, which can arise in formal verification of hybrid systems, is an undecidable problem when, *e.g.*, trigonometric functions are involved. The notion of δ-complete decision procedure [30] was defined by Gao, Avigad and Clarke in

order to combat the undecidability of general sentences over the reals. This approach has been extended to a bounded probabilistic reachability method with statistically valid enclosures for the probability that a hybrid system can reach a goal state within a given time bound and the number of steps [74]. State-of-the-art statistical techniques [24, 74] cannot fully compute the bounded reachability probability function for the systems under consideration, as they struggle with nondeterministic systems.

All the above-mentioned facts serve as a motivation for us to find efficient and numerically accurate statistical techniques that can deal with realistic, nonlinear hybrid systems. In particular, we address hybrid systems with random parameters whose distribution is subject to nondeterministic parametric uncertainty, and we aim at (approximately) solving the *bounded* probabilistic reachability problem (with bounded meaning that we consider only a finite number of discrete steps and finite time in the system evolution). For nondeterministic systems, this problem amounts to computing a range of reachability probabilities depending on how nondeterminism is resolved.

In my work, I developed methods that compute under- and over-approximation of the reachability probability, which involves computing multi-dimensional integrals. I investigated four main approaches to compute such integrals: formal, Monte-Carlo (MC), Quasi-Monte Carlo (QMC) and Gaussian Process (GP). It is known that the number of system evolutions to explore in order to accurately compute integrals grows exponentially with respect to the number of dimensions [85]. This motivates the exploration of a combination of MC and QMC methods and numerical decision procedures in order to define efficient, numerically accurate estimation techniques.

It is well-known that the Law of Large Numbers and random sampling serve as a basis for MC methods. Instead, QMC methods are based on *deterministic* sampling from so-called quasi-random sequences [78]. Theoretically, it is possible to compute the estimation error of the QMC method by the Koksma-Hlawka inequality. The aim of the inequality is to bound the QMC estimation error by the discrepancy of the sample points and the variation of the integrand product. These two quantities measure sample point consistency and integrand roughness, respectively. Unfortunately, the practical usage of the the Koksma-Hlawka

inequality involves a number of calculation difficulties [37]. The Central Limit Theorem (CLT), which states that a properly defined sum of independent random variables tends towards a normal distribution even if the original variables are not normally distributed themselves, cannot be used for estimating the integration error, as the terms of quasi-random sequences are statistically dependent. However, we can successfully apply the CLT for estimating the error of *Randomised* Quasi-Monte Carlo (RQMC) methods.

In my thesis I present a comparison of different interval estimation techniques, particularly in the extreme cases of probability close to 0 or 1, where the actual coverage probability of many Confidence Intervals (CI) techniques can be poor [16, 59].

I also motivate the use of GP and apply it for approximating the bounded reachability probability function over the nondeterministic parameters domain. Given an (unknown) smooth real function and a set of function evaluations at a *finite* set of (training) input points, a GP offers a statistical approximation for said function over its *entire domain* (with asymptotic guarantees). This contribution allows us to look at approximating the reachability probability function from another side, by using machine learning techniques. In perticular, I first show that the reachability probability function for our class of stochastic hybrid systems is a smooth function of the nondeterministic parameters. This fact thereby justifies our use of GP to approximate the reachability probability function. Next, I provide a comparison of GP approximation with statistical model checking (SMC) methods and show that GP offers comparable accuracy to SMC while requiring much less simulation effort.

The experiments show that our modified CLT technique and GPs are usable in practice even for complex dynamics and for probabilities close to the bounds. The QMC-based techniques we considered have excellent convergence and efficiency especially when the number of samples is small. The results presented in this thesis also prove the GP advantage in terms of CPU time, number of samples and CI average size with respect to standard statistical model checking. For example, the Gaussian process approach needs between 3-135 times fewer samples, it spends between 2-56 times less CPU time for calculating results and it shows between 2-67 times smaller confidence interval average size, depending on the confidence

level. Based on my analysis of the CIs, I suggest that my results can be used as guidelines for probability estimation techniques.

In my thesis I provide new results of the comparing a novel statistical technique for computing bounded reachability probability in Stochastic nondeterministic Parametric Hybrid Systems (SnPHSs) (combined approach) and the GP Expectation Propagation (GPEP) method. The combined approach represents a new perspective combination of machine learning technique and formal methods for approximation. It gives statistically rigorous confidence intervals by combining the formal approach with an appropriate precision of probability enclosures and the GP regression method.

This combination provides very promising results in terms of calculation precision and computational complexity. On the basis of the small research into the SnPHS model it is feasible to predict an effectiveness of the combined approach application and choose an appropriate precision. The latter fact gives us hope that GP regression in combination with the formal approach as well as GPEP approach can be an effective solution not only for rare event cases but also in general.

1.2 Related Work

The reachability probability can be defined by adding a quantitative measure to bounded reachability by introducing random parameters to a hybrid system. A range of reachability probabilities are introduced after adding nondeterministic parameters to the system above (*i.e.*, the reachability probability becomes a function of nondeterministic parameters).

It is possible to verify such systems in two ways: formally, by rigorously computing the probability measure of random parameters for the parameter sets satisfying the bounded reachability property, or statistically, by sampling the parameter space according to the parameters' distributions and evaluating bounded reachability for each drawn sample. The former approach grants stronger (absolute) guarantees but has high computational complexity as disadvantage [93], while the latter approach provides weaker (statistical) guarantees relaxing the complexity [94].

Considering the formal verification approach, in [83] Stochastic Satisfiability Modulo Theory (SSMT) is suggested by expanding the classical nonlinear Satisfiability Modulo Theory with randomised quantifiers. Nonetheless, this work is restricted to finite domains (only discrete randomness is supported). In [33] this problem is dealt with by extending SSMT to continuous domains (CSSMT) to support continuous randomness, but the approach does not include Ordinary Differentiation Equations (ODEs), and the accuracy of the produced results is not guaranteed.

In [2] an approach to computing bounds on reachability probabilities for stochastic parametric hybrid systems is introduced, which involves abstraction by discrete-time Markov chains. This method is further developed to full Linear Temporal Logic (LTL) and nondeterminism [84]. Reference [60] describes model checking algorithms for Probabilistic Computation Tree Logic (PCTL) formulae over continuous-time stochastic parametric hybrid systems. However, in [2, 60, 84] the authors address the problem of the continuous state space through finite discretisation, providing approximate numerical solutions for the experiments. Instead, in this thesis I present algorithms which consider continuous time and space, and give full statistical/numerical guarantees.

Considering the statistical verification technique, in [24] we find a statistical model checking approach for verifying hybrid systems with continuous randomness and nondeterminism. Nonetheless, in the presented method SMT decision procedures are combined with fixed-sample size techniques, based on Hoeffding's inequality [24]. Besides, this work does not deal with stochastic hybrid systems, whose dynamics are defined by ODEs. In comparison with the algorithm developed by [74], the presented technique involves a more suitable non-sequential Bayesian approach, thus efficiently accelerating the verification procedure.

In this thesis I also consider QMC methods, which are regarded as a deterministic counterpart to classical MC methods [79]. In [37] the authors show the advantages of QMC methods with respect to MC but also highlight the main problem of QMC error bounds estimation and consider a practical application of the Koksma-Hlawka inequality. One of the newest approaches for QMC variance estimation is introduced in [1]. In [16] a number of interval estimation techniques for a binomial proportion are revisited and their coverage probability is examined.

I also focus on works that combine verification with GP-based methods. In [11] it is shown that the satisfaction probability of temporal logic formulas over uncertain continuous-time Markov chains is a smooth function of the nondeterministic parameters, and hence GP approximation can be used. Some authors suggested an online model learning using stochastic hybrid systems based on GP, but without formal justification [4]. In [12], reachable sets of (non-hybrid) dynamic systems are computed with the help of GPs, but the authors directly assume smooth system dynamics.

In my work I use the GP algorithm whose basis was initially proposed in [56] and then generalised in [53]. In [66], GP training using multiple annotators per input point is considered, although with binary values only. In the Shogun library [80], a range of tools are provided, including expectation-propagation algorithms for binary classification. References [24, 74] propose statistical approaches for nondeterministic stochastic hybrid systems, but it is only possible to compute the extrema of the reachability probability function, while my algorithms analytically approximate it over its entire domain.

1.3 Aims and Objectives

The aim of this work is to evaluate approximate probabilistic reachability techniques for the verification of stochastic parametric hybrid systems and provide efficient alternatives to known approximation methods. The following objectives were identified:

- investigate techniques for probabilistic reachability analysis and develop algorithms for computing bounded reachability probability in Parametric Hybrid Systems with statistical guarantees, which make a significant impact on the change of the state of art;
- provide a comprehensive evaluation of the confidence intervals calculation methods and produce guidelines for probability estimation techniques;
- show the theoretical basis importance of the presented novel approaches by proving that the reachability probability function is a smooth function of

the uncertain parameters of a model, hence showing that GP techniques can be used to obtain an efficient analytical approximation of the function;

- explore GP statistical methods for improving the performance of the formal approach while providing new techniques via a composition of statistically and numerically accurate results;

- implement attained theoretical findings in a software tool, and apply it to several complex case studies.

1.4 Thesis Outline

This section presents the outline and shows the contributions of this thesis with respect to the declared aim and objectives.

- **Chapter 2** introduces bounded reachability in Parametric Hybrid Systems (PHS) and defines Stochastic nondeterministic Parametric Hybrid System (SnPHS). This Chapter presents delta-complete decision procedures and integral estimation methods including MC, QMC and RQMC. Additionally, this Chapter describes CI error estimation approaches based on the standard CLT interval and on the *Beta* function and provides a brief description of GP methods for regression and classification.

- **Chapter 3** discusses estimation techniques for probabilistic reachability analysis and shows modifications of the classical CLT method. The Chapter also provides a proof of smoothness of the reachability probability function for SnPHSs, which is a precaution for GP approximation. This Chapter also investigates and develops a new combined approach: a promising combination of the formal and the GP approaches.

- **Chapter 4** describes the architecture and implementation details of the developed tools in ProbReach, which incorporates the algorithms introduced in Chapters 2 and 3.

- **Chapter 5** demonstrates several real-world case studies for the developed tool application, such as UVB therapy for treating psoriasis and pharmacokinetics model for anaesthesia delivery. This Chapter also presents a full description of the results, which have been obtained by using the algorithms from Chapter 2, 3 and tools from Chapter 4.
- **Chapter 6** contains final conclusions and some pointers for future work.

1.5 Publications

Portions of the work within this thesis have been documented in the following publications:

Conferences/Workshops:

M. Vasileva and P. Zuliani, "An evaluation of estimation techniques for probabilistic reachability. Full version", in *Symbolic and Numerical Methods for Reachability Analysis, 4th International Workshop, SNR 2018*. Avaliable at: https://arxiv.org/abs/1804.03121.

M. Vasileva, F. Shmarov and P. Zuliani, "Approximate Probabilistic Reachability for Uncertain Stochastic Hybrid Systems", in *8th International Conference on Computational Methods in Systems Biology, CMSB 2020*, to be submitted.

Chapter 2

Background

2.1 Introduction

It is impossible to overestimate the positive impact made by mathematical modelling and model verification on the process of system design. With their help, we can dismiss faulty implementations, while doing it experimentally would be very expensive, time-consuming, or even hardly possible. Also *in silico* (computational) analysis can grant further indications and prognosis for physical experiments.

Stochastic hybrid systems are widely used in modelling various real-world systems from different domains. For example, they are applied for modelling biological systems such as DNA reproduction and networks of gene regulation [48], closed-loop systems giving feedback such as the transmission of insulin for type 1 diabetes patients [41] (also referred to as artificial pancreas), and cyber-physical systems such as powertrains [43], wind turbines [89] and autonomous underwater vehicles [17].

The verification of stochastic hybrid systems helps to deal with important problems, such as analysis of probabilistic reachability and safety [3], arranging control and planning strategies [26], and parameter identification [49].

This Chapter presents an introduction to Stochastic nondeterministic Parametric Hybrid Systems (SnPHS) as a subclass of SPHS [74] - dynamical systems that combine discrete and continuous dynamic behaviour with continuous and dis-

crete parameters whose values are set in the initial state and remain unchanged during the system's evolution. The Chapter explains reachability checking problems in hybrid systems, in particular their undecidability, and suggest δ-complete decision procedures as a solution, which combat the undecidability of reasoning over real sentences [30] and correctly decide whether a slightly relaxed sentence is satisfiable or not. On this basis the needs of computing either rigorous enclosures [73] or verified confidence intervals [74] are further discussed.

Also, the Chapter discusses four main ways for integral estimation - formal approach, which is based on formal reasoning and provides absolute numerical guarantees and the MC, QMC and RQMC computation methods, which give statistical guarantees only. In this Chapter, I paid attention to the main idea behind QMC - the fact that the true randomness of the sampling process is less relevant rather than the even spread of the samples over the integration domain, which can significantly increase the accuracy of the estimation. The Chapter considered QMC problems in terms of the difficulty of computing an estimate of the integration error and suggested an efficient way to solve this problem by using RQMC. The latter allows Confidence Intervals (CIs) construction for error estimation. As a result, I introduced several CI construction techniques: CIs based on the Beta-Function, including Bayesian Interval, Jeffreys Interval, Clopper-Pearson Interval and CIs based on the central limit theorem, including Wilson Interval, Agresti-Coull Interval, Logit Interval, Anscombe Interval and Arcsine Interval. The Qint algorithm (see Subsection 2.3.5) proposed by Antonov for QMC variance estimation and based on a set of random quadrature formulas is also considered in this Chapter.

Finally, this Chapter introduces machine learning techniques application to SnPHSs models. In particular, I present GP approximation methods, which severely depend on the initial mean and covariance functions. The Chapter exhaustively discusses selecting an appropriate covariance function and covariance hyperparameters issues for hybrid system models. The Chapter also covers Gaussian Process Classification (GPC) algorithms for probabilistic classification, where predictions take the form of class probabilities, and explain the main differences between GPC and GP learning on the example of different multiple annotators techniques. At the end of this Chapter, I provide the description of

the Expectation Propagation (EP) technique [53, 56], which solves the problem of non-Gaussian inference in our SnPHSs.

2.2 Hybrid Systems

Hybrid systems [7] can be considered as a generalisation of finite-state machines for representing continuous behaviour.

Hybrid systems consist of two components: *flows* that generally can be modelled using nonlinear ODEs and behave according to the laws of physics, and *jumps*, which model discrete state changes between different flows. A set of Boolean predicates are used to detect the possibility of a discrete transition between the continuous flows.

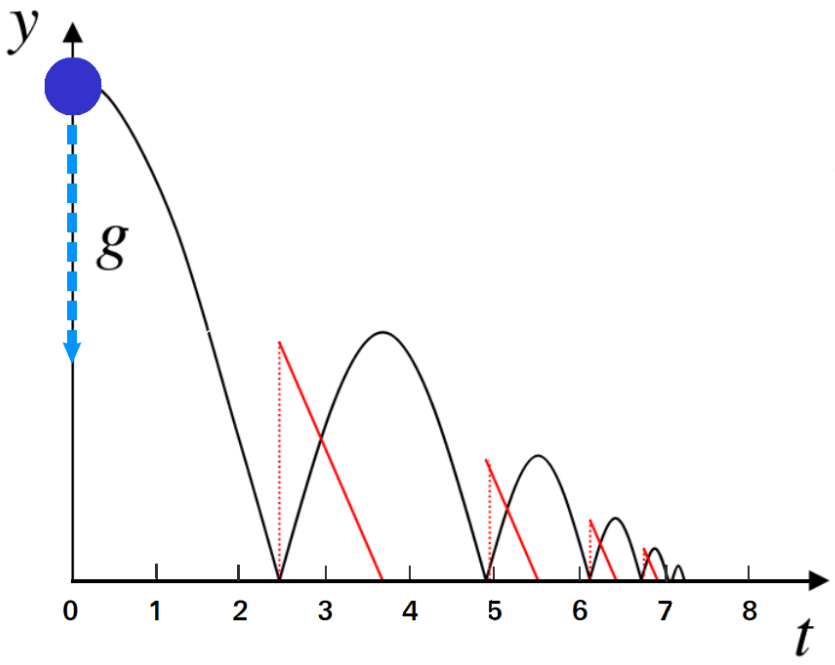


Figure 2.1: A trajectory of a bouncing ball thrown down from the point $(0, y)$ with gravitational constant g. The jump and reset states are shown by the red dashed line and the flow state evolution until the next fall is shown by the red solid line.

Also, a *reset* of the initial values of continuous variables in the successor flow can be caused by a discrete transition. Besides, very common features of hybrid

systems are invariants – predicates which should be true for all time points in the flow.

A bouncing ball model presents a simple example of a hybrid system. Consider a ball thrown down with some initial speed, depending on the gravitational constant g (see Figure 2.1). The ball's dynamics evolve continuously over time while it is in the air (*flow*), and when it touches the ground the discrete state change occurs (*jump*). At the same time, the speed of the ball is reduced when it touches the ground (*reset*).

Stochastic parametric hybrid systems (SPHS) can be used to model such systems containing uncertain parameters. There are a wide range of SPHSs types, which show the contrasting level of abstraction: starting from systems with stochastic dynamics (defined by stochastic differential equations) to systems which have random and nondeterministic parameters [48, 74, 90].

2.2.1 Definitions

Parametric Hybrid Systems (PHSs) (see Definition 2.1) represent continuous and discrete dynamic behaviour dependent on initial parameters, which remain unchanged during the system evolution. Such systems can both flow, described by a differential equation and jump, described by differential equations or control graphs.

Definition 2.1. (Parametric Hybrid System [74])

A Parametric Hybrid System (PHS) is a tuple

$$H =< Q, \Upsilon, X, P, Y, R, \text{jump}, \text{goal} >$$

where

- $Q = \{q_0, \cdots, q_m\}$ *is a set of modes (discrete components of the system),*
- $\Upsilon = \{(q, q') : q, q' \in Q\}$ *is a set of transitions between modes,*
- $X = [u_1, v_1] \times \cdots \times [u_n, v_n] \subset \mathbb{R}^n$ *is is a domain of continuous variables,*
- $P = [a_1, b_1] \times \cdots \times [a_k, b_k] \subset \mathbb{R}^k$ *the parameter space of the system,*

- $Y = \{\mathbf{y}_q(\mathbf{p}, t) : q \in Q, \mathbf{p} \in X \times P, t \in [0, T]\}$ *the continuous system dynamics where* $\mathbf{y}_q : X \times P \times [0, T] \to X$*,*
- $R = \{\mathbf{g}_{(q,q')}(\mathbf{p}, t) : (q, q') \in \Upsilon, \mathbf{p} \in X \times P, t \in [0, T]\}$ *are 'reset' functions* $\mathbf{g}_{(q,q')} : X \times P \times [0, T] \to X$ *defining the continuous state at time* $t = 0$ *in mode* q' *after taking the transition from mode* q*.*

and predicates (or relations)

- $\text{jump}_{(q,q')}(\mathbf{x})$ *defines a discrete transition* $(q, q') \in \Upsilon$ *which may (but does not have to) occur upon reaching the jump condition in state* $(\mathbf{x}, q) \in X \times P \times Q$*,*
- $\text{goal}_q(\mathbf{x})$ *defines the goal state* $\mathbf{x}$ *in mode* q*.*

Stochastic Parametric Hybrid Systems (SPHS) [74] are dynamical systems that combine discrete and continuous dynamic behaviour with continuous and discrete parameters whose values are set in the initial state and remain unchanged during the system's evolution. The parameters can be probabilistic, in which case a probability measure is associated to them, or nondeterministic otherwise.

In this thesis, I focus on hybrid systems with nondeterministic parameters. I define Stochastic *nondeterministic* Parametric Hybrid Systems (SnPHS) as a subclass of SPHS [74]:

Definition 2.2. *A* Stochastic nondeterministic Parametric Hybrid System *(SnPHS) is a tuple* $\langle Q, \Upsilon, X, P, Y, I, \Xi, \mathbf{jump}, \mathbf{goal} \rangle$*:*

- $Q = \{q_0, \ldots, q_m\}$ *is a set of modes (discrete components of the system),*
- $\Upsilon = \{(q, q') : q, q' \in Q\}$ *is a set of transitions between modes,*
- $X = [u_1, v_1] \times \cdots \times [u_n, v_n] \subset \mathbb{R}^n$ *is a domain of continuous variables,*
- $P \subset \mathbb{R}^k$ *is the (compact)* nondeterministic *parameter space, with associated probability density functions* $f_1(\cdot, \mathbf{p}), ..., f_r(\cdot, \mathbf{p})$ *for* r random *parameters with* $\mathbf{p} \in P$ *and with domain* $R = [a_1, b_1] \times \cdots \times [a_r, b_r] \subset \mathbb{R}^r$*,*
- $Y = \{\mathbf{flow}_q : q \in Q\}$ *where* $\mathbf{flow}_q : X \times R \times [0, T] \to X$ *is the continuous system dynamics,*

- $I = \{\mathbf{init}_q : q \in Q\}$ *where* $\mathbf{init}_q : R \to X$ *computes the initial continuous state in mode* q,
- $\Xi = \{\mathbf{reset}_{(q,q')} : (q, q') \in \Upsilon\}$ *where* $\mathbf{reset}_{(q,q')} : X \times R \times [0, T] \to X$ *defines the continuous state at time* $t = 0$ *in mode* q' *after taking the transition from mode* q.

and predicates (or relations)

- $\mathbf{jump}_{(q,q')}(\mathbf{x}) \equiv$ *discrete transition* $(q, q') \in \Upsilon$ *occurs upon reaching the jump condition in state* $(\mathbf{x}, q) \in X \times R \times [0, T] \times Q$,
- $\mathbf{goal}_q(\mathbf{x}) \equiv$ *state* $\mathbf{x} \in X \times R \times [0, T]$ *in mode* q *is a goal state.*

SnPHS restrict SPHS [74] by disallowing nondeterministic jumps, *i.e.*, we require that every $\mathbf{reset}_{q,q'} \in \Xi$ is a Boolean predicate and their *true* preimages are disjoint. We assume that the domain of the random parameters does not depend on the nondeterministic parameters $\mathbf{p}$, but this is not a restriction in practice. Indeed, the domain of certain distributions (*e.g.*, the uniform distribution) can depend on nondeterministic parameters. However, it is possible to apply a change of variable to make the domain independent of the nondeterministic parameters (*e.g.*, given $a < b$ and a random variable U uniformly distributed over an interval $[0, 1]$, then $Z = (b - a)U + a$ is uniformly distributed over $[a, b]$). Also, the boundedness of R is not a significant restriction in practice, since any probability density can be approximated to arbitrary precision by a density over a bounded support. The continuous dynamics Y is made of Lipschitz-continuous ODEs, which have a unique solution for any initial condition in $X \times R \times [0, T]$ according to the well-known Picard-Lindelöf theorem. Again, both nondeterministic and random parameters do not change through the system's evolution.

2.2.2 Undecidability of Reachability

Checking reachability in hybrid systems, even for linear ones, is known to be generally undecidable [7]. Moreover, bounded reachability in hybrid systems with nonlinear continuous dynamics is undecidable for the reason of undecidability of nonlinear arithmetic over the reals. Despite the fact that the first-order logic of

real polynomials has been proven by Tarski to be decidable [82], the introduction of trigonometric functions (*e.g.* $\sin, \cos$) makes the problem undecidable [47, 64, 88].

For robust hybrid systems (systems where the reachability property holds under small input perturbations) the reachability problem becomes decidable [27, 30].

One-sided guaranteed answers can be returned by certain decision procedures. For instance, δ-satisfiability solves the problem of undecidability over the reals by applying a procedure which returns one-sided guaranteed answers: one of the two answers is accurate while the other one is liable to some over-approximation [30, 55]. There is an algorithm which always terminates correctly returning one of the answers above, thus making such a problem (called δ-decision) decidable.

Similarly, in [28] a procedure is suggested that incorporates interval constraint propagation (ICP), which consists in iterating domain reductions by using the set of constraints until no domain can be contracted and SAT solving techniques to grant one-sided decisions for Boolean combinations on nonlinear arithmetic constraints.

In addition, a semi-terminating algorithm for safety verification of nonlinear hybrid systems is introduced in [62]. This algorithm terminates for robust instances of the problem (*i.e.*, safety holds comfortably for some positively perturbed version of the system), and it may run eternally otherwise.

2.2.3 Delta-Complete Decision Procedure

In order to combat the undecidability of reasoning over real sentences Gao *et al.* [30] defined δ-complete decision procedures, which correctly decide whether a slightly relaxed sentence is satisfiable or not.

The basic definitions [74] are given next:

Definition 2.3. (δ-Weakening [30]) *Given an arbitrary $\delta > 0$ and a bounded $\mathcal{L}_{\mathbb{R}}$-sentence (see Definition 2.4)*

$$\phi := Q_1^{X_1} x_1, \ldots, Q_n^{X_n} x_n : \bigwedge_{i=1}^{m} \Big(\bigvee_{j=1}^{k_i} \big(f_{i,j}(x_1, \ldots, x_n) \circ 0 \big) \Big),$$

where the $f_{i,j}(x_1, ..., x_n)$ are compositions of Type 2-computable functions [44] (these are essentially "numerically computable" real functions, including transcendental functions and solutions of differential equations), $Q_i = \{\exists, \forall\}$, and $\circ \in \{>, \geq\}$.

Definition 2.4. (Bounded $\mathcal{L}_{\mathbb{R}}$-formula [30]) *A bounded $\mathcal{L}_{\mathbb{R}}$-formula is defined as follows [74]:*

$$\begin{aligned} t &:= c \mid x \mid f\big(t(x)\big), \\ \phi &:= t(x) > 0 \mid t(x) \geq 0 \mid \phi \wedge \phi \mid \phi \vee \phi \mid \exists^{[u,v]} x : \phi \mid \forall^{[u,v]} x : \phi, \end{aligned} \tag{2.1}$$

where c is a constant, x is a variable, f is a computable real function, and $\exists^{[u,v]}x, \forall^{[u,v]}x$ are bounded quantifiers – shorthand for $\exists x \in [u, v]$ and $\forall x \in [u, v]$.

Given a sentence ϕ as above and $\delta \in \mathbb{Q}^+$ the δ-decision problem asks to correctly decide one of the following: unsatisfiable (ϕ is false), or δ-satisfiable (ϕ^δ is true), where ϕ^δ is the δ-weakening *of ϕ: The δ-weakening of ϕ is the formula:*

$$\phi^\delta := Q_1^{X_1} x_1, \ldots, Q_n^{X_n} x_n : \bigwedge_{i=1}^{m} \Big(\bigvee_{j=1}^{k_i} \big(f_{i,j}(x_1, \ldots, x_n) \circ \delta\big) \Big).$$

In this thesis, Type 2 computability terms (see Definition 2.5) is used to define computable real function (see Definition 2.6). In general, we define a real function as computable if its value can be algorithmically approximated with arbitrary finite precision. One of the most important properties of computable functions is that they are continuous [44]. Computable real functions include transcendental functions and solutions of differential equations.

Definition 2.5. (Computable Real Number [13, Definition 3.1]) *A real number x is computable if there exists a computable sequence of rational numbers $(q_n)_{n\in\mathbb{N}}$ that converges to x (*i.e., *$\forall i : |x - q_i| < 2^{-i}$).*

Definition 2.6. (Computable Real Function [13, Definition 4.1]) *A function $f : \mathbb{R}^n \to \mathbb{R}$ is computable if there is an oracle Turing machine M that, given any precision $k \in \mathbb{N}$ and input $x \in dom(f)$, quires another procedure for an arbitrarily good rational approximation q_i of x satisfying $|x - q_i| < 2^{-i}$, and produces a rational number $M(q_i)$ such that $|f(x) - M(q_i)| < 2^{-k}$.*

The bounded δ-SMT problem asks for the following: given a sentence ϕ of the above form and $\delta \in \mathbb{Q}^+$, correctly decide one of the following:

- **unsat**: ϕ is false,
- δ**-true**: ϕ^δ is true.

If the two cases overlap either decision can be returned. Standard bounded reachability questions over PHSs can be coded as sentences of the type introduced by Definition 2.3 and "δ-decided" by δ-complete decision procedures [22, 32].

Definition 2.7. (δ-complete decision procedure [30]) *Given $\delta > 0$, a δ-complete decision procedure correctly decides whether an arbitrary bounded $\mathcal{L}_\mathbb{R}$-sentence is false or its δ-weakening is true, returning* **unsat** *and* δ**-sat** *respectively. When both cases overlap, either answer can be returned.*

It can be concluded from the definitions above that **unsat** means that the given $\mathcal{L}_\mathbb{R}$-sentence is false. However, it is also can be noticed that although δ**-sat** implies satisfiability of the δ-weakening of the shown sentence, its original version can still be false. This is usually known as a *false* alarm, which means that for the unsatisfiable $\mathcal{L}_\mathbb{R}$-sentence a δ-complete decision procedure returns δ**-sat**. It happens because of the crude over-approximation introduced by δ.

There are SAT ODE solvers that implement a δ-decision procedure - *e.g.* `iSAT-ODE` [23], and `dReal` [31].

2.2.4 Bounded Probabilistic Reachability

In Figure 2.2, a simple thermostat hybrid system model is presented. On the basis of its behaviour, which depends on the guard conditions, a bounded reachability question can be formulated: does the temperature in the room, where such thermostat is installed reach the "bad region" in 5 steps? Steps here denote changes between two modes (see Figure 2.3). The notion of "bad region" here represents unwanted temperature values which can be reached during the thermostat work after 5 steps. In particular, Figure 2.3 assigns these values around the [18, 21.5] region. This term is used for defining a goal state, whose probability of occurrence we need to verify. For example, during a biological experiment with genetically

modified plants, we need to make sure that the plants are resistant to a sharp temperature change, so the thermostat should be set so that the temperature in the room should not be in the bad region because this region represents an uncomfortable climate. Similarly one can also define the goal set as a "good" region.

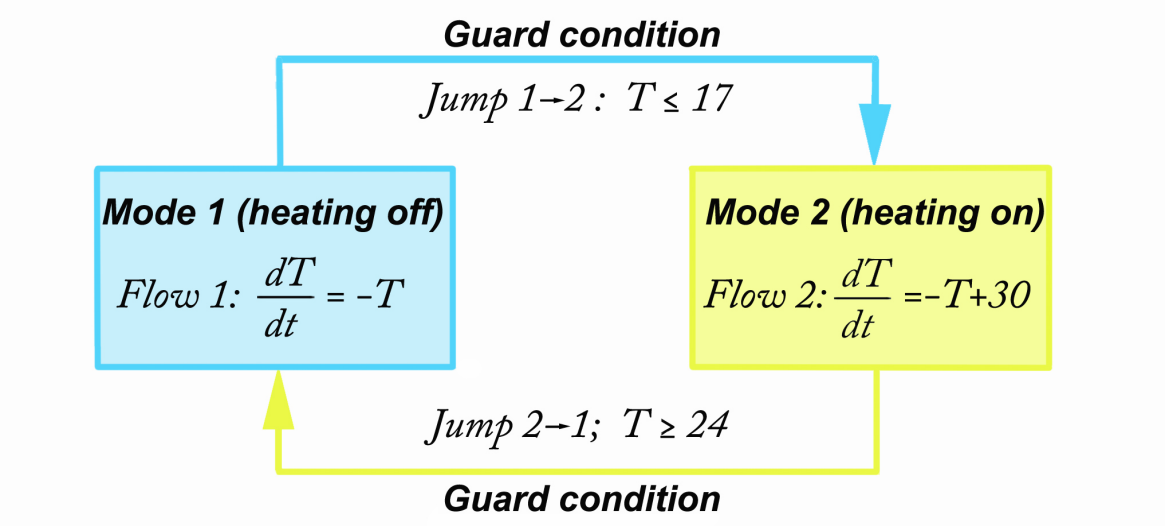


Figure 2.2: A thermostat hybrid system model. The system states are represented by **Mode 1** and **Mode 2**. Continuous behaviour is shown by *Flow 1* and *Flow 2*. The guard condition refers to system *Jump* between two modes.

As stated above, checking such reachability question in hybrid systems is generally undecidable, and thus we need to compute either rigorous enclosures [73] or verified confidence intervals [74]. Both approaches exploit δ-complete decision procedures [30] to reason about nonlinear hybrid systems.

We now move towards probabilistic bounded reachability. The following definition is a specialisation of the corresponding notion for SnPHS [75].

Definition 2.8. *Given a SnPHS and reachability depth* $l \in \mathbb{N}$*, the* bounded reachability probability *function* $\mathbf{Pr}: P \to [0,1]$ *is:*

$$\mathbf{Pr}(\mathbf{p}) = \int_G d\mu(\mathbf{p})$$

where $\mu(\mathbf{p})$ *is the probability measure given by the product of the probability den-*

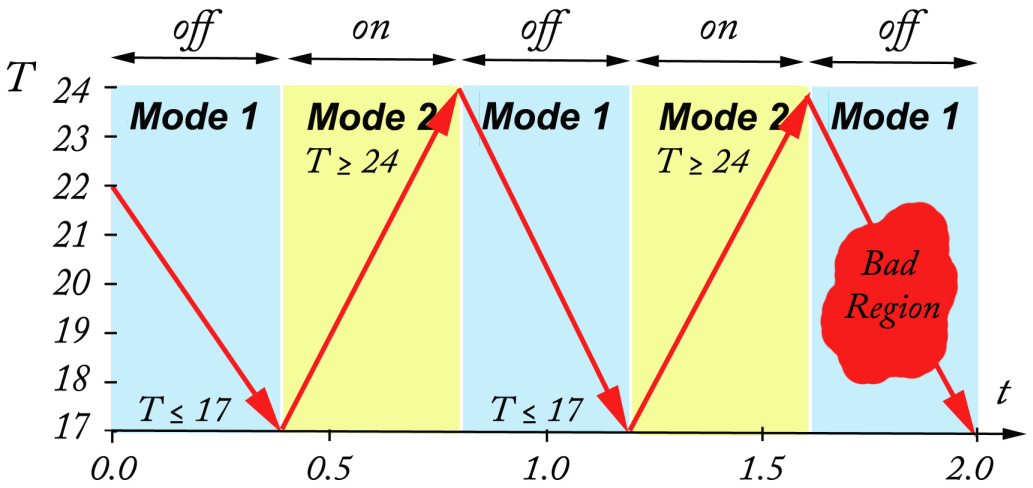


Figure 2.3: Representation of the thermostat hybrid system model bounded reachability question over time graph. The system evaluate from **Mode 1** to **Mode 2** and back through the system *Jump* according to the guard conditions. Bounded reachability question: Does the temperature reach the *Bad region* in 5 steps.

sities $f_1(\cdot, \mathbf{p}), \ldots, f_r(\cdot, \mathbf{p})$ *and* G *is the* goal set*:*

$$G = \{x \in R : \exists \pi \in \Pi(l) : \mathbf{reach}(\pi, x)\},$$

where $\Pi(l)$ *is the set of paths of length* l *in the SnPHS and* **reach** *is the sentence in standard form (see Definition 2.3) that defines* l*-step reachability:*

$$\mathbf{reach}(\pi, x) \equiv \exists^{[0,T]} t_0, \cdots, \exists^{[0,T]} t_{l-1} : \big(\mathbf{x}_0(t_0) = \mathbf{flow}_{\pi[0]}(\mathbf{init}_{\pi[0]}(x), x, t_0)\big) \wedge$$
$$\bigwedge_{i=0}^{l-2} \Big[\mathbf{jump}_{(\pi[i],\pi[i+1])}(\mathbf{x}_i(t_i), x, t_i) \wedge \big(\mathbf{x}_{i+1}(t_{i+1}) =$$
$$\mathbf{flow}_{\pi[i+1]}(\mathbf{reset}_{\pi[i],\pi[i+1]}(\mathbf{x}_i(t_i), x, t_i), x, t_i)\big)\Big] \wedge \mathbf{goal}_{\pi[l-1]}\big(\mathbf{x}_{l-1}(t_{l-1}), x, t_{l-1}\big).$$

We recall that in SnPHS the predicates **flow**, **init**, **jump**, **reset** and **goal** must involve Type 2 computable functions only.

It is possible to check the reachability of the goal mode in l steps via finding a path π (see Definition 2.9) such that:

Definition 2.9. (Path trajectory [72]) *Given a PHS* H*, a path* π *of depth* l *is a finite sequence of modes of* H *such that* $\boldsymbol{init}_{\pi[0]} \in \boldsymbol{init}$*,* $\boldsymbol{jump}_{(\pi[i],\pi[i+1])} \in \boldsymbol{jump}$

for $0 < i < l$, *and* $\boldsymbol{goal}_{\pi[l]} \in \boldsymbol{goal}$. *A trajectory defines a continuous evolution of the system along the given path for the given initial value of the continuous dynamics.*

- the initial element ($\pi[0]$) of π belongs to the set of initial modes,
- the last element ($\pi[l]$) of π is in the set of goal modes,
- for each pair of successive modes $(\pi[i], \pi[i+1])$ there exists a discrete transition defined by $\mathbf{jump}_{(\pi[i],\pi[i+1])}$ and $\mathbf{reset}_{(\pi[i],\pi[i+1])}$.

Unfortunately, the decision on the reachability of the goal state by finding such path π cannot be made, because there might not be a trajectory satisfying the corresponding invariants, goal predicates and jump conditions [77]. It requires checking the values of the continuous dynamics over $P \times [0, T]$.

The bounded reachability formulation that can be verified using δ-complete decision procedures is presented below.

Definition 2.10. (Bounded Reachability [72]) *The bounded reachability property for a PHS* H, *a reachability depth* l, *and a subset* B *of the parameter space of* H *is defined as the bounded* $\mathcal{L}_{\mathbb{R}}$*-sentence:*

$$\mathbf{Reach}(H, l, B) := \exists^{B}\mathbf{p}, \exists^{[0,T]}t_0, \forall^{[0,t_0]}t'_0, \cdots, \exists^{[0,T]}t_{|\pi|-1}, \forall^{[0,t_{|\pi|-1}]}t'_{|\pi|-1} :$$

$$\bigvee_{\pi \in \mathbf{Paths}(H,l)} \Bigg[\big(\mathbf{x}_0(t_0) := \boldsymbol{flow}_{\pi[0]}(\boldsymbol{init}_{\pi[0]}(\mathbf{p}), \mathbf{p}, t_0)\big) \wedge \boldsymbol{invt}_{\pi[0]}\big(\mathbf{x}_0(t'_0), \mathbf{p}, t'_0\big) \wedge$$

$$\bigwedge_{i=0}^{|\pi|-2} \Big[\big(\mathbf{x}_{i+1}(t_{i+1}) := \boldsymbol{flow}_{\pi[i+1]}(\boldsymbol{reset}_{(\pi[i],\pi[i+1])}(\mathbf{x}_i(t_i), \mathbf{p}), \mathbf{p}, t_{i+1})\big) \wedge$$

$$\boldsymbol{jump}_{(\pi[i],\pi[i+1])}(\mathbf{x}_i(t_i), \mathbf{p}, t_i) \wedge \boldsymbol{invt}_{\pi[i+1]}\big(\mathbf{x}_{i+1}(t'_{i+1}), \mathbf{p}, t'_{i+1}\big) \Big] \wedge$$

$$\boldsymbol{goal}_{\pi[|\pi|-1]}\big(\mathbf{x}_{|\pi|-1}(t_{|\pi|-1}), \mathbf{p}, t_{|\pi|-1}\big) \Bigg].$$

The definition of the formula **Reach** with the examples of application can be found in [72].

It is necessary now to introduce a universal quantifier in formula **Reach** to check reachability for all values in given parameter subsets as shown below.

Definition 2.11. (Universal Bounded Reachability) *The universal bounded reachability property for a PHS H, a reachability depth l, and a subset B of the parameter space of H is defined as the bounded $\mathcal{L}_{\mathbb{R}}$-sentence* $\mathbf{Reach}^{\forall}(H, l, B) := \forall^{B}\mathbf{p} : \mathbf{Reach}(H, l, \{\mathbf{p}\})$.

2.2.5 Verifying Bounded Reachability in SnPHS

The formula $\mathbf{Reach}^{\forall}$ defined previously (see Definition 2.11) can be verified by a δ-complete decision procedure as it is defined by bounded $\mathcal{L}_{\mathbb{R}}$-sentences. As stated in Subsection 2.2.4, given a bounded $\mathcal{L}_{\mathbb{R}}$-sentence and a positive δ, a δ-complete decision procedure can correctly decide whether the relaxed version of the given sentence (δ-weakening) is true (outputing *δ-sat*) or it is false (returning *unsat*).

It is important to highlight here that *δ-sat* might in fact be a false alarm because of the over-approximation properties characterised by $\delta > 0$, and therefore, does not guarantee satisfiability of the given bounded $\mathcal{L}_{\mathbb{R}}$-sentence, while *unsat* is a stronger answer implying unsatisfiability of the given formula.

More concretely, a δ-complete decision procedure applied to $\mathbf{Reach}^{\forall}$ returns:

- *unsat* for $\mathbf{Reach}(H, l, B)$ if for all parameter values in $B \subseteq P$ the system H does not reach a goal state;
- *δ-sat* if there exists $\mathbf{p} \in B$ such that $\big(\mathbf{Reach}(H, l, B)\big)^{\delta}$ (the weakening of $\mathbf{Reach}(H, l, B)$) is true.

The decision algorithm combines the properties of δ-complete decision procedures and formulae $\mathbf{Reach}^{\forall}$. Given a PHS H, a reachability depth l, a parameter space P, a subset $B \subseteq P$ and a positive δ, the decision procedure procedure **evaluate** [74] returns:

- **sat** if for all parameter values in B the goal state is reachable in l steps;
- **unsat** if no values from B satisfy the bounded reachability;
- **undet** if neither of the above can be decided, or a *false* alarm occurred due to a too large value of δ being used by the δ-complete decision procedure.

Choosing a smaller δ can sometimes help to reduce the number of *false* alarms.

2.3 Integral Estimation Methods

From the previous section, we can conclude that we can apply a bounded reachabilty probability for approximation SPHSs. In fact, we can determine bounded reachability probability in terms of the algorithm for computing confidence intervals, which is based on the integration of the probability measure and can be presented in the form of sample approximation.

In this thesis four main ways of integral estimation are considered - Formal, Monte-Carlo (MC), Quasi-Monte Carlo (QMC) and Randomised Quasi-Monte Carlo (RQMC). The formal approach is based on formal reasoning and hence provides absolute numerical guarantees. It guarantees an interval computation, which contains the exact reachability probability value. The other methods are based on Monte Carlo sampling and give statistical guarantees only. They construct an interval, which contains the probability value with some statistical confidence.

In case of a SnPHS, that cannot easily be predicted due to the presence of random variables, simulation methods (MC, QMC and RQMC) are used to model the probability of different outcomes in such process. This method is widely used to evaluate the impact of risk and uncertainty in prediction and forecasting models. The simulation methods are also referred to as multiple probabilistic simulations [68].

QMC simulations can offer a much better solution when there is significant uncertainty in the process of making a forecast or estimation in comparison to just replacing the uncertain variable with a single average number. As was noted before, hybrid systems are afflicted by random variables and QMC simulations have a huge array of potential applications in these fields. They are used to estimate the probability of certain events for different models and the likelihood that a certain event can take place. Airlines use them to assess network performance in different scenarios and optimise such networks on the basis of obtained data [67, 81]. Analysts use them widely to analyse derivatives such as options and to

assess the risk that an entity will default [40]. QMC simulations have inumerous applications outside of business and finance, including meteorology, astronomy and particle physics [76].

2.3.1 Formal Approach

The technique presented in [73] and named Formal approach computes the probability enclosures for a certain range of parameters of the bounded reachability probability function **Pr**. The algorithm takes inputs:

- a SnPHS $(H, \mathbb{P})$,
- a reachability depth $l \in \mathbb{N}$,
- a precision $\epsilon > 0$ for the size of the probability enclosures,
- a constant $\kappa \in (0, \epsilon)$ for bounding the domain of continuous random parameters with the unbounded support,
- a precision vector ρ for nondeterministic parameter boxes,
- a parameter η controlling the precision of procedure **evaluate**,

and returns a collection of probability enclosures.

Such a collection is presented in the form of disjoint nondeterministic parameter boxes that fully cover parameters' domain (*i.e.*, each nondeterministic parameter box will be strictly associated with a probability enclosure). In other words, the algorithm initially divides the domain of nondeterministic parameters into boxes of desired precision and then obtains a probability enclosure for each such box. This division is made by computing the under-approximation and the over-approximation of the given integral over the whole random parameter space for the chosen nondeterministic box.

If a SPHS does not feature nondeterministic parameters or in case of SnPHS with fixed nondeterministic parameter, only one probability enclosure will be returned. The size of the probability enclosure will be bounded above by the ϵ input if the given system generates robust bounded $\mathcal{L}_{\mathbb{R}}$-sentences (see Theorem 3.1 in [72]). In general when a model contains nondeterministic parameters, the

size of the enclosure(s) cannot be controlled by ϵ. This is because the reachability probability function **Pr** might be discontinuous. Finally, the size of the smallest nondeterministic parameter box that will be analysed is limited by the precision vector ρ, which guarantees the termination of the algorithm in the most general case.

2.3.2 Monte Carlo Simulations

MC methods enable computational algorithms to repeatedly generate random numbers to solve a given problem. The most precise definition was given by Halton [39]. In general, MC methods can be defined by providing the solution of a problem through a parameter of a hypothetical population. In other words we can construct a sample of the population, from which statistical estimates of the parameter can be obtained, by using a random sequence of numbers. Next, we consider the application of MC to integral estimation and discuss the approximation error.

Consider the integral $I = \int_a^b f(y)dy$, and a random variable U on $[a, b]$. The expectation of $f(U)$ is:

$$\mathbb{E}[f(U)] = \int_a^b f(y)\varphi(y)dy,$$

where φ is the density of U. If U is uniformly distributed on $[a, b]$, then the integral becomes:

$$I = \int_a^b f(y)dy = (b-a)\mathbb{E}[f(U)].$$

If we take N points $(u_1, ..., u_N)$, uniformly distributed on $[a, b]$, and compute the sample mean $\frac{1}{N}\sum_{i=1}^{N} f(u_i)$, we obtain the MC integral estimation:

$$\int_a^b f(y)dy \approx (b-a)\frac{1}{N}\sum_{i=1}^{N} f(u_i). \tag{2.2}$$

According to the Strong Law of Large Numbers, this approximation is convergent (for $N \to \infty$) to I with probability one. The variance of the MC estimator (2.2) is:

$$Var(MC) = \int_a^b ... \int_a^b \left(\frac{1}{N}\sum_{i=1}^{N} f(u_i) - I\right)^2 du_1...du_N = \frac{\sigma_f^2}{N}. \tag{2.3}$$

The MC integration error mean is $\frac{\sigma_f}{\sqrt{N}}$, where σ_f^2 is the integrand variance, which is assumed to exist. In practice, the integrand variance is often unknown. That is why the next estimation is instead used:

$$\widehat{\sigma}_f^2 = \frac{1}{N-1}\sum_{i=1}^{N}\left(f(u_i) - \frac{1}{N}\sum_{i=1}^{N} f(u_i)\right)^2.$$

This estimator satisfies the unbiasedness property (an unbiased estimator can be defined when the mean of the statistic's sampling distribution is equal to the population's parameter): $\mathbb{E}[\widehat{\sigma}_f^2] = \sigma_f^2$.

2.3.3 Quasi-Monte Carlo Simulations

One of the known ways of increasing the efficiency of Monte Carlo simulation is the usage of deterministic sequences. This method is discussed in detail in the work of Niederreiter [54]. The main idea behind QMC is the fact that the true randomness of the sampling process is not that relevant rather than even spreads of the samples over the integration domain. The latter can significantly increase the accuracy of the estimation. This leads us to the problem of the deterministic nodes choice in such a way that the approximation error bound is as small as possible instead. The QMC theory and tools are different from the ordinary MC method - QMC is based on abstract algebra and number theory while MC is based on probability and statistics theory.

In other words, QMC methods can be regarded as a deterministic counterpart to classical MC methods. Unlike MC integration, which uses estimates (see Equation 2.2) with randomly selected points, QMC methods select the points u_i *deterministically*. As it was noted before, QMC techniques produce deterministic sequences of points that provide the best-possible spread over the integration domain.

In general, it is possible to define an integration rule that yields a prescribed level of accuracy in advance. It is just necessary to replace the MC random samples by well-chosen deterministic points. The discrepancy can be regarded as a quantitative measure of the deviation from a uniform distribution.

The deterministic sequences are often referred to as low-discrepancy sequences. The Sobol sequence [79] is a well-known example of a low-discrepancy sequence.

In Figure 2.4, we present a simple example of the comparison between Sobol and pseudorandom distribution points. An effective way to use the QMC method is by performing a change of variables to reduce integration to the $[0, 1]$ domain. When we need to integrate over a large domain $[a, b]$, that avoids multiplying the QMC estimate by a large factor $(b - a)$ as required by Equation (2.2).

A QMC advantage with respect to MC is that its error is $O\left(\frac{1}{N}\right)$, while the MC error is $O\left(\frac{1}{\sqrt{N}}\right)$, where N is the sample size. The Koksma-Hlawka inequality [54] bounds the error of QMC estimates, but in practical applications it is very hard to estimate [37], thereby hampering the use of QMC methods. As such, other methods for estimating the QMC error need to be developed.

Suppose we want to compute $I = \int_{U_d} f(x)dx$, where U_d is the hypercube over $[0, 1]^d$. Let $\{u_1, ..., u_n\}$ be a set in U_d. Then the Koksma-Hlawka inequality [54] is:

$$\left| I - \frac{1}{n} \sum_{i=1}^{n} f(u_i) \right| \leqslant V(f) D_n^*\{u_1, ..., u_n\}, \tag{2.4}$$

where $V(f)$ is the bounded variation in the sense of Hardy and Krause:

$$V(f) = \sum_{k=1}^{d} \sum_{1<i_1<...<i_k<d} V_{Vit}^k(f; i_1, ..., i_k),$$

where $V_{Vit}^k(f; i_1, ..., i_k)$ is the variation in sense of Vitali [86], applied to the restriction of f to the space dimension $k\{(u_1, ..., u_d) \in [0, 1]^d : u_j = 1 \text{ for } j \neq i_1, ..., i_k\}$. If $k = d$ we obtain an empty set, which cannot be calculated.

The star-discrepancy D_n^* is defined as follows:

$$\mathbb{D}_n^*\{u_1, ..., u_n\} = \underset{B \in W^*}{sup} \left| \frac{\#\{u_i : u_i \in B\}}{n} - \lambda_d(B) \right|,$$

where $\#\{u_i : u_i \in B\}$ are points from the set B and W^* is defined as the set of the form:

$$\prod_{k=1}^{d} [0, c_k) = \{y \in U_d : 0 \leqslant y_k < c_k\}.$$

Unfortunately, Equation (2.4) can not serve as a basis for a constructive evaluation of the integration error in practical applications. In particular, computing the star-discrepancy of an arbitrary set is an NP-hard problem [37]. Also, estimating the Hardy-Krause variation is a very complicated computational problem.

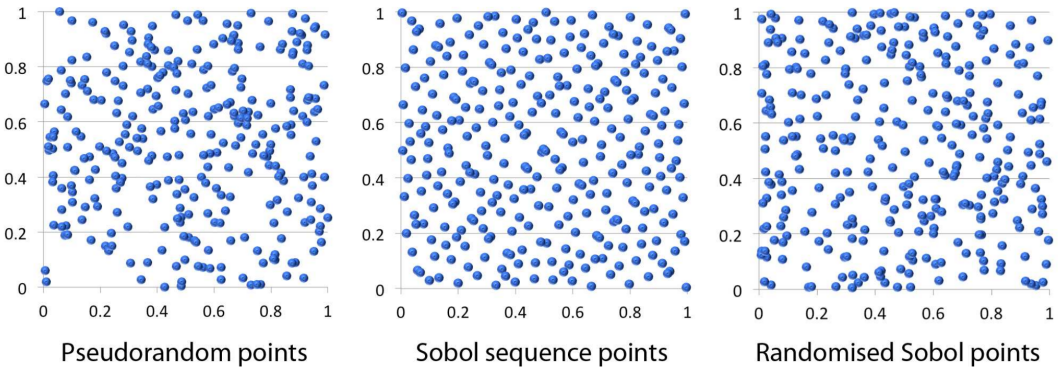


Figure 2.4: Sobol sequence, uniform pseudorandom and randomised Sobol sequence points (obtained by transformation $\Gamma = (\mathfrak{X}+\epsilon)mod1$, where ϵ is a random sample from MC sequence and $\mathfrak{X}$ is low-discrepancy sample from Sobol sequence) distribution in the 2-dimensional unit space. The comparison is based on the first 300 points of sequences.

2.3.4 Randomised Quasi-Monte Carlo Simulations

One of the main problems with QMC, as discussed earlier, is the difficulty of computing an estimate of the integration error. Theoretically, it is possible to estimate an upper bound but it is far from being possible in all cases. An efficient way to solve this problem is to use Randomised Quasi-Monte Carlo (RQMC). Allowing the randomisation technique into the deterministic QMC procedure enables constructing CIs to estimate the error.

This gives us the best out of two methods: a tool to estimate a confidence interval of the error by using randomisation while keep the accuracy of QMC. RQMC can be regarded as a trade-off between sacrificing some precision in order to get a better error estimation. One of the main disadvantages of RQMC as well as QMC is that the method is only applicable when using inversion methods for sampling from the distribution.

There are different randomisation methods that can be used, see *e.g.* [57]. In this thesis, I use the shifting method, as described in [86].

In general an RQMC procedure can be described as follows. Suppose that $\mathfrak{X} = \{x_1, ..., x_n\}$ is a deterministic low-discrepancy set. By means of a transformation $\tilde{\mathfrak{X}} = \Gamma(\mathfrak{X}, \epsilon)$ a finite set $\tilde{\mathfrak{X}}$ is generated by the random variable ϵ and has the same

quasi-random properties as set $\mathfrak{X}$ (see Figure 2.4). For a randomised set $\tilde{\mathfrak{X}}_i$ we construct a RQMC estimate similar to (2.2):

$$RQMC_{j,n} = \frac{1}{n}\sum_{i=1}^{n} f(\tilde{\mathfrak{X}}_{i,j}) \tag{2.5}$$

for $0 < j \leqslant r$, where r is the total number of different pseudo-random sequences. Then, we take their average for the overall RQMC estimation (2.5):

$$RQMC_n = \frac{1}{r}\sum_{j=1}^{r} RQMC_{j,n}. \tag{2.6}$$

If we choose the Γ transformation in such a way that each of the estimates $RQMC_{j,n}$ has the unbiasedness property, *i.e.*, $\forall j\ \mathbb{E}\left[RQMC_{j,n}\right] = I$, (*e.g.* $\Gamma = (\mathfrak{X} + \epsilon) \bmod 1$), then the estimator (see Equation 2.6) will also be unbiased, *i.e.*, $\mathbb{E}[RQMC_n] = I$. By independence of the samples used in Equation (2.5) and Equation (2.6), we have that for all $0 < j \leqslant r$:

$$Var(RQMC_n) = \frac{Var(RQMC_{j,n})}{r}.$$

Thus, we have the following variance estimation:

$$\widehat{Var}(RQMC_n) = \frac{1}{r(r-1)}\sum_{j=1}^{r}\left(RQMC_{j,n} - RQMC_n\right)^2.$$

2.3.5 Quadrature Formulas

The QMC methods presented in Subsection 2.3.3 can be improved by finding more effective deterministic sequences, which potentially can produce a new method for QMC variance estimation and thus allow us to construct CIs using fewer number of samples. One of such methods is considered in this Subsection.

The QMC method has a natural interpretation in terms of integrand functions decomposition in a series of piecewise constant functions. Many authors associate the remainder of integration with families of Haar functions [51] and Walsh functions [20], so that the remainder is determined by the asymptotic behaviour of such decomposition. On this basis, there arises the idea of constructing such a random quadrature formula that would be exact for the first n functions from the Haar function families.

The Generalised Haar System. The system of Haar functions $h_r(x)$, $(r = 0, ..., N-1)$ proposed by Haar in [38] is defined on the interval $0 \leq t \leq 1$. A similar idea of the generalised Haar system, but with a different approach to the definition, is presented in [34].

By analogy with the definitions given by Sobol in [79] the notation for an arbitrary natural r depends on two parameters k and i and can be shown as its decomposition in the form $r = 2^k + i$, where $i \in 1, ..., 2^k$ and $k \in \mathbb{N}_0$. For any value of $r \geq 0$, k and i are uniquely determined so that i is the remainder $r - 2^k$ and 2^k is the largest power of 2 contained in k $(2^p < k)$.

The Haar function for the r^{th} element is defined as follows:

$$h_1(x) = 1,$$
$$h_r(x) = \begin{cases} 2^{k/2}, & x \in l^{k+1}2i - 1 \\ -2^{k/2}, & x \in l^{k+1}2i \\ 0, & x \in [0,1] \setminus \bar{l}_i^k, \end{cases} \tag{2.7}$$

where $\bar{l}_i^k$ is the closure of $l_i^k = (\frac{i-1}{2^k}, \frac{i}{2^k})$. Moreover, all functions are continuous on the right at zero and continuous on the left at one, and at the inner points of the discontinuity, the Haar functions are defined as the half-sum of the left and right limits.

The Haar system (see Equation 2.7) is orthonormal in $L^2[0.1]$. In addition, Schauder [69] showed that the Haar system is a basis for all $L^2[0.1]$ spaces. Other important properties of the Haar system are described in detail in Sobol's book [78].

Qint Quadrature Construction and Variance Analysis. Ermakov and Antonov [1] have introduced a new method for QMC variance estimation. To construct an estimate of the integral I they use the set of random quadrature formulas, introduced by the Ermakov-Granovsky theorem [25]. This theorem allows us to construct N-point formulas with two important properties: the unbiasedness property for integral I and the accuracy property for the considered Haar

system. The nodes of the formula are random variables with distribution density:

$$\phi(u_1, u_2, ..., u_N) = \begin{cases} \frac{N^N}{N!} & \text{if } (u_1, u_2, ..., u_N) \in Lat(i_1, i_2, ..., i_N) \\ 0 & \text{otherwise,} \end{cases}$$

where $Lat(i_1, i_2, ..., i_N)$ is a Latin set that relates to the permutation $(i_1, i_2, ..., i_N)$ and can be defined by the next condition:

$$(u_1, u_2, ..., u_N) \in Lat(i_1, i_2, ..., i_N) \Leftrightarrow \forall j \in \{1, 2, ..., N\} u_j \in \mathcal{U}_{i_j},$$

where $\mathcal{U}_{i_j}$ is a set of permuted orthonormal Haar functions [1].

The variance of the constructed cubature formula $Cub[f] = \frac{1}{N}\sum_{i=1}^{N} f(u_i)$ can be calculated as:

$$\mathbb{D}Cub[f] = \int_{\mathcal{U}^N} Cub[f]^2 d\phi - \left(\int_{\mathcal{U}^N} Cub[f] d\phi \right)^2 =$$

$$= \mathbb{D}_{MC}[f] + \frac{1}{N}(a_1 + a_2 + ... + a_N)^2 - a_1^2 - a_2^2 - ... - a_N^2 = \mathbb{D}_{MC}[f] - \frac{1}{N}\sum_{i<j}(a_i - a_j)^2,$$

where $\mathbb{D}_{MC}$ is the variance of MC method (see Equation 2.3) and $a_i = \int_{\mathcal{U}_i} f(u)\mu(du)$ for $i = 1, 2, ..., N$.

In other words, it allows us to calculate the integral estimation variance as:

$$Var(QMC) = Var(MC) - \frac{1}{N}\sum_{i<j}(a_i - a_j)^2 \ . \tag{2.8}$$

2.4 Confidence Interval Estimation and Error Analysis

The construction of a Confidence Interval (CI) enables us to estimate the true mean and true standard deviation of a function via the mean and standard deviation of the gathered sample data. However, the most important questions here are: how dependable are the sample data at representing the population data? Can we precisely estimate the population data by using sample data? Therefore, when constructing the mean of the obtained sample data it is necessary to indicate the reliability of the data, *i.e.* the quality of the estimation of the true function mean from the sample mean. This can be done by using CIs. CIs are usually represented in the form of a range with the center-mean and bounds,

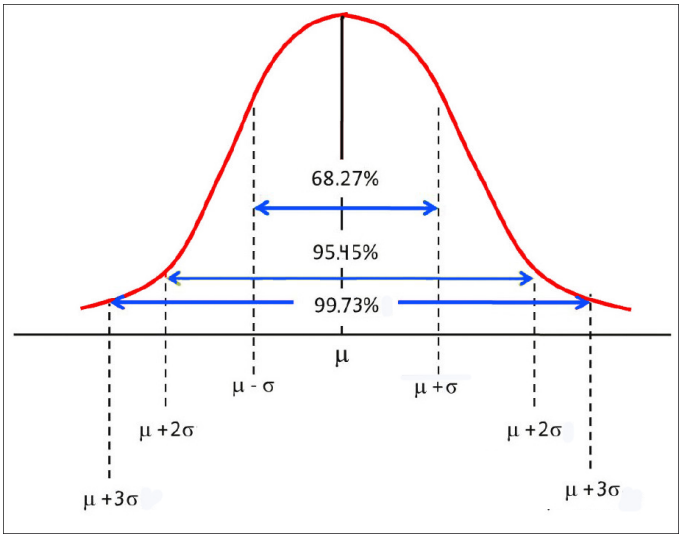


Figure 2.5: The 68–95–99.7 rule indicating 68.27%, 95.45% and 99.73% confidence intervals construction.

representing the probability of observing the true mean. The confidence of the interval can be chosen manually, however, choices common CIs are 65%, 95% and 99%.

For this case, the standard deviation distribution is used to calculate confidence intervals for the population standard deviation. In Figure 2.5 the 68–95–99.7 rule, also known as the empirical rule, is presented. It is used to show the different percentage confidence bounds, which can be constructed in a normal distribution by using mean and standard deviation - more accurately, 68.27%, 95.45% and 99.73% of the values lie within one, two and three standard deviations of the mean, respectively.

In the following the next notation is used:

- $\tilde{X} = \frac{1}{n}\sum_{i=1}^{n} x_i$ - the sample mean.
- $C_a = Quant(1-\frac{a}{2})$ - the inverse cumulative distribution function of a normal random variable with mean 0 and standard deviation 1; parameter a defines the confidence level at $1 - a$.
- $\hat{p} = n_s/n$ - the binomially-distributed proportion, where: n_s - the number

of "successes" and n_f - the number of "failures" in a Bernoulli trial process; n - the total number of Bernoulli "trials".

- $\hat{q} = 1 - \hat{p}$.

2.4.1 Intervals Based on the Beta-Function

The Beta function represented by a parametrized family of Beta distributions $f(\cdot; \mathbf{v})$ with $\mathbf{v} = \{\alpha, \beta\}$, where $\alpha \geq 1$ and $\beta \geq 1$ are the parameters of a Beta distribution [72]. The probability density function of a Beta distribution is the function:

$$f(x, \alpha, \beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha, \beta)},$$

where $B(\alpha, \beta)$ – Beta function (see more information in Section 4.3 of [72]).

Bayesian Interval This method is based on the assumption that the (unknown) probability p to estimate is a random quantity [97]. The Bayesian interval is also called "credible", because it computes the posterior distribution of the unknown quantity by using its prior distribution and Bayes theorem. The prior distribution can be constructed by means of the *Beta* distribution, which is widely used for computing inferences on p. If p has a prior distribution $Beta(\alpha, \beta)$ then p has posterior distribution $Beta(n_s + \alpha, n - n_s + \beta)$. We can construct a Bayesian equal-tailed interval by the formula:

$$CI_B = \left(Beta^{-1}(\frac{a}{2}, n_s + \alpha, n - n_s + \beta), Beta^{-1}(1 - \frac{a}{2}, n_s + \alpha, n - n_s + \beta) \right), \tag{2.9}$$

where, $Beta^{-1}(a, \alpha, \beta)$ is the inverse of the cumulative distribution function of $Beta(\alpha, \beta)$.

Jeffreys Interval The Jeffreys interval is a Bayesian interval and uses the Jeffreys prior [50], which involves a non-informative prior given by the *Beta* distribution with parameters $(\frac{1}{2}, \frac{1}{2})$. We can form the Jeffreys equal-tailed interval by Equation (2.9) with parameters $(\frac{1}{2}, \frac{1}{2})$.

Clopper-Pearson Interval This method was introduced by Clopper and Pearson in 1934 [14] and is based on the inversion of binomial test, rather than on approximations. The Clopper-Pearson interval is:

$$CI_{CP} = \left(Beta^{-1}(\frac{a}{2}, n_s, n - n_s + 1), Beta^{-1}(1 - \frac{a}{2}, n_s + 1, n - n_s)\right) . \quad (2.10)$$

The CI_{CP} interval states that the computed coverage probability is always above or equal to the $1 - a$ confidence level. In practice, it can be achieved in cases when n is large enough, while in general, the actual coverage can exceed $1 - a$. We can conclude from Equation (2.10) that due to the absence of the α and β parameters, the appropriate result can be achieved only by increasing number of "trials".

2.4.2 Intervals Based on the CLT Interval

The Central Limit Theorem (CLT) states that if we assume that all samples are independent and distributed identically then the distribution of sample means approximates a normal distribution simultaneously with the increase of the sample size, regardless of population distribution shape. In other words, the CLT states that the mean of all samples from the same population will be approximately equal to the mean of the population, given that there is a sufficiently large number of samples from a population with finite variance. Furthermore, all variances are approximately equal to the variance of the population divided by each sample's size and all the samples follow an approximately normal distribution. In this Subsection, I consider the most effective CIs which are based on the CLT interval.

Wilson Interval It was introduced by Wilson in 1927 in his fundamental work [21] and uses the inversion of the CLT interval. Additionally, it involves a modified center by quantile formula mean value. The interval has the following form:

$$CI_W = \left(\frac{n_s + \frac{C_a^2}{2}}{n + C_a} - \frac{C_a\sqrt{n}}{n + C_a^2}\sqrt{\hat{p}\hat{q} + \frac{C_a^2}{4n}}; \frac{n_s + \frac{C_a^2}{2}}{n + C_a} + \frac{C_a\sqrt{n}}{n + C_a^2}\sqrt{\hat{p}\hat{q} + \frac{C_a^2}{4n}}\right) . \quad (2.11)$$

This interval has some obvious advantages - it can not exceed the probability boundaries, and it can be easily calculated even if $\hat{p}$ is 0 or 1. At the same time, CI_W has downward spikes when $\hat{p}$ is close to 0 and 1, because it is formed by an

inverted CLT approximation.

Agresti-Coull Interval This method was introduced by Agresti and Coull in 1998 [5]. One of the most interesting features of this CI is that it makes a crucial assumption about n_s and n_f. This interval formally adds two successes and two failures to the obtained values in case of 95% confidence level and then uses the CLT method. The interval can be constructed as follows:

$$CI_{AC} = \left(\tilde{X} - \frac{1}{n + C_a^2}(n_s + \frac{1}{2}C_a^2); \tilde{X} + \frac{1}{n + C_a^2}(n_s + \frac{1}{2}C_a^2) \right). \tag{2.12}$$

Additionally, this interval can be modified by using the center of the Wilson interval (see Equation 2.11) in place of $\hat{p}$:

$$CI_{AC_W} = \left(\frac{n_s + \frac{C_a^2}{2}}{n + C_a} - C_a\sqrt{\hat{p}\hat{q}(n + C_a^2)}; (\frac{n_s + \frac{C_a^2}{2}}{n + C_a} - C_a\sqrt{\hat{p}\hat{q}(n + C_a^2)} \right). \tag{2.13}$$

Logit Interval The Logit interval is based on a transformation of the standard interval [19]. It uses the empirical logit transformation: $\lambda = ln(\frac{\hat{p}}{1-\hat{p}}) = ln(\frac{n_s}{n-n_s})$. The variance of λ is: $\widehat{Var}(\lambda) = \frac{n}{n_s(n-n_s)}$ and the Logit interval can be estimated as:

$$CI_L = \left(\frac{e^{\lambda_L}}{1 + e^{\lambda_L}}, \frac{e^{\lambda_U}}{1 + e^{\lambda_U}} \right), \tag{2.14}$$

where the lower bound transformation is $\lambda_L = \lambda - C_a\sqrt{\widehat{Var}(\lambda)}$ and the upper bound transformation is $\lambda_U = \lambda + C_a\sqrt{\widehat{Var}(\lambda)}$.

Anscombe Interval This interval was proposed by Anscombe in 1956 [9] and is based on the Logit interval (see Equation 2.14). The key difference is in λ and $\widehat{Var}(\lambda)$ estimation, where λ is defined as $\lambda = ln(\frac{n_s+\frac{1}{2}}{n-n_s+\frac{1}{2}})$ and the variance is $\widehat{Var}(\lambda) = \frac{(n+1)(n+2)}{n(n_s+1)(n-n_s+1)}$. On this basis, the Anscombe interval CI_{Anc} is estimated in the same way as the Logit interval (see Equation 2.14).

Arcsine Interval It uses a variance-stabilising transformation of $\hat{p}$. In 1948, Anscombe introduced an improvement [8] for achieving better variance stabilisa-

tion by replacing $\hat{p}$ to $p^{\dagger} = \frac{n_s+3/8}{n+3/4}$, obtaining

$$CI_{Arc} = \left(\sin(\arcsin(\sqrt{p^{\dagger}}) - \frac{C_a}{2\sqrt{n}})^2, \sin(\arcsin(\sqrt{p^{\dagger}}) + \frac{C_a}{2\sqrt{n}})^2 \right). \quad (2.15)$$

2.5 Gaussian Processes

We can also look at the evaluation of the approximate probabilistic reachability in SnPHSs from the data collection and prediction point of view. In this case, we compose a training data set by sampling the probability values at a number points from the nondeterministic parameters domain. As a result of the training process, we have a set of points and a set of probability values, so that we can then make a prediction for other points from the same nondeterministic domain without new sampling process. Our aim is to minimize the amount of time and data which we need to produce the most accurate evaluation of the probability function. In order to solve this problem machine learning techniques can be successfully applied.

It is possible to generally divide machine learning algorithms into three main groups: supervised learning, unsupervised learning, and reinforcement learning. Supervised learning is known to be the most thoroughly studied and easiest to perform. It concerns learning a relationship from inputs to targets (or outputs). Supervised learning can be subdivided into two primary tasks: classification and regression. In the former, the outputs are discrete labels, while in the latter the outputs are continuous variables.

In this thesis I will focus on the regression task, as GP models, in their simplest form, are used for regression. Despite the fact that the regression problem is one of the basic and most general statistical problems, it underlies many machine learning tasks. Therefore, reliable general methods for regression are crucial for the whole field of study and can be applied in more advanced and specific learning tasks.

GPs are considered to be a simple and general type of probability distributions on functions. They were first applied for time series prediction [45, 91]. A probability distribution on functions $p(f)$ is defined by means of a GP. This can be used as a Bayesian prior for the regression, and Bayesian inference can be

applied to make predictions from data (*i.e.* building the posterior distribution).

In this section, I define GP regression and classification and show how they can naturally be used to define distributions over functions.

2.5.1 Gaussian Process Regression

A GP is a collection of random variables, any finite number of which have a joint multidimensional Gaussian distribution. A Gaussian process can be described as a generalisation of the Gaussian probability distribution. The major advantage of GPs is that inference can obtain properties of the function at a finite number of points, ignoring infinitely many points, with the same quality as if we would have taken them all into account [61].

GPs can be fully described by the mean and covariance functions. The mean function $m(\mathbf{x})$ shows the expected data taken before any observations. The covariance function $k(\mathbf{x}, \mathbf{x}')$, which is also known as kernel function represents the expected correlation between the observations $\mathbf{x}$ and $\mathbf{x}'$. In general, these functions are defined as follows:

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})],$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))].$$

We can then model any function using GP as:

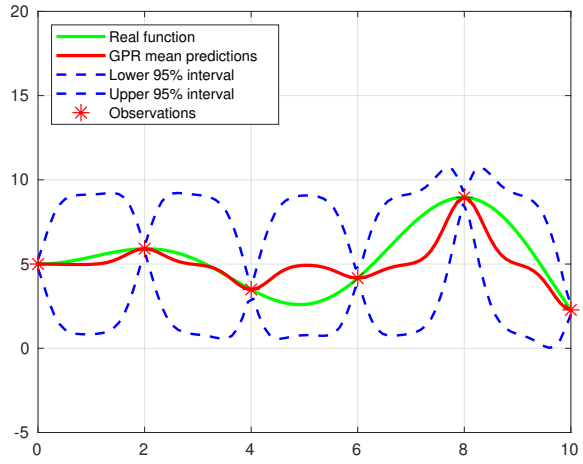
$$f(\mathbf{x}) \sim GP(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')).$$

Depending on the initially known properties of the function we need to choose the initial mean and covariance functions. In most cases, a zero mean function and squared exponential (SE) covariance kernel function are suggested for these purposes:

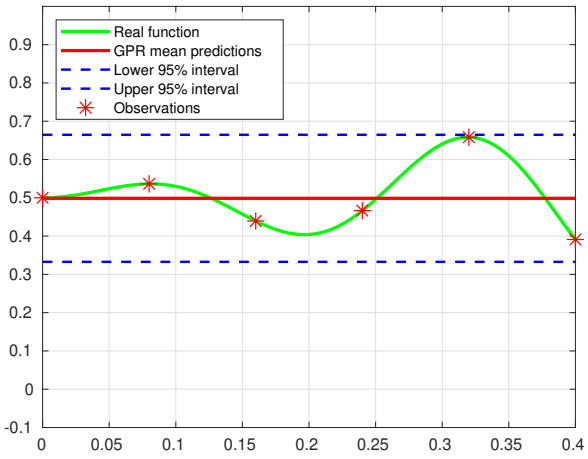
$$m(\mathbf{x}) = 0,$$

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 exp\big[-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^T diag^{-1}(\mathbf{x} - \mathbf{x}')\big] + \delta_{\mathbf{x},-\mathbf{x}'}\sigma_w^2,$$

where σ_f is the variance of the kernel function, δ is the Kronecker delta, σ_w is the

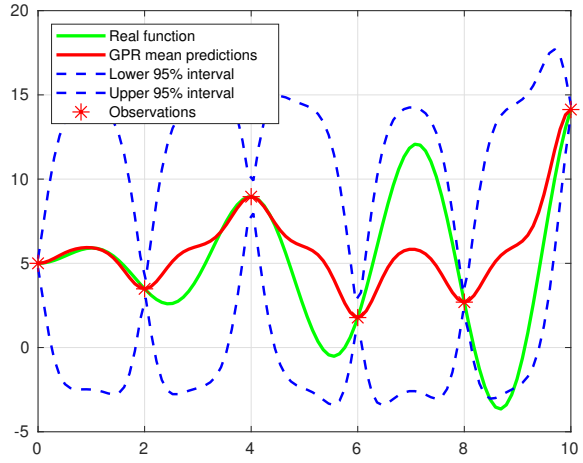


(a) Large integration domain [0,10]

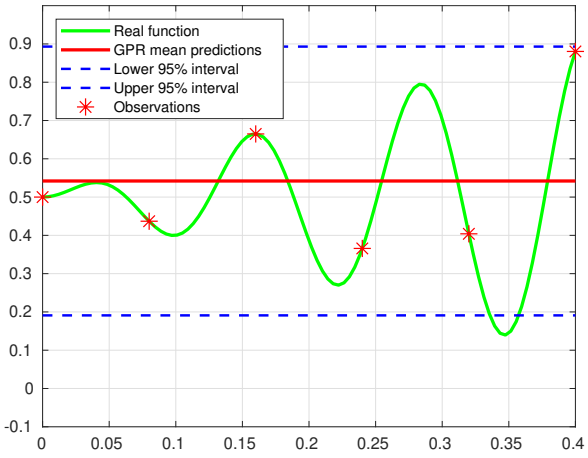


(b) Small integration domain [0,0.4]

Figure 2.6: Dependence of GP predictive distribution with SE parameters a=1 and λ=1 on the integration domain of a smooth function. The single test point predictive distribution is presented as mean and two 95 % standard deviation. The real function is shown in green.



(a) Large integration domain [0,10]



(b) Small integration domain [0,0.4]

Figure 2.7: Dependence of GP predictive distribution with SE parameters a=1 and λ=1 on the integration domain of an oscillating function. The single test point predictive distribution is presented as mean and two 95 % standard deviation. The real function is shown in green.

noise variance and $diag^{-1}$ is the inverse of diagonal length-scales matrix. Given N samples $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_N)$ we build a probability distribution over functions by using the next Gaussian joint distribution:

$$p(\mathbf{y}) \sim \mathcal{N}(0, \mathbf{K}(\mathbf{X}, \mathbf{X})), \tag{2.16}$$

where $\mathbf{K}(\mathbf{X}, \mathbf{X})$ is the covariance matrix, composed by the chosen kernel and mean functions.

The main aim of the GP method is to obtain the posterior distribution. After training our simulation model with the training set $\mathbf{X}$ we collect the test set $\mathbf{X}_*$. This set defines the points where we want to receive the function prediction. From Equation (2.16) we can show the joint distribution of the unknown $\mathbf{y}_*$ using the known $\mathbf{y}$:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{y}_* \end{bmatrix} = \mathcal{N}\left(0, \begin{bmatrix} \mathbf{K}(\mathbf{X}, \mathbf{X}) & \mathbf{K}(\mathbf{X}, \mathbf{X}_*) \\ \mathbf{K}(\mathbf{X}_*, \mathbf{X}) & \mathbf{K}(\mathbf{X}_*, \mathbf{X}_*) \end{bmatrix}\right).$$

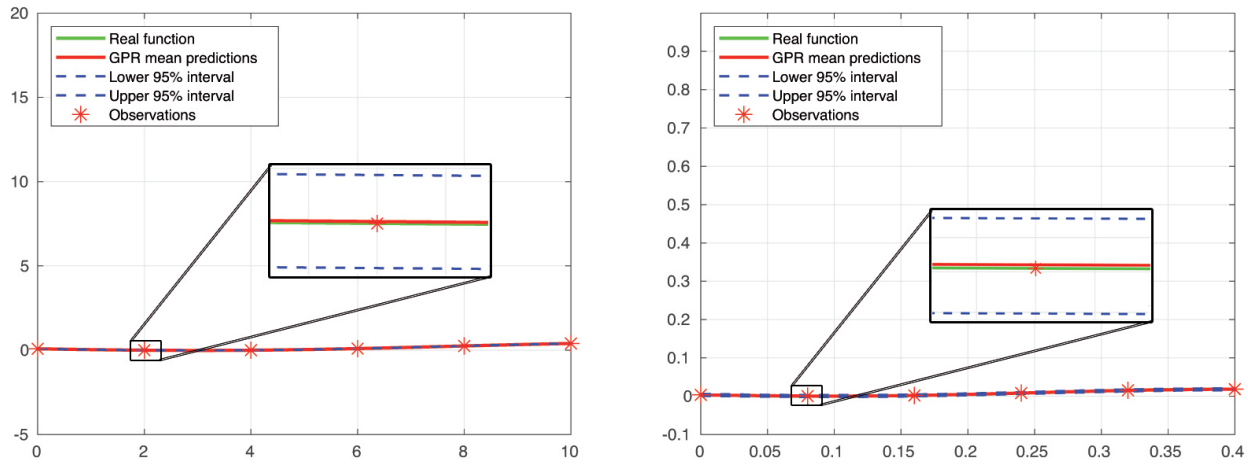
We can now define mean and covariance of the conditional Gaussian distribution for the posterior distribution $p(\mathbf{y}) \sim \mathcal{N}(0, \mathbf{K}(\mathbf{X}, \mathbf{X}))$ as:

$$\mathbb{E}[\mathbf{y}_*|\mathbf{y}, \mathbf{X}, \mathbf{X}_*] = \mathbf{K}_*^T \beta,$$

$$var[\mathbf{y}_*|\mathbf{y}, \mathbf{X}, \mathbf{X}_*] = \mathbf{K}_{**} - \mathbf{K}_*^T (\mathbf{K} + \sigma_w^2 \mathbf{I})^{-1} \mathbf{K}_*,$$

where $\mathbf{K}_* = k(\mathbf{X}, \mathbf{X}_*)$, $\mathbf{K}_{**} = k(\mathbf{X}_*, \mathbf{X}_*)$, $\mathbf{K} = k(\mathbf{X}, \mathbf{X})$ and $\beta := (\mathbf{K} + \sigma_w^2 \mathbf{I})^{-1} \mathbf{y}$.

In this Section I consider three different types of functions for GP approximation: simple smooth functions (see Figure 2.6), oscillating functions (see Figure 2.7) and rare-event functions, whose values are very close to zero (see Figure 2.8). The shape of the approximant function significantly affects the approximation results. For example, comparing in Figure 2.6 (a) and Figure 2.7 (a) the CIs constructed over the same domain by using the SE function with the same parameters we can easily note that Figure 2.6 (a) has better approximation. The CI shown in Figure 2.7 (a) is almost twice wider, which leads us to the conclusion that the accuracy of the GP approximation is directly correlated with the unknown function shape. Another key point for good approximation by GP is an



(a) Large integration domain [0,10]

(b) Small integration domain [0,0.4]

Figure 2.8: Dependence of GP predictive distribution with SE parameters a=1 and λ=1 on the integration domain of a rare-event function (near 0 bound). The single test point predictive distribution is presented as mean and two 95 % standard deviation. The real function is shown in green.

appropriate number of sample points, which unfortunately can not be predicted in advance without any knowledge about the function being sampled. However, in case of very small values (near 0 bound), GPs approximate a rare-event function very successfully (see Figure 2.8 (a)).

Although most GP sources [12, 61] consider only well-behaved functions over large domains, it is necessary to pay attention to small domains as well. In the real world we deal with very hardly predictable hybrid systems, whose behaviour vary significantly. In this thesis, we estimate functions predominantly on a relatively small domain, where function values change only within the [0,1] interval (probability functions). Unfortunately, if we keep the same simple functions features, which were used for a large domain (see Figure 2.6 (a)) and apply it to a smaller one (see Figure 2.6 (b)) we will see that GPs are not as effective as before. The same picture can be seen for a curved function as well (see Figure 2.7 (a) and (b)). The confidence interval bounds represent now a flat line despite the information obtained from our data observation. Though for rare event case an effective GP behaviour still remains (see Figure 2.8 (b)).

2.5.2 Covariances and Hyperparameter Learning

It was already stated in Subsection 2.5.1 that it is necessary to define the mean and covariance function to specify a particular GP prior. Most of the properties of sample functions drawn from the GP prior can be determined by the covariance function (e.g. smoothness, lengthscales, amplitude etc). Therefore, one of the key steps in GP modelling is selecting an appropriate covariance function for a particular problem or hybrid system model.

In fact, there is a wide range of flexible and computationally efficient covariance functions. The reader can find more information about such functions in [61]. In this research, the SE covariance function was chosen for the tests and implementation as one of the most efficient. It is widely used in machine learning. It very efficiently draws sample functions that are infinitely differentiable (*i.e. smooth*):

$$K_{SE}(r) = a^2 exp(-\frac{r^2}{2\lambda^2}), \tag{2.17}$$

where $r = \|\mathbf{x} - \mathbf{x}'\|$. The SE covariance is stationary (a function of $\|\mathbf{x} - \mathbf{x}'\|$ – invariant to translations) and more practically isotropic (a function of r – invariant to both rotation and translation). It is governed by two hyperparameters a and λ, that change properties of approximant functions: λ controls the typical lengthscale of variation and a controls the typical amplitude.

Figures 2.9 (a) and 2.9 (b) show GP function estimation obtained by using the SE covariance function with two different λ hyperparameters: 0.15 and 9.5 respectively. It easily can be seen that the confidence interval shape and the mean line for these pictures significantly differ from one with the SE default parameters settings (see Figure 2.6 (a)). A notable increase of the λ parameter transforms CIs into a line while decreasing change the CIs form.

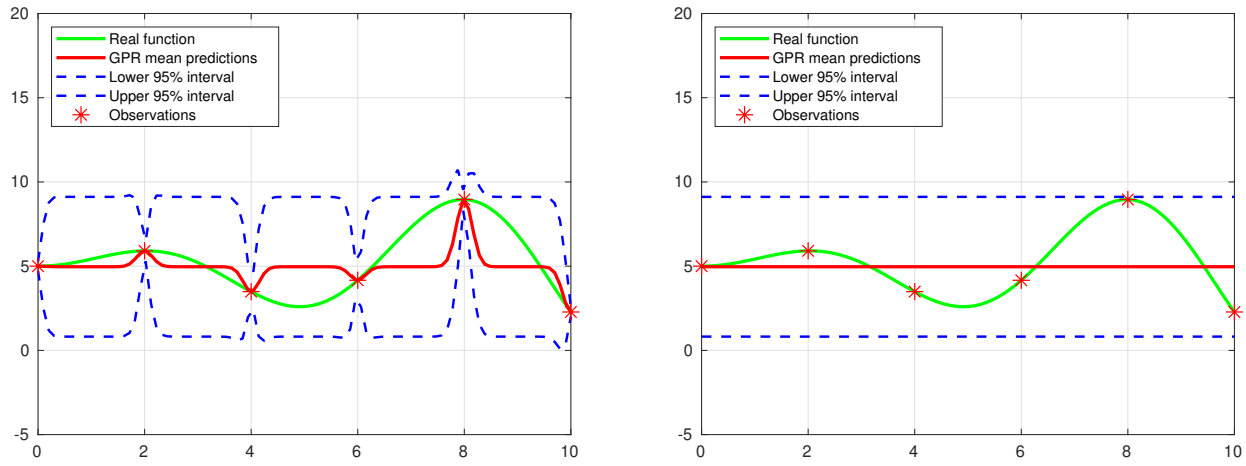
Figures 2.10 (a) and 2.10 (b) compare GP estimation with the same λ hyperparameters equal to 0.15 and 9.5. It can be noted now that the plot with the λ parameter equal to 9.5 (see Figure 2.10 (b)) has better approximation in comparison to Figure 2.9 (b).

Finally, in Figures 2.11 (a) and 2.11 (b) the results with λ hyperparameters 0.01 and 0.15 for GP function of Figure 2.8 are presented. It can be seen that

the CIs for both figures are significantly larger. In Figure 2.11 (a) a very bad flat approximation is presented, while in Figure 2.11 (b) a small correlation towards the observed point is present.

It can easily be seen that the smoothness of the sample functions arises from the form of the covariance Equation (2.17). We can see a strong correlation between those variables, which are close in the input space. At the same time, function variables far apart relative to the lengthscale λ are uncorrelated. One of the possible disadvantages of the SE covariance is that in realistic regression tasks it may be unreasonably smooth.

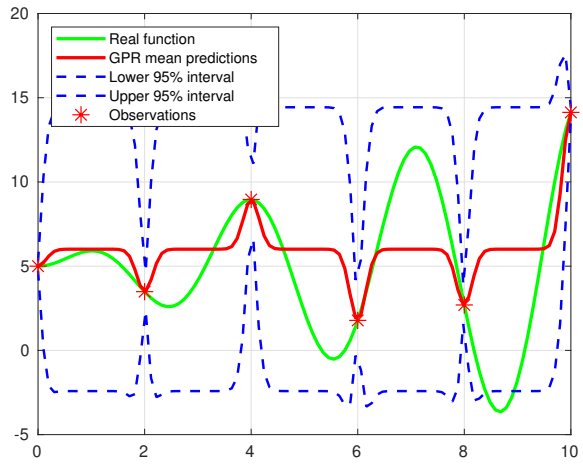
The reader can note now that the choice of SE hyperparameters is not universal for all function kinds and should be considered closely in every particular case. More information about covariance functions choice strategies alongside with their comparison can be found in [61].



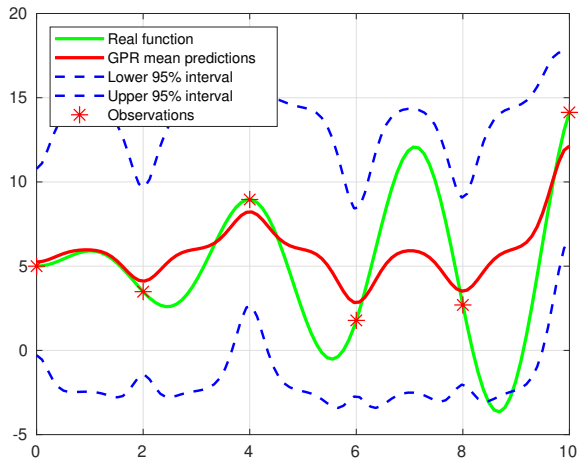
(a) SE, $\lambda = 0.15$ (b) SE, $\lambda = 9.5$

Figure 2.9: Gaussian process regression with the SE covariance function λ parameter equals to (a) and (b). The same real function is presented in Figure 2.6.

One of the most exciting features of GPs in comparison with other methods is the possibility to choose covariance hyperparameters from the training data directly. This can be especially effective in case other hyperparameters selecting methods, *e.g.* cross-validation, can not be successfully applied.

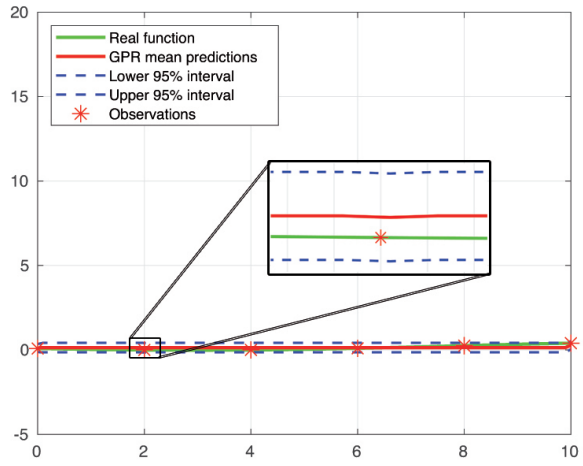


(a) SE, $\lambda = 0.15$

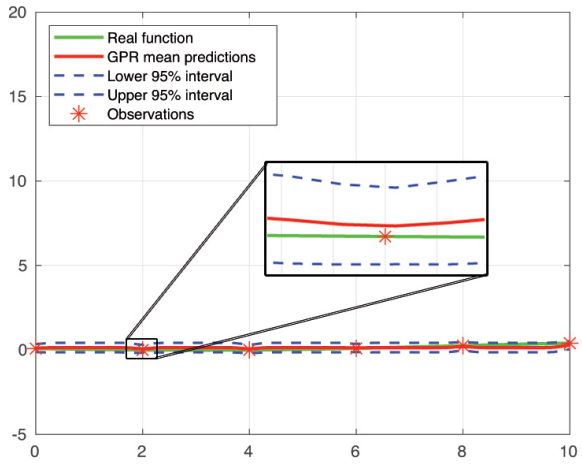


(b) SE, $\lambda = 9.5$

Figure 2.10: Gaussian process regression with the SE covariance function λ parameter equals to (a) and (b). The same real function is presented in Figure 2.7.



(a) SE, $\lambda = 0.01$



(b) SE, $\lambda = 0.15$

Figure 2.11: Gaussian process regression with the SE covariance function λ parameter equals to (a) and (b). The same real function is presented in Figure 2.8.

In an ideal world, we would like to place a prior and compute a Bayesian posterior $p(\theta|y)$ according to the hyperparameters. Unfortunately, computing

the posterior is not analytically tractable in general. In order to solve this issue, the marginal likelihood can be used as an appropriate cost function. In other words, we aim to minimise the negative *log* marginal likelihood $\mathcal{L}$ with respect to the hyperparameters of the covariance θ:

$$\mathcal{L} = -log\, p(y|\theta) = \frac{1}{2} log\, det \mathbf{C}(\theta) + \frac{1}{2} y^T \mathbf{C}^{-1}(\theta) y + \frac{N}{2} log(2\pi), \tag{2.18}$$

where $\mathbf{C} = \mathbf{K}_N + \sigma^2 \mathbf{I}$ is the covariance matrix and $\mathbf{I}$ is the identity matrix.

In any other operations where we use the training data to optimise parameters, we need to worry about overfitting. Practically, GP processes do not optimise f function variables by themselves, but rather integrate over their uncertainty. The optimisation of the GP hyperparameter occurs at a greater hierarchical level.

The minimisation of the *log* marginal likelihood of Equation (2.18) represents a non-convex optimisation task. When gradients are readily acquired, conventional gradient optimisers can, therefore, be used. Of course, the accurate details will rely on the covariance function selection. The cost of computing the marginal probability and gradients of *log* is again dominated by the inversion of the $\mathbf{C}$ covariance matrix.

Local minima can be an issue, especially if there is a tiny quantity of information, and hence the solution is unclear. Local minima can correspond to alternative, reliable explanations for the information in this scenario (such as low noise and brief lengthscale vs. elevated noise and lengthy lengthscale). Therefore, it is often worthwhile to make several optimisations from random starting points and investigate the different minima.

2.5.3 Gaussian Process Classification

Gaussian Process Classification (GPC) is an effective algorithm for probabilistic classification, where predictions take the form of class probabilities.

The standard binary GPC aims to predict the class membership probability for a new test point $\mathbf{x}_*$, giving a set of training points $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_N]^T$ of size N and correlated class label $\mathbf{y} = [\mathbf{y}_1, ..., \mathbf{y}_N]^T$ of the same size. GPC exploits a latent function f by mapping its values within [0,1] interval using the probit regression function Φ [61]. The idea behind the probit regression is to turn the output of a

model into a class probability, using the cumulative density function of a standard normal distribution: $\Phi(z) = \int_{-\infty}^{z} \mathcal{N}(x|0,1)dx$, where $\mathcal{N}$ is the density of a normal distribution with 0 mean and 1 standard deviation.

The class membership probability $p(y = 1|\mathbf{x})$ can be presented by $\Phi(f(\mathbf{x}))$. Hence, GPC then can be performed by placing a GP prior over the latent function $f(\mathbf{x})$. In other words, similar to the GP regression algorithm (see Subsection 2.5.1), we can compute the distribution of the latent variable $f(\mathbf{x})$ corresponding to the new test input point $\mathbf{x}_*$ in order to make prediction:

$$p(f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) = \int p(f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{f}) p(\mathbf{f}|\mathbf{X}, \mathbf{y}) d\mathbf{f}, \tag{2.19}$$

where $f = [f_1, ..., f_N]^T$. The distribution (see Equation 2.19) can be now applied to draw a full picture of class membership distribution.

$$p(y_* = 1|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) = \int \Phi(f_*)|p(f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) df_*.$$

Unfortunately, the solution of classification problems using Gaussian processes is rather more expensive than the one for the regression problems considered in the previous subsection [61].

2.5.4 Gaussian Process Multiple Annotators Classification

In many situations gold ground truth information is unavailable, so there has been much work on estimating ground truth labels from multiple annotators, for example in the context of bio-statistics and epidemiology [18, 42]. This strategy, however, first estimated the ground truth from estimates of annotators and used probabilistic ground truths to learn a classifier. Some other methods [10, 15, 71] exploit previous knowledge of the labels similarities.

The main difference between GPC and GP learning from multiple annotators classification is that instead of a single class label - y_i for the i^{th} instance, like it was described in Subsection 2.5.3 we need to deal now with a vector of class labels - $\mathbf{y}_i = [\mathbf{y}_i^1, ..., \mathbf{y}_i^R]^T$, representing noisy labels provided by R annotators. In our case we have a set of observations of function **Pr** (see Definition 2.8 from Subsection 2.2.4), obtained by checking probabilistic reachability over a finite set

of random points. In other words per every input point, we have a reachability decision 0 or 1, which forms a row of 0/1 decisions for a particular reachability question.

The latest and most efficient methods in the most general setting, *i.e.* approaches that do not suppose any previous knowledge of the labels or their interactions are presented below.

Majority voting. A commonly used strategy in the case of multiple labels is to use a label that the majority agree on as an estimate of the true label. A concealed true label $\hat{y}_i$ can be calculated as follows for a binary case classification:

$$\hat{y}_i = \begin{cases} 1, & \text{if } (1/R) \sum_{r=1}^{R} y_i^r > 0.5 \\ 0, & \text{if } (1/R) \sum_{r=1}^{R} y_i^r < 0.5, \end{cases} \tag{2.20}$$

where the case of tie ($\hat{y}_i = 0.5$) can be broken by a super-expert or randomly.

Independent approach. The approach demonstrated in [63] assumes the y^r annotator labels are independent of the input features given the true labels exist. This hypothesis may not be precise, particularly when one has easy instances where the annotators are less likely to make mistakes. This approach is graphically represented in Figure 2.12.

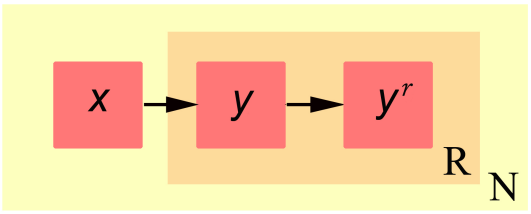


Figure 2.12: The independent multiple annotators classification approach proposed by Rayker et al. [63].

A maximum-likelihood estimator (MLE) that jointly learns the classifier, the annotator accuracy, and the true label represents the basis of this algorithm. The performance of the r^{th} annotator can be measured in terms of the sensitivity and

the specificity with respect to the unknown gold standard. In this thesis, the sensitivity is denoted as α^r and the specificity as β^r respectively. α and β are engaged in distinguishing the true positive rate from the true negative rate. In other words, $\alpha^r = Prob[y^r = 1|y = 1]$ and $\beta^r = Prob[y^r = 0|y = 0]$.

The MLE approach is used to estimate the sensitivity and the specificity and defined as follows:

$$Prob[\mathcal{D}|\theta] = \prod_{i=1}^{N} Prob(y_i^1, ..., y_i^R|x_i, \theta),$$

where $\theta = \{\omega, \alpha, \beta\}$ and ω is the weight vector.

Dependent approach. The previous assumption is relaxed by the work in [92]. It takes into consideration that some annotators are better at labelling certain kinds of information points. Another difference is that [92] assumes that for positive and negative examples, each annotator's performance is symmetric. This approach is graphically represented in Figure 2.13. In this figure, it can be noted that there is an edge between x and y^r that was not present in Figure 2.12. These edge models how the input instance depends on the annotated labels.

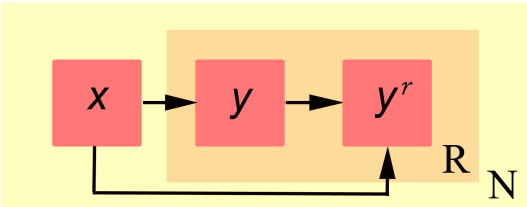


Figure 2.13: The dependent multiple annotators classification approach proposed by Yan et al. [92].

This method exploits the dependence of the input labels via Gaussian distribution as follows:

$$Prob[y_i^{(r)}|x_i, y_i] = \mathcal{N}(y_i^{(r)}; y_i, \sigma(x_i)),$$

where $\sigma(x_i)$ is the variance of the distribution given by $\sigma(x_i) = 1/(1 + e^{-(u^r)^T x_i})$, where u^r is the weight vector for the annotator r.

In [63], the parameters $\theta = \{u_t, \omega_t\}$ are estimated by MLE. This is achieved by maximizing the likelihood function which can be derived as in the Independent approach.

Latent approach. In [66], the authors claim that in instances where the number of classes is large, using ground truth labels as latent variables is inefficient. Another problem with using ground truth labels as latent variables is the explosion of the number of parameters which need to be stored. For example, for K classes, the probability $Prob(y_i^r|y_i)$ for each annotator r and for each class i is needed to be stored. That means that we need to store $K \times K$ parameters for each annotator. Simultaneously with K increasing the number of parameters becomes very large and this often leads to overfitting. The authors propose a formulation which models the annotator accuracies as latent variables to handle this problem. This approach is graphically represented in Figure 2.14.

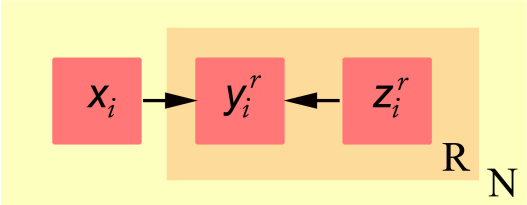


Figure 2.14: The dependent multiple annotators classification approach proposed by Rodrigues et al. [66].

In this work the authors define the variable z_i^r which represents a binary random variable in this case. z_i^r indicates whether the r^{th} annotator labelled the i^th instance correctly or not so that $z_i^r \sim Bernoulli\ \pi_r$, where π_r denotes the annotator accuracy, x_i indicates the features for the i^{th} instance and y_i^r represents the label assigned by annotator r to instance i. Similar to [63], there is no arc from x_i to z_i^r (*i.e.*, the latent variable annotator accuracy in this case does not depend on the input instance labels' characteristics). If for a particular instance i, an annotator r has $z_i^r = 0$, the label assigned to i by r is provided by a random model $P_{rand}(y_i^r = k|x_i)$.

The likelihood function for this case is defined as follows:

$$p(\mathcal{D}, \mathcal{Z}|\theta) = \prod_{i=1}^{N}\prod_{i=1}^{R} p(z_{i=1}^{R}|\pi^r)(p(y_i^r|x_i, z_i^r, w),$$

where $\mathcal{D}$ is the dataset and $\theta(\pi, w)$ are the model parameters: π is the annotators' accuracy vector and w is the classifier weights. The MLE can be simplified by representing logistic regression as the classifier:

$$p(\mathcal{D}, \mathcal{Z}|\theta) = \prod_{i=1}^{N}\prod_{i=1}^{R}(\pi^r p_{LogReg}(y_i^r|x_i, w) + (1 - \pi_r)p_{rand}(y_i^r|x_i)),$$

where $p_{LogReg}(y_i^r|x_i, w)$ is the logistic regression output.

Although in my research I considered and tested the application of the above mentioned multiple annotators techniques, in this thesis I did not use them due to the problems discussed earlier in Section 2.5. More information about learning from multiple annotators alongside with their comparison can be found in [70].

2.5.5 Expectation Propagation Method

In this thesis I use a Bayesian approach for approximating the **Pr** function, which means that **Pr** will be approximated by the posterior distribution of a stochastic process over (the uncertain parameters domain) P given a set of observations of **Pr**. In our case such observations are obtained by checking probabilistic reachability over a finite set of points in P. Hence the likelihood function at each point will be a Bernoulli. For prior I instead choose a GP, which can be adapted to sample from certain types of continuous functions. In particular, I show in Theorem 3.1 that the **Pr** function is smooth, hence I employ GPs which sample from the space of smooth functions over P.

GP posterior inference requires a normal likelihood, while in our case we have Bernoulli's. To solve this problem I follow the approach presented in [11], in which the posterior inference is efficiently handled via the Expectation Propagation (EP) algorithm [53, 56]. More details about the GPEP approach can be found in [61]. EP is an efficient approximate inference algorithm that unifies two techniques - assumed density filtering and loopy belief propagation (an extension of belief

propagation in Bayesian networks). It was developed by Opper and Winther [56] and then adapted for the general case by Minka [53].

EP is an iterative algorithm in which a target density $f(\theta)$ is approximated by a density from some specified parametric family $q(\theta)$. It assumes that our target density $f(\theta)$ has proper factorisation as:

$$f(\theta) \propto \prod_{i=0}^{h} f_i(\theta).$$

The target f is the posterior density $p(\theta|y)$ in the classical case of Bayesian inference. Hence, we can assign one factor as the prior and other factors as the likelihood for one data point. In comparison with GP regression, the EP classification deals with non-Gaussian likelihood. In our case we have a Bernoulli distribution - per every input point, we sample non-deterministic parameters from the certain range and receive $0/1$ decision, which indicates the possibility to reach a goal state. To solve this problem, EP iteratively approximates $f(\theta)$ with a density $q(\theta)$ which takes the same factorisation:

$$q(\theta) \propto \prod_{i=0}^{h} q_i(\theta).$$

This approximation, which associates the factors $f_i(\theta)$ with the approximation $q_i(\theta)$, is usually called *sites* approximation.

At each iteration of the algorithm, and for $i = 1, ..., h$, we take the current approximating function $q(\theta)$ and replace $q_i(\theta)$ by the corresponding factor $f_i(\theta)$ from our target distribution. Now we can define the *cavity* distribution as:

$$q_{-i}(\theta) \propto \frac{q(\theta)}{q_i(\theta)},$$

with the *tilted* distribution equals to:

$$q_{\setminus i}(\theta) \propto f_i(\theta) q_{-i}(\theta).$$

In general, EP at first constructs an approximation $q^{new}(\theta)$ for the tilted distribution $q_{\setminus i}(\theta)$ and then updates approximation to the target density's $f_i(\theta)$, which can be computed as $q_i^{new}(\theta) \propto q^{new}(\theta)/q_{-i}(\theta)$. From the definition $q_i^{new}(\theta)$

can be estimated via the Kullback–Leibler divergence [61]:

$$q_i^{new}(\theta) = \arg\min D(f_i(\theta)q_{-i}(\theta)||q_i(\theta)q_{-i}(\theta)),$$

where $D(||)$ is the Kullback–Leibler divergence. It is important to note that other minimisation methods can be also rather efficiently used for this purpose [53].

At a very simple level the EP algorithm works as follows:

1. Initialisation of the initial site approximation $q_i(\theta)$.
2. Repeat for $i = 1, ..., h$ until all site approximations $q_i(\theta)$ convergence:
   - Compute cavity parameters approximation $q_{-i}(\theta) \propto q(\theta)/q_i(\theta)$;
   - Update site parameters approximation $q_i(\theta)$ and re-compute the posterior parameters so that $q_i(\theta)q_{-i}(\theta)$ approximates as $f_i(\theta)q_{-i}(\theta)$.
3. Return natural site parameters.

In section 5.6 I exhaustively consider the advantages of the EP algorithm application on different models.

2.6 Summary

The Chapter shows verifying bounded reachability in SnPHS options and provides a decision algorithm, which combines the properties of δ-complete decision procedures and formulae $\mathbf{Reach}^{\forall}$ (see Subsection 2.2.4).

It has been explained in this Chapter that reachability, including its bounded version (with a finite number of discrete transitions in the reachability analysis), is undecidable, even for linear hybrid systems as well as for stochastic hybrid systems. In this Chapter I described the idea of using a δ-complete decision procedure [30], which allows us to decide whether a bounded reachability question is unsatisfiable (*i.e.*, it is impossible to reach a goal state), or its relaxed version is satisfiable, which is characterised by some positive, user-defined over-approximation. I also show that that it is possible to solve bounded reachability in hybrid systems with the help of tools such as `dReach` [46] and `iSAT-ODE` [23], which incorporates SMT solvers that implement δ-decision procedures.

This Chapter also focuses on providing some background in the verification of stochastic parametric hybrid systems, *i.e.*, hybrid systems described in terms of parameters as random and nondeterministic initial conditions.

In this Chapter, parametric nondeterministic hybrid systems are formally characterised and the bounded reachability property is defined in terms of bounded $\mathcal{L}_{\mathbb{R}}$-sentences (see Definition 2.4). Further, some theoretical materials are provided to explain the mathematical methods used, including a brief introduction to MC and QMC methods, and in conclusion an overview of RQMC methods. Error estimation of these methods, including the theoretical usage of the Koksma-Hlawka inequality and Qint method, is also considered.

Finally, a detailed introduction to GP models is presented, aiming at understanding the essence of the stochastic process and how it is used to characterise a distribution over functions. GP regression and classification are described in a simple way. Some arguments for the advantages of practical usage of GPEP method are given, and in conclusion, the current trends in GP research are considered.

Chapter 3

Estimation Techniques for Probabilistic Reachability and Gaussian Processes

3.1 Introduction

In this Chapter the bounded reachability probability is defined in terms of expectation of Bernoulli random variables, and algorithms for computing confidence intervals for the bounded reachability probability are described. MC and QMC methods are evaluated in terms of bounded probability reachability and are illustrated with an example of their application. CI estimation problems are considered in the case when the actual probability is near the borders of the [0,1] interval. In this Chapter I also introduce a new method of error approximation, based on the classical CLT approach. The aim of the above mentioned algorithm is to solve the problem of poor confidence interval actual coverage probability estimation near the boundaries (0 and 1). In this method a sequential estimation of the sample standard deviation is used and CI at every new sample is reestimated. In Chapter 5 I empirically demonstrate the advantages of this approach.

I also consider the problem of computing probabilistic reachability for stochastic models with parametric uncertainty (nondeterministic parameters). The use of the EP algorithm, which performs model checking for probability functions

with non-Gaussian likelihood, can be enabled only on the basis of the fact that reachability probability function should be a smooth function of the model parameters. I give a proof that the reachability probability function of a SnPHS is, under mild conditions, a smooth function of the uncertain parameters. This crucial fact allows us to show that GP approximation can be successfully applied in SnPHSs. This is accomplished by exploiting the nature of the function's smoothness, which allows modelling explicitly correlations through previous distribution over a smooth function space (a GP) and place observations at individual parameter values to create a function's own analytical approximation.

In this Chapter, I introduce a novel statistical technique for computing bounded reachability probability in SnPHSs. The presented algorithm grants statistically rigorous confidence intervals by combining the formal approach, based on formal reasoning which provides absolute numerical guarantees, and the GP regression method, which provides statistical guarantees. This algorithm help to reduce the computational cost with respect to the formal approach. In particular, during the first phase of the proposed method the formal algorithm returns probability enclosures for the points from the parameter's domain of the bounded reachability probability function. GP then utilises these probability enclosures data with the aim to construct two regression approximations for upper and lower bound of the probability enclosures.

Finally, I outline the theoretical basis of the formal and GP regression combination technique and provide an informal evaluation of its computational characteristics including calculation precision and computational complexity. I also show that it is feasible to discover an effective trade-off between output precision and the computational complexity. The latter fact gives us hope that GP process regression in combination with some formal approach method can be an effective solution not only for rare event cases but also in general. This point will be more precisely covered in Section 5.7. The application of the developed technique is also discussed in more detail in Section 5.7.

3.2 Monte Carlo and Quasi-Monte Carlo Methods Validation

A statistical approach to probabilistic reachability is essential because it scales with system size much better than other methods and can still provide correctness guarantees. For instance, statistical model checking [94] can be quicker than probabilistic model checking, which is based on exhaustive state-space search [93].

Monte Carlo probability estimation methods assume that the random variable representing the real system behaviour can be sampled. However, this is impossible in practice because reachability is undecidable. The methods described in this section explicitly take into account undecidability and numerical accuracy.

In this section, the bounded reachability probability is defined in terms of Bernoulli random variables and an algorithm for computing confidence intervals for the bounded reachability probability in SPHSs is discussed. Then I show a method for computing an approximation of the maximum/minimum bounded reachability probability. I also present MC and QMC methods in terms of bounded probability reachability and provide an application example.

3.2.1 Computing Confidence Intervals for Bounded Reachability

In order to define bounded reachability in SPHS $(H, \mathbb{P})$ it is possible to use a Bernoulli random variable of the following form [73]:

$$X(\mathbf{p}_N, \mathbf{p}_R) = \begin{cases} 1 & \text{if system } H \text{ reaches the goal in } l \text{ steps for } \mathbf{p}_N \in P_N, \mathbf{p}_R \in P_R \\ 0 & \text{otherwise,} \end{cases} \tag{3.1}$$

where P_N and P_R are the nondeterministic and random parameter domain, respectively.

It implies that the expected value of X is equal to the bounded reachability probability, *i.e.*, $\mathbf{Pr}(\mathbf{p}_N) = \mathbb{E}[X(\mathbf{p}_N)]$, where $\mathbf{p}_N$ is a nondeterministic parameter and $\mathbf{p}_R$ is a random parameter. Unfortunately, as it was already discussed in

Subsection 2.2.1 it is impossible to directly evaluate samples of variable X because of the undecidability of bounded reachability in hybrid systems.

Instead, two Bernoulli random variables X_{sat} and X_{usat} whose values can be computed can be used for bounding the range of X [72]. For any given $\delta > 0$, $\mathbf{p}_N \in P_N$ and $\mathbf{p}_R \in P_R$, X_{sat} and X_{usat} are defined as the following:

$$X_{sat}(\mathbf{p}_N, \mathbf{p}_R, \delta) = \begin{cases} 1 & \text{if } \mathbf{evaluate}(H, l, \{\mathbf{p}_N, \mathbf{p}_R\}, \delta) = \mathbf{sat}, \\ 0 & \text{otherwise}, \end{cases}$$

$$X_{usat}(\mathbf{p}_N, \mathbf{p}_R, \delta) = \begin{cases} 0 & \text{if } \mathbf{evaluate}(H, l, \{\mathbf{p}_N, \mathbf{p}_R\}, \delta) = \mathbf{unsat}, \\ 1 & \text{otherwise}. \end{cases}$$

The $\mathbf{evaluate}(H, l, p, \delta)$ procedure here outputs *sat* if H reaches a goal in l steps for p; *unsat* if H does not reach the goal and *undet* if none of the above could be determined (see Section 2.2). Therefore, if it is possible to conclude that H reaches the goal state for the given $\mathbf{p}_N$ and $\mathbf{p}_R$ then $X_{sat}(\mathbf{p}_N, \mathbf{p}_R, \delta)$ equals 1. Otherwise if it can be decided that H does *not* reach the goal state for these $\mathbf{p}_N$ and $\mathbf{p}_R$ then $X_{usat}(\mathbf{p}_N, \mathbf{p}_R, \delta)$ equals 0. When no decision can be made (because of the precision δ being used or the nature of the reachability question), $X_{sat}(\mathbf{p}_N, \mathbf{p}_R, \delta)$ and $X_{usat}(\mathbf{p}_N, \mathbf{p}_R, \delta)$ take 0 and 1, respectively. Thus, the following holds for any $\delta > 0$ (see Section 2.3):

$$X_{sat}(\mathbf{p}_N, \mathbf{p}_R, \delta) \leq X(\mathbf{p}_N, \mathbf{p}_R) \leq X_{usat}(\mathbf{p}_N, \mathbf{p}_R, \delta) \ . \tag{3.2}$$

In our case, we deal with independent random parameters, so that for n samples $\mathbf{p}_i \in P_R$ the random variables X_{sat} and X_{usat} are independent and identically distributed, and we can define the next estimators:

$$\hat{S}_n = \frac{\Sigma_{i=1}^{n} X_{sat}(\mathbf{p}_N, \mathbf{p}_i, \delta)}{n}, \qquad \hat{U}_n = \frac{\Sigma_{i=1}^{n} X_{usat}(\mathbf{p}_N, \mathbf{p}_i, \delta)}{n}. \tag{3.3}$$

The presented Equation (3.3) can be successfully used for producing confidence intervals for the bounded reachability probability.

3.2.2 MC and QMC Bounded Reachability

Let us consider a hybrid system H with random parameters only. I consider two Bernoulli random variables: X_{sat}, which takes 1 if we can *correctly decide* that

system H reaches the goal in k steps for p and 0 otherwise; X_{usat}, which takes 0 if we can *correctly* decide that system H does not reach the goal and 1 otherwise [74]. Therefore:

$$X_{sat} \leqslant X \leqslant X_{usat}$$

and thus:

$$\mathbb{E}[X_{sat}] \leqslant \mathbb{E}[X] \leqslant \mathbb{E}[X_{usat}] \ .$$

By the definition of expectation, we get:

$$\int_{P_R} X_{sat}(p)dp \leqslant \int_{P_R} X(p)dp \leqslant \int_{P_R} X_{usat}(p)dp \ . \tag{3.4}$$

We take the sample approximation of (3.4) and obtain

$$\frac{1}{N}\sum_{i=1}^{N} X_{sat}(\mathbf{p}_i) \leqslant \frac{1}{N}\sum_{i=1}^{N} X(\mathbf{p}_i) \leqslant \frac{1}{N}\sum_{i=1}^{N} X_{usat}(\mathbf{p}_i),$$

where the $\mathbf{p}_i$'s can be sampled by using low-discrepancy sequences for QMC methods or pseudo-random sequences for MC methods.

3.3 Modified CLT Method

In this section, I investigate whether it is possible to apply statistical methods for efficient estimation of the approximation error near the probability bounds (0 or 1). I consider CLT methods as a possible solution due to their theoretical benefits in convergence and number of samples. In this section, I also discuss CI estimation problems when the actual probability is near the borders of the [0,1] interval and also introduce a novel approach for error approximation based on the classical CLT method.

3.3.1 Approximation of CI for Probabilities Near the Bounds

In situations when data are few, classical approaches to estimating the rate of occurrence of rare events show very poor performance. There have been some alternative empirical-based approaches suggested. They are based on median estimators or non-informative prior distributions. Although these alternatives have an advantage in point estimates of zero, they can be generally conservative. One

approach is offered by Empirical Bayes procedures, which is performed through pooling data across different hazards to support the stronger statistical inference [87].

There is a difficulty though that many problems normally face when investigating the probability of rare events. In that situation standard Monte Carlo needs on average a very large number of samples to witness the rare event once, for instance, 10^9 independent simulations on average for an event of probability 10^{-9}, a typical target. Consequently, some specific techniques have to be applied. Importance Sampling and Multilevel Splitting (sometimes called Subset Simulation) techniques are examples of the most prominent ones, which nevertheless are sometimes too complicated and involve high computational costs [35, 36].

We can set upper confidence bounds on event risks when no events are observed. It may be applicable when we try to determine possible risks for serious adverse events. A simple rule defined as the "rule of threes" has been widely used before. It states that if no events are observed in a group, then the upper confidence interval limit for the number of events is three, and for the risk (in a sample of size N) is $3/N$ [58]. The usage of this rule has not directly been suggested or estimated for systematic reviews.

One of the classical methods to estimate the rate of occurrence of events is calculating the ratio of the number of events that have occurred to the length of the period of observation. Theoretically, this process has the necessary asymptotic properties, as it is an unbiased estimate of the rate of occurrence of such incidents and the minimum variance unbiased estimator. In the condition when data are few, though, its performance is quite poor.

The methodology I present is based on the natural overall rate of the obtained data. It estimates appropriate adjustments from the pooled rate for each individual event. This methodology is based on the standard CLT method and aims to solve the problem of poor confidence interval actual coverage probability near the boundaries (0 and 1) [16, 59].

3.3.2 Modified Central Limit Theorem

First, I consider the case when the samples x_i are extracted from the normal distribution $N(\mu, \sigma^2)$ with unknown parameter μ and known σ^2, where μ is the mean or expectation of the distribution and σ^2 is the variance. Here, μ can be approximated by the sample mean: $\mu \approx \tilde{X}$, where $\tilde{X} = \frac{1}{n}\sum_{i=1}^{n} x_i$. To clarify this approximation, we need to construct a CI covering the parameter μ with a given confidence probability:

$$CI_{CLT} = \left(\tilde{X} - C_a\frac{\sigma}{\sqrt{n}}; \tilde{X} + C_a\frac{\sigma}{\sqrt{n}}\right), \tag{3.5}$$

where $C_a = Quant(1-\frac{a}{2})$ is the inverse cumulative distribution function of a normal random variable with mean 0 and standard deviation 1; parameter a defines the confidence level at $1-a$. If the variance σ^2 is unknown, we can use the same CI by replacing σ with its sample standard deviation $s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \tilde{X})^2}$. This method is widely used for estimating the distribution of the error regarding the binomially-distributed $\hat{p} = n_s/n$, where, n_s is the number of "successes" and n is the total number of Bernoulli "trials" which is the setting we are interested in this thesis. Many related works [14, 16, 19] note that CI_{CLT} approximation can be poor when applied to Bernoulli trials with $\hat{p}$ close to 0 or 1. It can be easily seen that when $\hat{p}$ is 0 (or 1), the standard confidence interval $C_a\sqrt{\frac{1}{n}\hat{p}(1-\hat{p})}$ cannot be constructed.

In order to solve this problem, I introduce a new method for variance estimation, which uses a sequential estimation of the sample standard deviation and calculates CI_{CLT} (3.5) at every new sample. My solution simply approximates the sample standard deviation with $\frac{1}{n^2}$ (where n is the total number of samples) at the initial stages of the computation if $\hat{p}$ is equal to 0 (or 1, when all returned results of a hybrid system model evaluation are the same) and propagates it through the computation until the necessary number of samples (differ from 0 or 1) to construct the interval are obtained. I will show the advantages of this approach in Subsection 5.3.1.

The same modification was applied to calculation of the Qint CI (see results in Subsection 5.3.2). Initially the Qint method (see Subsection 2.3.5) is also not able to compute CI in the cases where the output of Bernoulli trials is equal to 0 or 1 using standard CLT, that is why this modification is essential for calculation

results of the probability near the bounds for the Qint method, which has a strong potential to show better results among statistical methods.

3.4 Probabilistic Reachability Analysis

In this section, I consider the problem of computing the probability of a formula for stochastic models with parametric uncertainty, *i.e.* SnPHS. I show that the reachability probability function is under reasonable conditions a smooth function of the model parameters, so it can be successfully approximated by GP. This also enables the use of the EP algorithm, which can open GP approach with non-Gaussian likelihood. It uses observations of truth values of the formula over individual runs of the model at isolated parameter values to provide CIs over values of the model the whole parameters' range.

This problem addresses a very important question of whether the reachability probability function can be considered for Gaussian processes (GP) approximation. I solve it by exploiting the nature of smoothness of the function: by modelling explicitly correlations through a prior distribution over a space of smooth functions (a Gaussian Process), so it is possible to condition on observations at individual parameter values to construct an analytical approximation of the function itself. I also show that the reachability probability function can be approximated arbitrarily well by Quasi-Monte Carlo (QMC) sampling from a GP, and then the GP regression method can be successfully applied to it to obtain an analytical approximation.

Finally, in this section I discuss the smoothed model checking approach and the likelihood model I use. I provide a high-level description of the method and introduce an EP-based approximation algorithm.

3.4.1 Reachability Probability Function Smoothness

As it was stated above my aim is to show that the reachability probability of an SnPHS (see Definition 2.2 in Subsection 2.2.1) is a smooth function of the nondeterministic parameters, and thus GP can be used to approximate the function itself very efficiently over the entire domain of nondeterministic parameters.

Theorem 3.1. *Let H be an SnPHS in which the random parameter densities are smooth,* i.e., *$f_i(x, \mathbf{p}) \in C^\infty(R \times P)$ for all $1 \leqslant i \leqslant r$. Then, the reachability probability function of H is a smooth function of the uncertain parameters,* i.e., *$\mathbf{Pr}(\mathbf{p}) \in C^\infty(P)$.*

Proof. We recall that $P \subset \mathbb{R}^k$, $R \subset \mathbb{R}^r$ and G are the uncertain parameters space, random parameters domain and goal set of the SnPHS H, respectively. We need to show that the function of Definition 2.8:

$$\mathbf{Pr}(\mathbf{p}) = \int_G d\mu(\mathbf{p}) = \int_R I_G \, d\mu(\mathbf{p})$$

admits derivatives of any order, where I_G is the indicator function over G. Since the random parameters are independent, we can rewrite the above as:

$$\mathbf{Pr}(\mathbf{p}) = \int_R I_G(\mathbf{x}) F(\mathbf{x}, \mathbf{p}) \, d\mathbf{x}$$

where $\mathbf{x} \in \mathbb{R}^r$ and $F(\mathbf{x}) = \Pi_{i=1}^r f_i(x_i, \mathbf{p})$ is the product measure obtained from the random parameters' densities. For clarity of presentation, I assume that $\mathbf{p}$ is a single uncertain parameter, *i.e.*, $k = 1$. The extension to multiple uncertain parameters ($k > 1$) is easily obtained by considering each coordinate.

Since $\int_R I_G(\mathbf{x}) \, d\mathbf{x} < \infty$ (recall R is a bounded set), by Lebesgue's criterion [96, Theorem 1, Sect. 11.1] the function I_G is continuous almost everywhere on R. Let $D \subset R$ be the set of points at which I_G is discontinuous. Since D has measure zero we have that

$$\mathbf{Pr}(\mathbf{p}) = \int_{R \setminus D} I_G(\mathbf{x}) F(\mathbf{x}, \mathbf{p}) \, d\mathbf{x} \tag{3.6}$$

and by the hypothesis on the densities f_i's the function $I_G(\mathbf{x})F(\mathbf{x}, \mathbf{p})$ is then continuous over $R \setminus D \times P$ and has continuous partial derivative with respect to $\mathbf{p}$. Therefore, by Equation (3.6) and Leibniz's rule [96, Proposition 2, Sect. 17.5.1], we have that

$$\begin{aligned} \frac{d\mathbf{Pr}(\mathbf{p})}{d\mathbf{p}} &= \frac{d \int_{R \setminus D} I_G(\mathbf{x}) F(\mathbf{x}, \mathbf{p}) \, d\mathbf{x}}{d\mathbf{p}} = \int_{R \setminus D} \frac{\partial I_G(\mathbf{x}) F(\mathbf{x}, \mathbf{p})}{\partial \mathbf{p}} d\mathbf{x} \\ &= \int_{R \setminus D} I_G(\mathbf{x}) \frac{\partial F(\mathbf{x}, \mathbf{p})}{\partial \mathbf{p}} d\mathbf{x} \end{aligned}$$

and $\frac{d\mathbf{Pr}(\mathbf{p})}{d\mathbf{p}}$ is a continuous function over P by [96, Proposition 1, Sect. 17.5.1]. The proof now simply proceeds by induction on the order of the derivative, with

the smoothness hypothesis. □

In our case studies we shall use normally-distributed random parameters: it is easy to show that Gaussian densities satisfy the hypothesis of Theorem 3.1.

It is also important to note that the GP approach can be applied to any SnPHSs where the goal set does not depend on nondeterminstic parameters, which means that the presented above theoretical result can be successfully applied to a broad number of studies in the field.

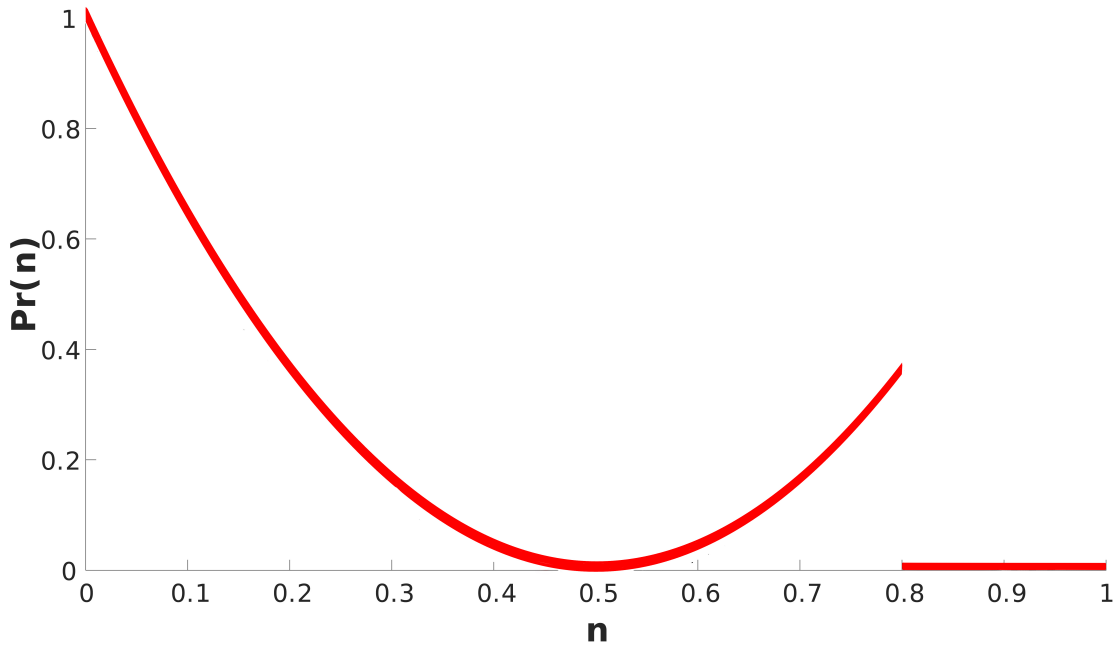


Figure 3.1: Non-smooth reachability probability function of "Bad" SnPHS model with the goal set which depends on nondeterminstic parameter n.

An example where it is impossible to apply GP can be shown on the example of the simple model used in this thesis and called "Bad" (see Section 5.2). The predicate $(x(0) = r) \wedge (n \in [0, 1])$, where r is uniformly distributed over [0,1], and n is a continuous nondeterministic parameter on $[0, 1]$ is used to define the original state of the system. Now we slightly change the goal so that: $(x \leq 2(n - 0.5)^2 + 0.5) \wedge (x \geq -2(n - 0.5)^2 + 0.5) \wedge (n \leq 0.8)$. The goal now directly depends on the nondeterministic parameter n. Therefore, the probability function will be equal to $\mathbf{Pr}(n) = 4n^2 - 4n + 1$ up to 0.8 point from n's parameter domain,

which reaches its minimum value of 0 at $n = 0.5$ and then will have a jump to 0 flat line values. We can see in Figure 3.1 that the graph of the reachability probability function obtained analytically represents a non-smooth function.

3.4.2 GP Approximation of a Smooth Probability Function

In order to successfully apply GP, we need to determine an initial prior distribution, which describes our probability reachability function in the most effective way (see Section 2.5). Then it is necessary to define the functional form of the likelihood. This form influences the dependence of the probability of the observed satisfaction values at individual parameters on the (unknown) true value of the reachability probability at that point. The final step is computing an approximation of the posterior distribution over functions, given the observations via Bayes's theorem. We receive the required estimate and confidence interval by evaluating the statistics of the induced posterior distribution on the function values at each point. The latter can be done by using EP methods, described in Subsection 2.5.5.

Given a SnPHS $\mathcal{S}_\theta$ depending on a vector of parameters $\theta \in \mathcal{D}$, our goal is to find a statistical estimate of the reachability probability as a function of θ, *i.e.* of the function:

$$f(\theta) = P(\vartheta | S_\theta) = \mathbf{Pr}(\theta).$$

Evaluating reachability over a SnPHS model returns Boolean observations - 1/0 according to every new sample. The probability of the observations being 1 is a Bernoulli distribution with parameter $f(\theta)$. Per each ϑ in the training set, we generate n observations, which should be drawn from a Binomial random variable $Binomial(n, f(\theta))$ (see Algorithm 1 in Section A.1 of Appendix). Therefore per every nondeterministic point we have an output in form of the observations row (*e.g.* $[0, 1, 1, 0, 0, 1]$). An approximation of $f(\theta)$ can be computed directly from such binomial variable observations. For the convenience of the algorithm, we take the probability value for each point equals to the mean of the observations row, so that if we have $[0, 1, 1, 0, 0, 1]$ observations the probability for the chosen point will be equal to 0.5. In statistical model checking the accuracy of such approximation would only be guaranteed in the limit of $n \to \infty$.

There is no need to conduct this intermediate assessment in a GP context: we can directly use the binomial observation model in the Bayes theorem. Thus, it is possible to use the exact statistical model of the process, converging in the limit of a large number of observations to the true function. The advantage of this approach follows directly from the GP definition (see Subsection 2.5.1): GPs provide a full range approximation with few samples per each input point $\theta_1, ..., \theta_k$, which is very efficient from a computational point of view.

It has been already noted in Subsection 2.5.3 that the observation of 1/0 labels per every input point makes this process similar to a classification problem. However, in classical GP classification, we deal only with two classes. The classification problem can be extended to consider multiple labels (see Subsection 2.5.4), but the crucial difference is that in a multiple annotator classification all the probability classes should be initially known, which is impossible in our case. In our situation, the reachability probability function observations are produced at isolated parameter values through (Boolean) reachability evaluations of a SnPHS model. Therefore, the Gaussian likelihood cannot be applied directly, meaning that a closed-form solution to the inference problem can not be found. At the same time, our observations have infinitely many classes of probability values (see Subsection 2.5.3) so that multiple annotators techniques are not also useful.

A possible solution is the use of a modified version of the EP algorithm (see Algorithm 2 in Section A.1 of Appendix) as proposed by Minka [53]. However, to map probabilities to the full real line, we need to introduce the inverse probit transformation [61]:

$$\Psi(w) = \zeta \Leftrightarrow w = \int_{\infty}^{\zeta} \mathcal{N}(0, 1),$$

where $\mathcal{N}(0, 1)$ is the standard Gaussian density (with mean zero and variance 1), $\forall w \in [0, 1]$ and $\zeta \in \mathbb{R}$. Note that the function $\zeta(\theta) = \Psi(f(\theta)$ is a smooth real valued function of the model parameters by definition, which allows us to use GP methods.

In general, at each training point (parameter value), our data would consist of N binary satisfaction evaluations. Binary evaluation represents independent

draws from the same Bernoulli distribution with $f(\theta)$ likelihood of success at each parameter value. The overall joint probability of the probability function $f(\theta)$ and of the observations $\mathcal{O}$ can be presented as:

$$p(\mathcal{O}, f(\theta)) = GP(\Psi(f(\theta))) \prod_{i=1}^{C} \prod_{j=1}^{N} Bernoulli(\mathcal{O}_{i,j}|f(\theta_j).$$

Note that computing the target function at new parameter value θ (*e.g.* $f(\theta)$) can be solved through computing the posterior distribution of the probability reachability function at θ^* (see Algorithm 3 in Section A.1 of Appendix). The EP procedure of computing the posterior distribution is described in detail in Subsection 2.5.5.

3.5 Formal and GP Combination Approach

In this section, I present a new distinct technique: a promising combination of the formal approach, based on formal reasoning which provides absolute numerical guarantees, and the GP regression method, which provides statistical guarantees only.

The formal algorithm computes probability enclosures for the range of the bounded reachability probability function **Pr** defined in Subsection 2.2.4. In other words the formal algorithm returns a probability interval, with the absolute guarantee that the reachability probability function is within its bounds. GPs then utilise these probability enclosures with the aim to construct two regression approximations for the upper and lower limit of the enclosures.

In this Section I discuss computing probability enclosures aspects that directly affect the formation of initial data for GP training. Finally, I demonstrate the theoretical basis of the formal and GP regression combination procedure and evaluate its computational characteristics.

3.5.1 Computing Probability Enclosures

The formal approach allows us to compute probability enclosures for the range of the bounded reachability probability function **Pr**. The algorithm takes as input:

a SPHS, a reachability depth $l \in \mathbb{N}$, a precision $\epsilon > 0$ for the size of probability enclosures, a constant $\kappa \in (0, \epsilon)$ for bounding the domain of continuous random parameters, a parameter η controlling the precision of procedure **evaluate** and a precision vector ρ for nondeterministic parameter boxes. The output is a probability enclosures list [72]. This list represents a finite set of disjoint nondeterministic parameter boxes which fully cover the parameters' domain P_N. The technique behind the algorithm is first to partition the domain with boxes of non-deterministic parameters and then to obtain a probability enclosure for each such box. It can be done by sequentially adjusting under-approximations and over-approximations of the definite integral representing the **Pr** function over the random parameter space for every corresponding nondeterministic box. The proof of correctness of the described algorithm is given in [72]. The algorithm for building probability enclosures is given in Algorithm 4 in Section A.1 of Appendix.

If an SPHS does not feature nondeterministic parameters, only one enclosure will be returned, and its size will be bounded above by the ϵ input. If nondeterministic parameters are present, the size of the smallest nondeterministic parameter box can be limited by the precision vector ρ, allowing termination of the algorithm in the most general case.

In general, if procedure **evaluate** returns **unsat** then there is no value in $B_N \times B_R$ for which the goal state is reachable, and the upper bound of the probability enclosure $[a, b]$ can be reduced. If **evaluate** returns **sat** then for every value in $B_N \times B_R$ it is possible to reach the goal and the lower bound of the probability enclosure $[a, b]$ can be increased.

In was proven in [72] that **evaluate** returns formally correct answers because they rely on the *unsat* answer from the δ-complete decision procedure.

An example of enclosures returned by the formal approach is presented in Figure 3.2. In this example, the size of each enclosure is $\epsilon = 5 \cdot 10^{-2}$, implying that the algorithm terminated because every nondeterministic parameter box reached the minimal size of ρ.

Summing up, the formal approach's output quality relies on three input arguments: the probability enclosure precision ϵ, the precision value ρ for nondeterministic parameter boxes and η – argument for controlling precision of the procedure **evaluate**. The first argument ϵ describes the primary goal — to de-

crease the sizes of probability enclosures to an arbitrarily small value.

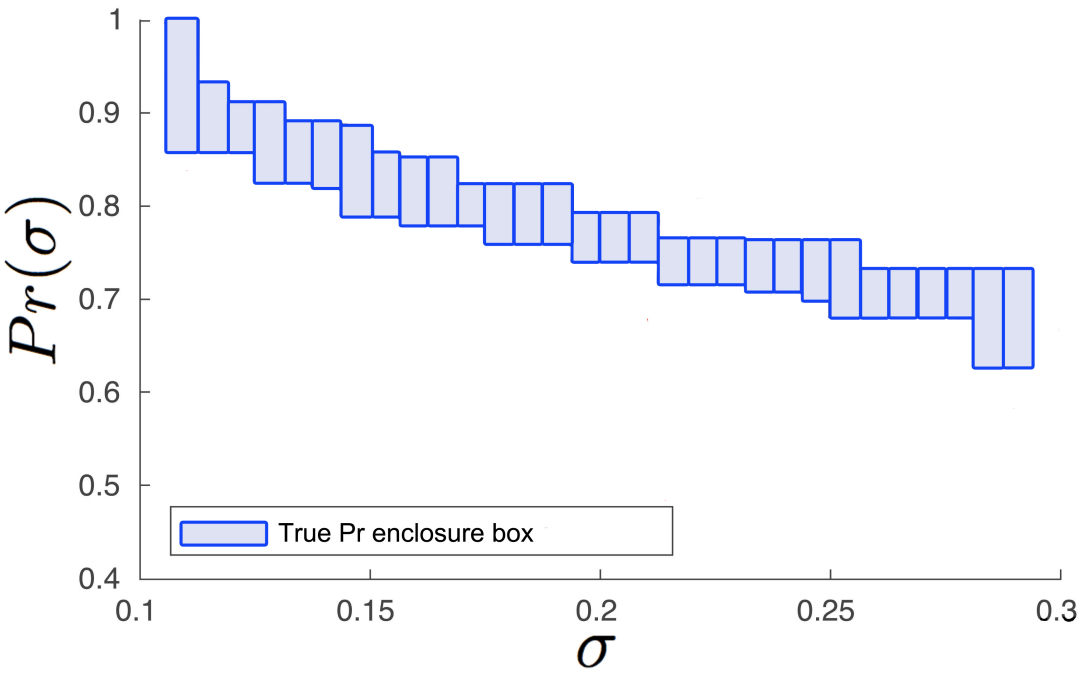


Figure 3.2: Formal enclosure boxes (computed with precision parameter $p = 5 \cdot 10^{-}2$) for stochastic model Collision, type advanced with one uncertain parameter σ.

3.5.2 Probability Enclosures GP Approximation

As it was mentioned above, the formal approach returns probability enclosure boxes (see *e.g.* Figure 3.2) when exploring all parameter's values in a certain range. It can be seen from the figure that the formal approach can return the same probability values for a certain region of the parameter space. However, for an efficient application of our novel combined approach, we do not need to have so much information and do not need to spend time on extra computation. Instead, we ask to compute probability enclosures for certain points only, chosen over the nondeterministic parameter's space. In Figure 3.3 we can see that the formal approach returns enclosures for the chosen 11 points of the parameter space σ. The number of points can vary in accordance with the approximation purposes. Such points can be sampled by a simple partition or more effectively

by using low-discrepancy sequences, *e.g.* QMC. The formal approach provides absolutely precise probability enclosures (see Subsection 3.5.1), which allows us to use their upper and lower points as absolutely precise bounds for function **Pr** (see Algorithm 4 in Section A.1 of Appendix).

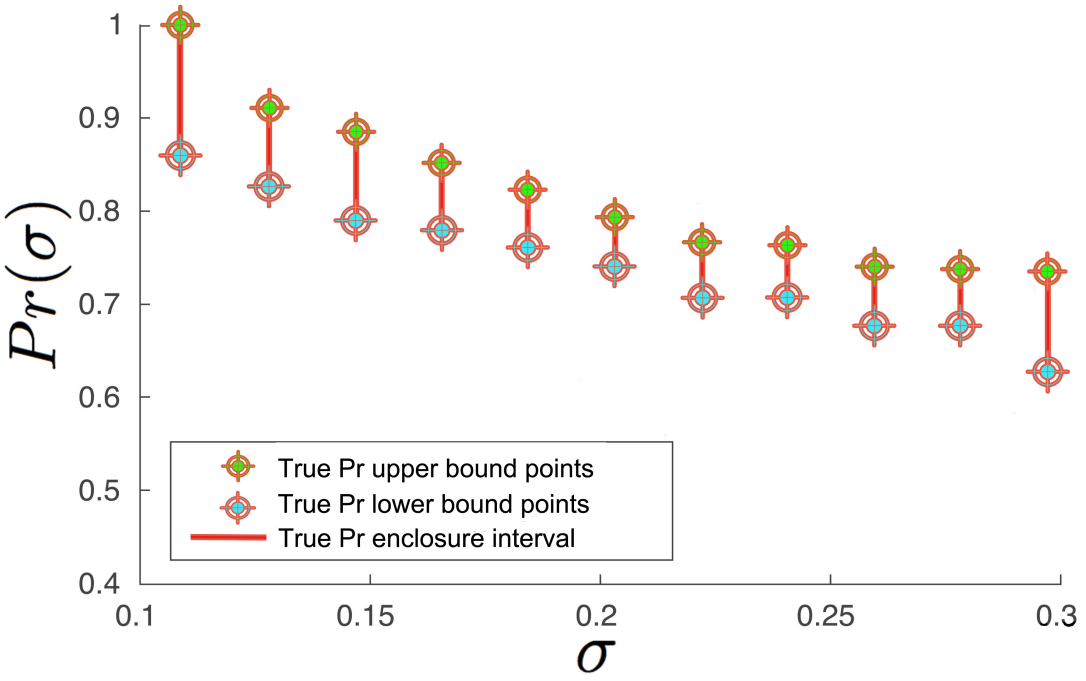


Figure 3.3: Upper and lower latent function construction for GP regression approximation via formal enclosure intervals (computed with precision parameter $\epsilon = 5 \cdot 10^{-2}$) for stochastic model Collision type advanced with one uncertain parameter σ.

GP regression (see Subsection 2.5.1) is computed on the basis of information obtained from the initial training dataset. The dataset is formed by the input parameter points' values and latent probability function values, the real shape of which we aim to approximate. In other words, we have a X set of nondereministic points σ and a Y set of probability values according to every nondeterministic point. However, for our combined approach we need to use two input datasets - one for the upper latent **Pr** function Y_1 and another for the lower latent **Pr** function Y_2 (see Figure 3.3), which means that for the combined approach we essentially have two probability values sets, Y_1 and Y_2. So that we need to run two GP regressions for Y_1 and Y_2. The GP regression output will then provide

estimated mean approximation and two CIs - lower and upper for the two function approximations. It can be easily shown that the final CI returned by our proposed approach can be formed by the upper CI of the upper latent **Pr** function (with Y_1 training set) and the lower CI of the lower latent **Pr** function (with Y_2 training set). The GP regression algorithm for the combined approach is presented in Algorithm 5 in Section A.1 of Appendix.

The major advantage of the developed method is based on the basic GP feature. The inference in the GP can define properties of the upper and lower **Pr** functions at a finite number of points, ignoring infinitely many points, with the same quality as if we would have taken them all into account [61]. It allows us to provide information about CIs, which include the true probability function over the whole parameter's domain. At the same time, the uncertainty of CIs in case of functions with no strong nonliniarities does not increase and remains almost the same for latent points over parameter's domain. It also brings a huge computational reduction. Unfortunately, the guarantees provided by this combined approach are statistical, while formal approach provides absolutely precise probability enclosures. At the same time in comparison with the GPEP approach the combined approach should be able to provide much more precise CIs size, which will be significantly smaller for the full parameter's domain. The latter fact will be closer considered in the Chapter 5.

Next, we discuss the precision and computational cost of the developed approach more precisely.

Precision. The combined approach is based on the formal approach, which can compute probability enclosures for the range of the bounded reachability probability function. In some cases, such enclosures can be arbitrarily tight, in particular, for systems featuring at least one continuous random parameter and no nondeterministic parameters. For example, in Figure 3.3 probability enclosures are presented, computed with approximation parameter $\epsilon = 5 \cdot 10^{-2}$.

The second part of the combined algorithm consists of the GP regression method solely. The precision of GP computation is usually presented by CIs, which can also be computed up to the requested precision level. It can be concluded intuitively that the final GP CI, constructed from the two GP regression

calculations (for upper and lower **Pr** function, see Figure 3.3) will be larger than the formal probability enclosure. However, the main GP advantage that covers this deficiency is the fact that even based on a small number of data points from nondeterministic parameter domain, GP is able to provide CIs over the whole parameter space. Moreover, in case of even data points distribution, which can be ensured by QMC sampling, GP regression provides CIs of almost the same size.

Computational Complexity. Computational complexity is very important for the application of the combined approach. The time for parameter space search grows exponentially with the number of system parameters. In the formal approach, the partitioning procedure [72] constructs 2^n boxes for each parameter box with n positive edges if the required precision ϵ is not reached. Thus, the algorithm's general computational complexity increases exponentially with the number of system parameters.

The GP regression method in turn does not require such computational overhead as formal method does. The "heaviest" element of GP regression (for testing and training) is the inversion of the covariance matrix (see Subsection 2.5.1), which should be performed for all pairs of parameter values. Having n points in the parameter's domain we obtain complexity $\mathrm{O}(n^3)$ [11]. In particular, the matrix inversion requires Cholesky factorization computation, which has $\mathrm{O}(n^3/6)$ complexity and further triangular systems solving which has $\mathrm{O}(n^2/2)$ complexity [61].

Even though we need to run two GP regression processes for our combined approach, the total computational time of the combined approach is mostly determined by the formal approach part. The GP advantage mentioned before allows us not to waste time again on parameter space exploration to estimate probability reachability function at new points like the formal approach does. It securely saves time for GP regression. In other words, it is possible to find a very efficient trade-off between the precision rate and computational complexity, which outdoes a single formal approach, covering the whole domain of nondeterministic parameters.

3.6 Summary

Chapter 3 provides the definition of bounded reachability probability in terms of Bernoulli random variables, and algorithms for computing confidence intervals for the bounded reachability probability. This Chapter addresses MC and QMC methods from the point of view of bounded probability reachability with an example of their usage and considered CI estimation problems when the actual probability is near the borders of the [0,1] interval.

Chapter 3 introduces a novel way of error approximation based on the classical CLT approach which fixes the problem of poor confidence interval actual coverage probability estimation near the boundaries (0 and 1) by sequential estimation of the sample standard deviation and re-estimation of CI at each new sample.

Chapter 3 also analyses the question of computing the probability reachability formula for stochastic models with parametric uncertainty. I show that the reachability probability function of a SnPHS is, under mild conditions, a smooth function of the uncertain parameters, which allows GP approximation to be successfully applied to SnPHSs.

A new statistical method for computing the bounded reachability probability in SnPHSs was also offered in this Chapter. I introduced an algorithm which gives statistically rigorous confidence intervals by combining the formal approach, based on formal reasoning providing absolute numerical guarantees, and the GP regression method, providing statistical guarantees. The main advantage of the presented algorithms over a simple formal approach is the reduction in the computational cost. Finally, Chapter 3 presented the theoretical grounds of the formal and GP regression combination technique and granted a theoretical estimation of its computational characteristics, including calculation precision and computational complexity. I also demonstrated that it is achievable to find a good trade-off between the precision of the results and the complexity of the computation.

Chapter 4

Implementation of Probability Reachability Evaluation Tools

4.1 Introduction

This Chapter presents the `Confidence Interval Estimation Tool` and the `GPEP Tool`, which I developed for computing bounded reachability probability in stochastic parametric hybrid systems. These tools enable the usage of the developed methods for CI estimation, including modified CLT method (see Subsection 3.3.2) and EP method (see Subsection 2.5.5). The `GPEP Tool` also provides a practical application of Theorem 3.1 presented in Subsection 3.4.2. The latter also allows the implementation of the novel statistical technique for computing the bounded reachability probability in SnPHSs, providing statistically rigorous confidence intervals by combining the formal approach, based on formal reasoning giving absolute numerical guarantees, and the GP regression method, giving statistical guarantees (see Section 3.5).

The two tools provide a C++ implementation (about 8,500 lines of code) of the algorithms introduced in the previous chapters. The implementation is developed within the ProbReach tool and uses also publicly available libraries, and is distributed under the GNU General Public License[1] (GPL).

This Chapter discusses the implementation details of the `Confidence Interval`

[1] http://www.gnu.org/licenses/gpl.html

`Estimation Tool` and the `GPEP Tool`, their architectures, and presents several usage scenarios.

4.2 ProbReach

The proposed techniques were developed on the basis of the ProbReach tool [73], which can be used to compute bounded reachability in SPHSs.

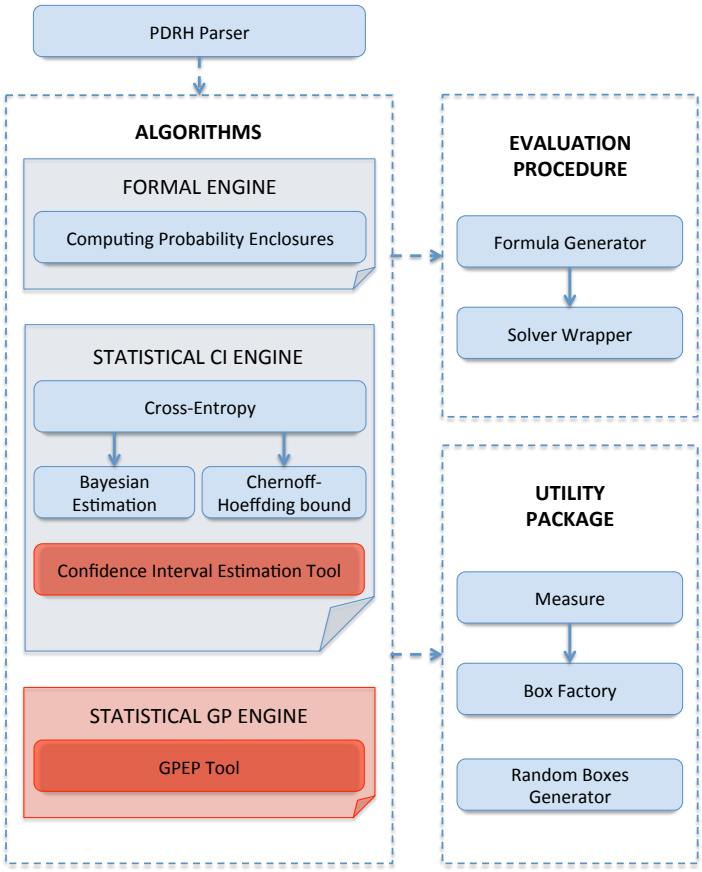


Figure 4.1: ProbReach Architecture with new developed tools.

ProbReach can use either dReal [31] or iSAT-ODE [22] for analysing (non-probabilistic) bounded reachability question.Essentially, both dReal and iSAT-ODE are implementations of δ-complete decision procedures for first-order logic

formulae over the reals that may include nonlinear functions and ordinary differential equations.

For CI estimation I implemented eight CI evaluation methods as part of the statistical engine of the Probreach tool. In order to evaluate GP algorithms for computing probabilistic reachability, I implemented a GPEP algorithm in ProbReach as part of the engine that performs GP classification and regression not only for uni-variate class labels but also for the multiple annotators case [65].

ProbReach consists of several components (as shown in Figure 4.1): *PDRH Parser*, *Evaluation Procedure*, *Utility Package* and *Algorithms*. In this thesis I only describe my owm contribution in detail, namely the developed CI estimation and GPEP algorithms only. More information about the PDRH Parser, Evaluation Procedure and Utility Package can be found in [72].

4.2.1 Confidence Interval Estimation Tool Input

As part of ProbReach, `Confidence Interval Estimation Tool` uses the **P**robabilistic **D**elta-**R**eac**H**ability (PDRH) format for SPHSs and SnPHSs encoding. PDRH extends **D**elta-**R**eac**H**ability (DRH) format utilised by dReach [46] with random parameters [72]. Figure 4.2 shows the PDRH encoding of Deceleration model (see Section 5.2 for the full model description). The full description of the PDRH format can be found in the ProbReach documentation[1].

```
#define v_max 55.56 // [200 km/h] maximum speed of a car (m/s)
#define v_100 27.78 // m/s
#define drag 3.028e-4 // some average value (1/m)
#define alpha 0.05776 //acceleration coefficient
#define t_react 1.2 //driver reaction time
[0,1000] s; // distance m
[0,v_100] v; // velocity m/s (16.67 m/s = 60 km/h)
[0,30] tau;  [0,30] time; // time in seconds
#define a_d 4.0; //deceleration parameter
dist_normal(4,0.1) beta; //random parameter
{ mode 1;      //accelerating
```

[1] https://github.com/mariiavasileva/EPPR-models

```
  flow:
    d/dt[s]= v;
    d/dt[v]= alpha*exp(-alpha*tau+beta)-drag*v*v;
    d/dt[tau]= 1.0;
    d/dt[a_d]=0.0;
jump: (v >= v_100)==>@2(and(s'=s)(v'=v)(tau'=0)(beta'=beta)(a_d'=a_d)); }
{ mode 2;       // reacting
  flow:
    d/dt[s]= v;
    d/dt[v]= -drag*v*v;
    d/dt[tau]= 1.0;
    d/dt[a_d]=0.0;
jump: (tau = t_react)==>@3(and(s'=s)(v'=v)(tau'=0)(beta'=beta)(a_d'=a_d)); }
{ mode 3;       // braking
  flow:
    d/dt[s]= v;
    d/dt[v]= -a_d;
    d/dt[v]= -a_d-drag*v*v;
    d/dt[tau]= 1.0;
    d/dt[a_d]=0.0;
jump: (v <= 0)==>@4(and(s'=s)(v'=v)(tau'=0)(beta'=beta)(a_d'=a_d)); }
{ mode 4;      // stopped
  flow:
    d/dt[s]= v;
    d/dt[v]= 0.0;
    d/dt[tau]= 1.0;
    d/dt[a_d]=0.0;
jump:  }
init: @1(and (s = 0) (v = 0) (tau = 0));
goal: @4(s>=400);
```

Figure 4.2: Deceleration model encoded in PDRH format for CI estimation.

4.2.2 GPEP Tool Input

The `GPEP Tool` also uses the PDRH input format for SPHSs and SnPHSs encoding. Figure 4.5 shows the PDRH encoding of Deceleration model (see Section 5.2 for the full model description). `GPEP Tool` also works with data source input in the library format [1]. Figure 4.3 shows a generated in hdf5 format data set, which can be used multiple times and immediately without the model evaluation procedure usage. Other input option in `GPEP Tool` is Inverse Covariance Matrix data, obtained after testing GP (see Figure 4.4). This input can be applied very effectively in case when we need to test new input points on the basis of the existing training data, so that we do not need to run training process again.

N%	x_points	y_points
0	{0.1046}	{0.09}
1	{0.1523}	{0.19}
2	{0.0574}	{0.00}
3	{0.0763}	{0.03}
4	{0.1742}	{0.21}
5	{0.1247}	{0.14}
6	{0.0261}	{0.00}
7	{0.0372}	{0.00}
8	{0.1376}	{0.16}
9	{0.1876}	{0.21}
10	{0.0874}	{0.07}
11	{0.0622}	{0.01}
12	{0.1636}	{0.20}
13	{0.1126}	{0.12}
14	{0.0111}	{0.00}
15	{0.0183}	{0.00}
16	{0.1187}	{0.13}
17	{0.1686}	{0.20}
18	{0.0689}	{0.02}
19	{0.0931}	{0.08}

Figure 4.3: Deceleration model data set obtained after the evaluation decision procedure. These data constructs the initial observations data set for 20 points and consists of parameter values - x_points and probability values - y_points.

[1] https://www.hdfgroup.org/solutions/hdf5/

	0	1	...20
0	1.9834045063027033	-0.14381230836290995	...
1	-0.14381230836291528	4.042066708093642	...
2	-0.07524409250141711	-0.007637212981806582	...
3	-0.1263020977650748	-0.06650109950814154	...
4	-0.03873445356676367	-0.5796685531537467	...
5	-0.1901100740238572	-0.3302802433652063	...
6	-0.048819614367838	0.01706682205382522	...
7	-0.06075278940399859	0.0070247384547653365	...
8	-0.17348822470222375	-0.4041542481149479	...
9	0.029638162747448356	-0.6076172799222995	...
10	-0.1618016145118691	-0.12228089159321176	...
11	-0.09783043958836685	-0.030544908436681395	...
12	-0.09719684085543936	-0.534646976952196	...
13	-0.1931761749182235	-0.25658385686623425	...
14	-0.039317722224407214	0.024106851896966555	...
15	-0.0437855017893501	0.020873074731551124	...
16	-0.19324312380769829	-0.2931325162129056	...
17	-0.06910837005851449	-0.5567953117257585	...
18	-0.11224704870658812	-0.04694991943663979	...
19	-0.17197414960442536	-0.15141534441850854	...

Figure 4.4: Deceleration model Inverse Covariance Matrix data obtained after training procedure. Please see more details of Covariance Matrix estimation process in Section 2.5.

```
#define v_max 55.56 // [200 km/h] maximum speed of a car (m/s)
#define v_100 27.78 // m/s
#define drag 3.028e-4 // some average value (1/m)
#define alpha 0.05776 //acceleration coefficient
#define t_react 1.2 //driver reaction time
[0,1000] s; // distance m
[0,v_100] v; // velocity m/s (16.67 m/s = 60 km/h)
[0,30] tau; [0,30] time; // time in seconds
#define a_d 4.0 //deceleration parameter

#define mu 4.0; //1st nondeterministic parameter mu is fixed
```

```
[0,0.2]sigma; //2nd nondeterministic parameter sigma
dist_normal(mu,sigma) beta; //random parameter

{ mode 1;   // accelerating
  flow:
    d/dt[s]= v;
    d/dt[v]= alpha*exp(-alpha*tau+beta)-drag*v*v;
    d/dt[tau]= 1.0;
jump: (and (v >= v_100))==>@2(and(s'=s)(v'=v)(tau'=0)); }
{ mode 2;   // reacting
  flow:
    d/dt[s]= v;
    d/dt[v]= -drag*v*v;
    d/dt[tau]= 1.0;
jump: (and (tau = t_react))==>@3(and(s'=s)(v'=v)(tau'=0)); }
{ mode 3;  // braking
  flow:
    d/dt[s]= v;
    d/dt[v]= -a_d-drag*v*v;
    d/dt[tau]= 1.0;
jump: (and (v <= 0))==>@4(and(s'=s)(v'=v)(tau'=0)); }
{ mode 4;   // stopped
  flow:
    d/dt[s]= v;
    d/dt[v]= 0.0;
    d/dt[tau]= 1.0;
jump: }
init: @1(and (s = 0) (v = 0) (tau = 0));
goal: @4(s>=400);
```

Figure 4.5: Deceleration model encoded in PDRH format for GP approximation.

4.2.3 Input for the Combined Approach

As part of the combined approach presented in Section 3.5 we first need to use the formal approach to obtain probability enclosures for SnPHS models. Unfortunately, the same PDRH Deceleration model shown for `GPEP Tool` input in Figure 4.5 can not be used directly due to the formal approach computation procedure technique. Instead, we need to replace derivative calculations for our random parameter manually. For example for the mentioned above Deceleration model (see Section 5.2) with the same nondeterministic parameter range and normal random parameter *beta* distribution it is necessary to replace the random parameter representation in PDRH format from *dist_ normal(mu,sigma)* to *dist_ normal(0,1)* and also change *mode1* derivative representation of $d/dt[v]$ from

$$alpha * exp(-alpha * tau + beta) - drag * v * v$$

to

$$alpha * exp(-alpha * tau + (beta * sigma + mu)) - drag * v * v.$$

See Figure 4.6 for more information. In other words we perform normal distribution by changing our random parameter *beta* in accordance with nondeterministic parameter ranges of *mu* and *sigma*.

```
...
#define mu 4.0; //1st nondeterministic parameter mu is fixed
[0,0.2]sigma; //2nd nondeterministic parameter sigma
dist_normal(0,1) beta; //random parameter

{mode 1;    // accelerating
  flow:
    d/dt[s]= v;
    d/dt[v]= alpha*exp(-alpha*tau+(beta*sigma+mu))-drag*v*v;
    d/dt[tau]= 1.0;
jump: (and (v >= v_100))==>@2(and(s'=s)(v'=v)(tau'=0)); }
...
```

Figure 4.6: Deceleration model encoded in PDRH format for combined approach.

Note that in case of uniform distribution (models "Good" and "Bad") where we have a random parameter r uniformly distributed over nondeterministic parameters $[\gamma, \theta]$ (see models description in Section 5.2), we need to scale r by $r * (\gamma - \theta) + \theta$ formula.

4.3 Confidence Interval Estimation Tool

The methods shown in Section 2.4 were realised as part of the Statistical CI engine. In particular I implemented Wilson, Agresti-Coull, Logit, Anscombe and Arcsine CIs (see Subsection 2.4.2) based on the CLT Interval. I also implemented the new modified CLT method (see Section 3.3), which is shown to have a better approximation of CI for probabilities near the bounds (see Section 3.3). In order to compare the results obtained from CIs based on the CLT Interval with CI based on Beta-Function, the Clopper-Pearson CI estimation method was also added (see Subsection 2.4.1). The Qint quadrature algorithm for QMC variance estimation (see Subsection 2.3.5) was investigated and implemented as an extra-method for CI construction.

The portion of `Confidence Interval Estimation Tool` in the ProbReach architecture is presented in Figure 4.1. The tool contains all the methods listed above. Every method can be run separately or simultaneously for easier results comparison. In this section I discuss an example input model, which can be used for CI construction and show examples of `Confidence Interval Estimation Tool` use.

More information about `Confidence Interval Estimation Tool's` usage with command line and output examples can be found in Subsection A.2.1 in Section A.2 of Appendix.

4.4 GPEP Tool

In this section I present the `GPEP Tool`, which uses the EP method (see Subsection 2.5.5) based on GP evaluation (see Section 2.5). The ability to apply GP method to SnPHSs was proved in Section 3.4 as a novel and essential aspect of GP application in hybrid systems featuring random parameters that depend on

nondeterministic parameters. The developed tool was also extended by the features to store and use current GP training data. These features allow the `GPEP Tool` to save a large amount of time by excluding time costs on model evaluation decision procedure and data training.

More information about `GPEP Tool's` usage with command line and output examples can be found in Subsection A.2.1 in Section Subsection A.2.2 in Section A.2 of Appendix.

Next in this section, I discuss input data options, provide brief architecture outlines and show examples of `GPEP Tool` usage.

4.4.1 Architecture

The main `GPEP` algorithm consists of several components (Figure 4.7): *OptimiseGP Hyperparameters*, *doTraining* and *ClassProbabilities Approximation*. The *doTraining* procedure (Figure 4.7) involves in turn *Expectation Propagation* algorithms (*OptimiseGP Hyperparameters* from Subsection 2.5.2 and *Expectation Propagation* from Subsection 2.5.5 are presented in Figure 4.8).

A more general `GPEP` class diagram is shown in Figure 4.9. This Figure represents the algorithms from Section 2.5, an efficient approximation inference method that uses two techniques - assumed density filtering and loopy belief propagation (see Subsection 2.5.5). For sampling the system's parameter space this algorithm uses RQMC (see Subsection 2.3.4) and QMC (see Subsection 2.3.3) methods as well as some auxiliary methods for parameter boxes (see Subsection 2.3.1) for the combined approach discussed in Section 3.5.

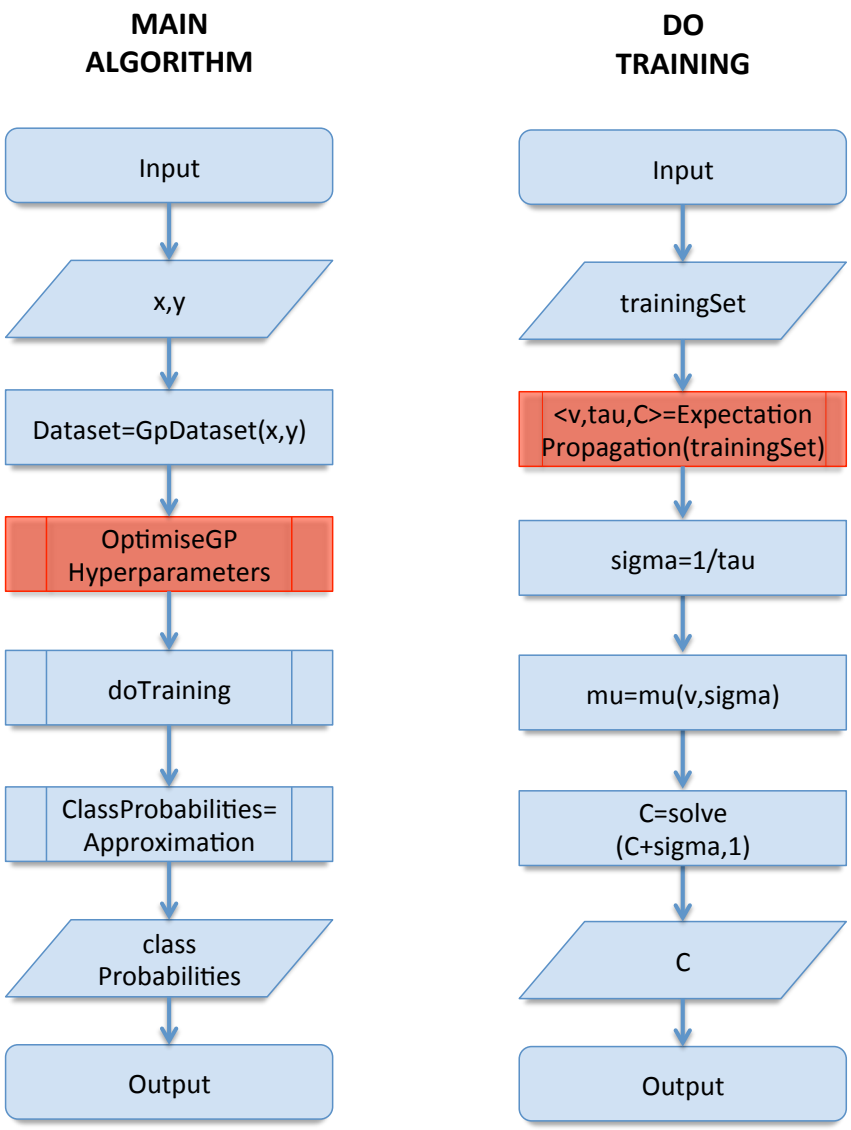


Figure 4.7: The GPEP architecture: main block and Training block.

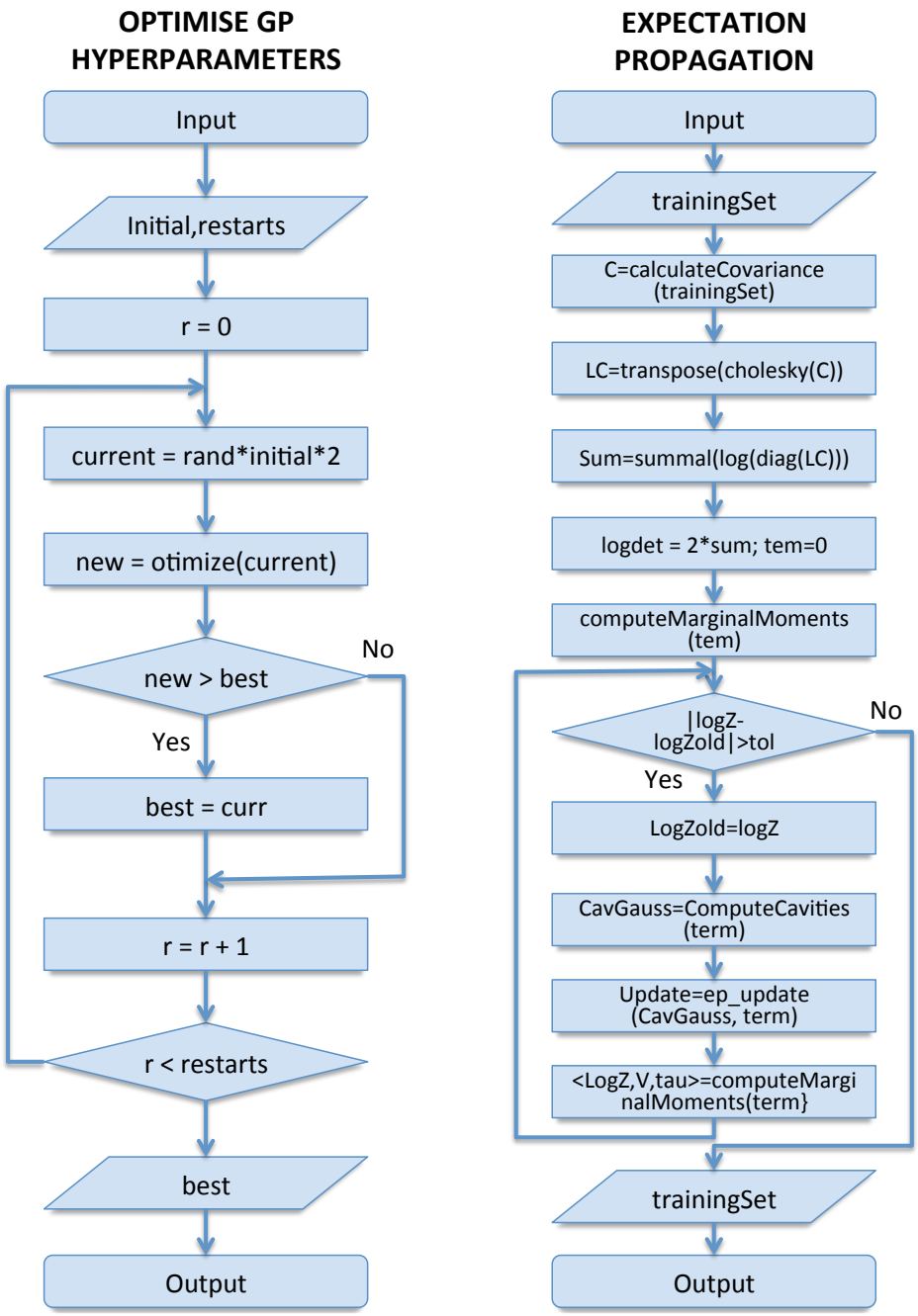


Figure 4.8: The GPEP architecture: Hyperparameters Optimiser block and Expectation Propagation block.

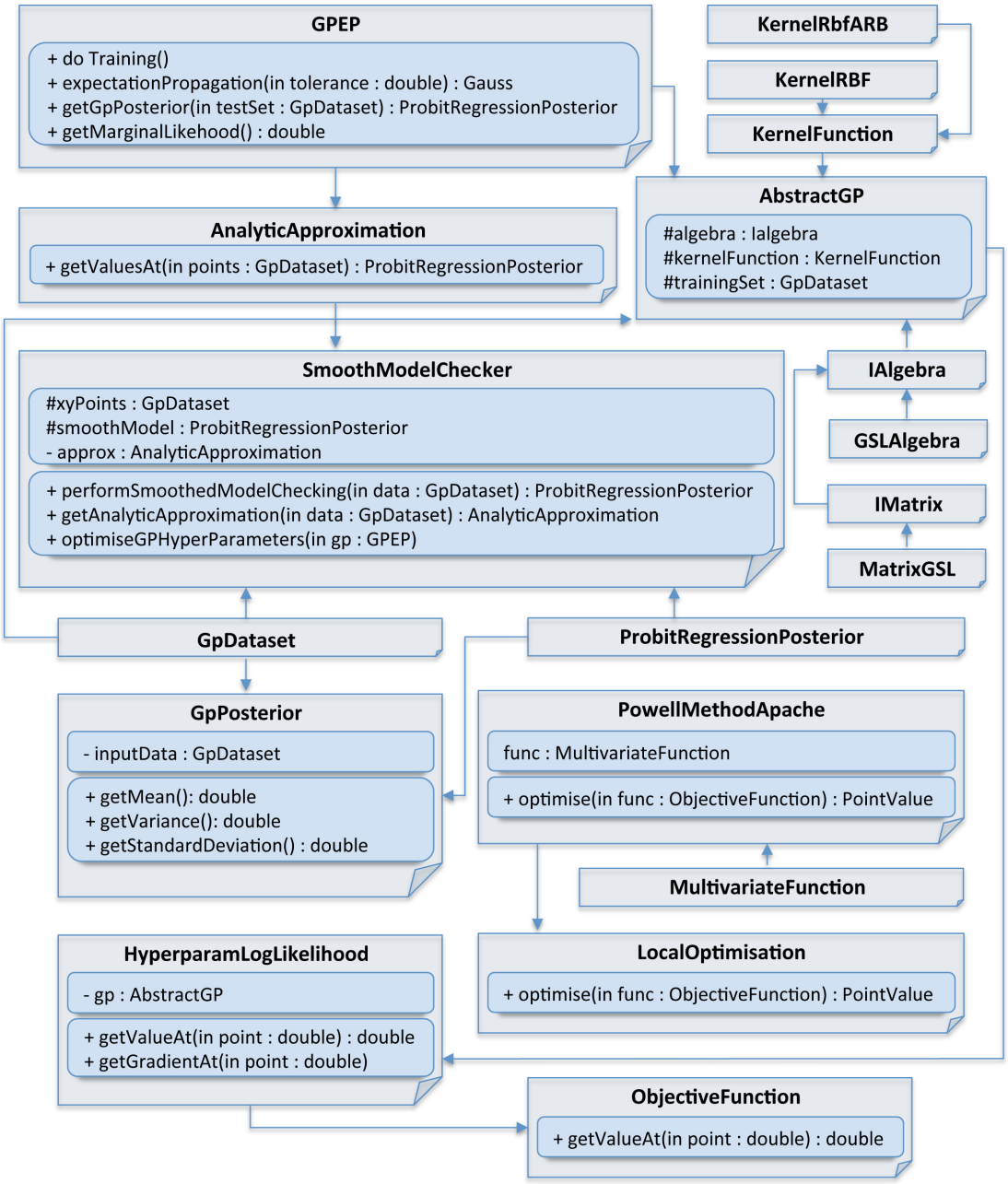


Figure 4.9: The GPEP tool classes.

The `GPEP` tool architecture, adapted from the Java implementation of the U-check tool (a tool for model checking for uncertain continuous time markov chains [11]) to C++ code presented in Figure 4.9 consists of the next main units:

- *SmoothModelChecker* implements the EP procedure for obtaining *ProbitRegressionPosterior*, via computing *GpPosterior*, which provides mean, variance and standard deviation for an approximated probability function.
- *AnalyticApproximation* utilises the *GPEP* procedure, which is used by *SmoothModelChecker* for computing the probability values. The *GPEP* procedure also provides training of the EP algorithm by using initial *GpDataset* and all parameters data in *AbstractGP*, including information about *KernelFunction* and matrices.
- *GpDataset* initial optimisation is made by using *LocalOptimisation* procedure via *PowellMethodApache* for finding a local minimum of a function through continuous reestimation of the nondeterministic parameters set for computing the probability value for multiple continuous random parameters.
- *AbstractGP* also generates *HyperparamLogLikelihood* for calculating likelihood hyperparameters (see Section 2.5) for EP marginal moment iteration as presented in Figure 4.8.

4.5 Summary

This Chapter introduces the `Confidence Interval Estimation Tool` and the `GPEP Tool`, which I developed for computing bounded reachability probability in stochastic parametric hybrid systems [73]. These tools enable the use of the developed methods for CI estimation, including the modified CLT method (see Subsection 3.3.2) and EP method (see Subsection 2.5.5).

This Chapter also includes the implementation of the novel statistical technique for computing the bounded reachability probability in SnPHSs, providing

statistically rigorous confidence intervals by combining formal approach, and the GP regression method (see Section 3.5).

The implementation is developed within the ProbReach tool and uses publicly available libraries, and it is distributed under the GNU General Public License[1] (GPL). It provides a C++ implementation for all of the algorithms presented in this thesis, which were parallelised using OpenMP. `ProbReach`, which includes `Confidence Interval Estimation Tool` and `GPEP Tool` is publicly available and does not require any commercial software.

[1] http://www.gnu.org/licenses/gpl.html

Chapter 5

Experiments Results

5.1 Introduction

In this Chapter, I show a successful usage of the implemented approaches in a few real-world studies. I compare different CIs estimation techniques for extreme probability cases, including the Qint method (see Subsection 2.3.5). I also present a comparison between the modified Qint method and the original algorithm. This Chapter also deals with the difference between the Bayesian CI and the CIs based on CLT, and the use of MC and QMC techniques for interval calculation.

As one of the most important outcomes of my research, I present the results produced via GP estimation techniques based on the EP algorithm and compare them to statistical model checking (SMC) CI estimation based on the standard Clopper-Pearson technique with standard MC sampling. I evaluate the accuracy of the GP approach by computing the average CI interval size and root mean squared error (RMSE) of my estimates across all input points. The results presented in this Chapter demonstrate that for all the SnPHS models examined GPs are more accurate than SMC. We can conclude that GP estimation with EP is generally very accurate and more accurate than Clopper-Pearson SMC, and thus it can be used for the verification of SnPHS.

In this Chapter I also study CPU time costs of the GP and SMC techniques and contrast the sample size necessary for both techniques to give results of similar accuracy. The obtained results demonstrated an excellent performance of

the GPEP algorithm showing a particular that SMC requires much longer CPU time and more samples to reach the CI results of GPEP.

Finally, in this Chapter I show the novel combined approach results. The experiments prove that the combined approach grants very encouraging results in terms of CI size and CPU time in comparison with the GPEP method. This approach definitely demands further research in order to find a good trade-off between the number of samples and CPU time spent and expansion on rare-event SnPHS models.

5.2 Case Studies

The main aim of my experiments was to reveal how different model types and their complexity can affect the computational result. Six models were chosen for the experiments. Despite the fact that the number of selected models is not large, the models allow me to effectively test the tools I have proposed because the selected set consists of both simple models whose reachability probability function can be calculated analytically and difficult complex models whose evaluation takes much time. At the same time, the selected models allow me to give an honest assessment of the proposed reachability probability evaluation methods since their probabilistic functions include complex, curved and almost flat lines that are located both in the middle of the probability space and close to the boundaries and cover different sizes of parameters regions - from very small (0.2) to huge (10,000). The full description of the models and the files can be found at: https://github.com/mariiavasileva/EPPR-models.

5.2.1 "Good" and "Bad" Models

These models are two basic introductory examples of a single mode non-hybrid system with constant flow dynamics ($\frac{dx}{dt} = 0$). The predicate $(x(0) = r) \wedge (n \in [0, 1])$, where r is uniformly distributed over $[m, f]$, where $m \in [0, 0.8]$ and $f \in [1.2, 2]$ are nondeterministic parameters is used to define the original state of the system. We used `ProbReach` for calculating 0-step bounded reachability

probability for two different goals: a *good* goal defined by the predicate

$$(x \leq 0.9n + 0.1) \wedge (x \geq 0.9n)$$

and a *bad* one shown by

$$(x \leq 2(n - 0.5)^2 + 0.5) \wedge (x \geq -2(n - 0.5)^2 + 0.5).$$

The projections of the goal set on the domain of continuous random parameters $P_R = [0, 1]$ for each n are the intervals $[0.9n, 0.9n + 0.1]$ and $[-2(n - 0.5)^2 + 0.5, 2(n - 0.5)^2 + 0.5]$ for the *good* and the *bad* cases, respectively. Taking in consideration that the random parameter r is distributed uniformly and x is a constant, the probability of reaching the goal can be computed as the difference of the right-hand side and the left-hand side of these intervals. Therefore, the probability function is constant $\mathbf{Pr}(n) = 0.1$ in the *good* case, while in the *bad* case it is equal to $\mathbf{Pr}(n) = 4n^2 - 4n + 1$, which reaches its minimum value of 0 at $n = 0.5$ and the maximum value of 1 at $n = 0$ and $n = 1$. Figure 5.1 demonstrate the graphs of the reachability probability functions obtained analytically for "Good" and "Bad" models respectively.

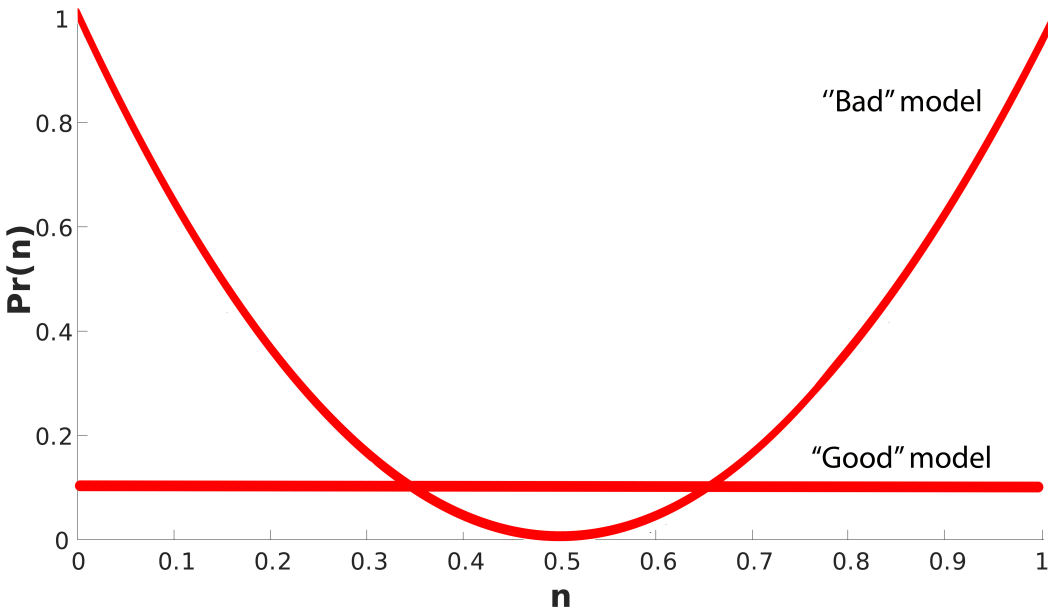


Figure 5.1: Reachability probability functions of "Good" and "Bad" models with the goal sets which depend on nondeterminstic parameter n.

Note that for SPHS good model results I use nondeterministic constant values: min value = 0.091 and max value = 0.091 and for SPHS bad model results I use min value = 0.5 and two max values: max = 0.987 and max2 = 0.028.

5.2.2 Deceleration Model

This model describes a car deceleration scenario. In the first mode the car accelerates from 0 to 27.78 m/s (0 to 100 km/h). During this period its velocity shifts according to $\frac{dv(t)}{dt} = \alpha \exp(-\alpha t + \beta) - c_d v^2(t)$, where $\alpha = 0.05776$ and $\beta \sim \mathcal{N}(\mu, \sigma)$ are coefficients modelling the acceleration properties of the car, which depend on the nondeterministic parameters $\mu \in [3.9, 4.1]$, $\sigma \in [0, 0.2]$ and $c_d = 3.028 \cdot 10^{-4}\ m^{-1}$ is the drag coefficient.

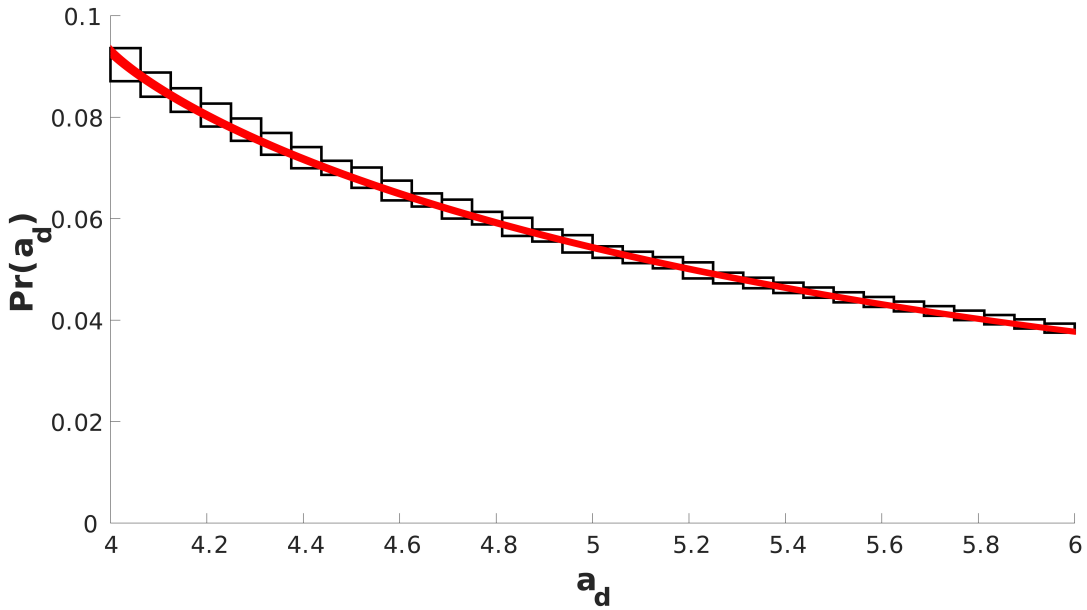


Figure 5.2: Reachability probability function of "Deceleration" model with the goal set which depends on nondeterminstic parameter a_d, where black boxes - probability enclosures computed by ProbReach and red line - graph of the probability enclosures mean function.

When the target velocity 27.78 m/s is achieved, it takes $t_{react} = 1.2$ seconds for the driver to react and to begin decelerating. There is no acceleration of the car in the "reaction" mode, and its velocity is controlled by the equation

$\frac{dv(t)}{dt} = -c_d v^2(t)$. In the final (braking) mode the car's deceleration is governed by the equation $\frac{dv(t)}{dt} = \mu a_d - c_d v^2(t)$ where $a_d \in [4.0, 6.0]$ is the car's deceleration (a nondeterministic parameter), and $\mu = 1$ is the coefficient modelling the road characteristics, such as slope, friction, etc. At full length of the modes the distance $s(t)$ covered by the car is controlled by $\frac{ds(t)}{dt} = v(t)$. We use `ProbReach` to calculate the probability of the car stopping within 400 meters. Figure 5.2 demonstrate the graph of the reachability probability functions obtained via ProbReach for the Deceleration model.

Note that for SPHS model results I use a_d min constant value = 5.697 and a_d max constant value = 4.117.

5.2.3 Collision Model

In this model a two-car collision scenario is presented, described by a 2-step bounded reachability problem in a three mode SnPHS. Two cars (Car1 and Car2) are moving on the same track, starting at $s_1(0) = 0$ and $s_2(0) = v_1 \cdot t_{safe}$ respectively (where t_{safe} is an interval of time for keeping a safe space between the cars). The initial velocity of both cars is 11.12 m/sec. In the initial mode Car1 changes tracks and begins to accelerate at $a_{a1} = 3\ m/sec^2$, while Car2 is going with the unchanged speed v_2. Car1 continues accelerating until it overtakes Car2 by the distance $v_2 \cdot t_{safe}$. Within the second mode Car1 returns to the initial lane and begins to decelerate at $a_{d1} \sim N(-2.0, 0.2)\ m/sec^2$, while the driver of Car2 is taking time to react and then beginning deceleration with (negative) acceleration $a_{d2} \sim N(-0.5, 0.1)\ m/sec^2$. It takes Car2 driver t_{react} seconds to react before braking. After that the final mode is initiated, where Car2 also decelerates with acceleration a_{d2} until it stops. We used `ProbReach` to calculate a set of enclosures for the probability of the cars collision for three different variants of this model:

- **Basic** - including one random 'and one nondeterministic parameter,
- **Extended** - featuring two random and one nondeterministic parameter,
- **Advanced** - with two random and two nondeterministic parameters.

Table 5.1 shows the parameter values and distributions used for calculating SPHS models, where nondeterministic parameters are presented in form of max

Model	a_{d1}	a_{d2}	t_{safe}	t_{react}
Basic	$\mathcal{N}(-2.0, 0.2)$	min=-0.393 max=-0.672	1.0	1.5
Extended	$\mathcal{N}(-2.0, 0.2)$	$\mathcal{N}(-0.5, 0.1)$	min=1.816 max=1.08	1.5
Advanced	$\mathcal{N}(-2.0, 0.2)$	$\mathcal{N}(-0.5, 0.1)$	min=1.921 max=1.221	min=1.081 max=1.107

Table 5.1: Parameter values and distributions for the cars collision model, where **Model** - name of the model, a_{d1} - deceleration of Car1, a_{d2} - deceleration of Car2, t_{safe} - time for maintaining safe distance, t_{react} - reaction time of the driver in Car2, $\mathcal{N}(\mu, \sigma)$ - represents the normal distribution with mean μ and standard deviation σ.

and min constants. We used ProbReach to calculate the probability of the cars collision for two different versions of this model: the *basic* model features one random parameter - deceleration of Car1; the *extended* model includes two random parameters for the deceleration of Car1 and Car2. Table 5.1 demonstrates the `ProbReach` settings.

Note also that for calculating SnPHS models the Basic type of the of the model was used with random parameter $a_{d1} \sim N(\mu, \sigma)$, which depends on the nondeterministic parameters $\mu \in [-2.1, -1.9]$ and $\sigma \in [0.1, 0.3]$ and the Extended type was used with two random parameters $a_{d1} \sim N(\mu 1, \sigma 1)$ and $a_{d2} \sim N(\mu 2, \sigma 2)$, which depends on the nondeterministic parameters $\mu 1 \in [-2.1, -1.9]$, $\sigma 1 \in [0.1, 0.3]$, $\mu 2 \in [-0.6, -0.4]$ and $\sigma 2 \in [0, 0.2]$.

5.2.4 Pharmocokinetics Model for Anaesthesia Delivery

This case study considers a pharmacokinetics model for anaesthesia delivery which tracks how the drug concentration changes as it is being metabolised by the body [29]. The model features three species:

- c_p - concentration of the drug in the plasma,
- c_1 - concentration of the drug in the fast peripheral compartment,
- c_2 - concentration of the drug in the slow peripheral compartment.

The dynamics of the system are governed by the set of differential equations (5.1).

$$\begin{aligned}
&\frac{dc_p(t)}{dt} = -(k_{10} + k_{12} + k_{13})c_p(t) + k_{12}c_1(t) + k_{13}c_2(t) + \frac{u(t)}{V_1},\\
&\frac{dc_1(t)}{dt} = k_{21}c_p(t) - k_{21}c_1(t),\ \frac{dc_2(t)}{dt} = k_{31}c_p(t) - k_{31}c_2(t),\\
&\frac{du(t)}{dt} = p\cos(\frac{2t\pi}{T_j}).
\end{aligned} \tag{5.1}$$

The model's scenario assumes that drug delivery is continuous, but every 15 minutes (starting at time 0) the drug infusion rate is subject to random errors. `ProbReach` computes the probability of reaching the unsafe state:

$$(c_p(t) \geq 6) \vee (c_p(t) \leq 1) \vee (c_1(t) \geq 10) \vee (c_1(t) \leq 0) \vee (c_2(t) \geq 10) \vee (c_2(t) \leq 0)$$

in 3 jumps within 60 minutes. Therefore, *the model features 4 continuous random parameters* Δu_i (one in the initial state and one per each jump)(see Table 5.2). This model does not feature any nondeterministic parameters, which allow `ProbReach` returns only one probability enclosure of the required length ϵ.

Initially chosen precision for the experiment conduction was equal to $\epsilon = 10^{-2}$. Unfortunately, `ProbReach` did not return the probability enclosure of the required precision with 360 hours. At the same time `ProbReach` used almost 10 Gigabytes of RAM for storing the parameter boxes partitioning the parameter space. This example illustrates that the computation time grows dramatically with the number of parameters (see Table 5.2). In order to solve this problem, it was decided to increase the precision value of ϵ to 5×10^{-2}. As a result the probability enclosure $[0.009769, 0.042274]$ of length 0.032505 was obtained within 80,823 seconds.

$du0$	$du1$	$du2$	$du3$	$u0$
$\mathcal{N}(0, 0.3 \cdot u0)$	$\mathcal{N}(0, 0.3 \cdot u0)$	$\mathcal{N}(0, 0.3 \cdot u0)$	$\mathcal{N}(0, 0.3 \cdot u0)$	7000

Table 5.2: Parameter values and distributions for the Anaesthesia model, where $du0$, $du1$, $du2$ and $du3$ - continuous random parameters, $\mathcal{N}(\mu, \sigma)$ - represents the normal distribution with mean μ, standard deviation σ and constant $u0$.

5.2.5 UVB Irradiation Therapy for Treating Psoriasis.

In this case a simplified version of a UVB irradiation therapy model [95] is used, which is applied in the treatment of psoriasis, an immune system-mediated chronic skin condition which is characterised by overproduction of keratinocytes.

The model consists of three classes of normal and three classes of psoriatic skin cells, their dynamics are presented by the nonlinear ODEs (5.2).

$$
\begin{aligned}
\frac{dSC}{dt} &= \gamma_1 \frac{\omega(1-\frac{SC+\lambda SC_d}{SC_{max}})SC}{1+(\omega-1)(\frac{TA+TA_d}{P_{ta,h}})^n} - \beta_1 In_A SC - \frac{k_{1s}\omega}{1+(\omega-1)(\frac{TA+TA_d}{P_{ta,h}})^n SC + k_1 TA},\\
\frac{dTA}{dt} &= \frac{k_{1a,s}\omega SC}{1+(\omega-1)(\frac{TA+TA_d}{P_{ta,h}})^n} + \frac{2k_{1s\omega}}{1+(\omega-1)(\frac{TA+TA_d}{P_{ta,h}})^n + \gamma_2 GA - \beta_2 In_A TA - k_{2s}TA - k_1 TA},\\
\frac{dGA}{dt} &= (k_{2a,s}+2k_{2s})TA - k_2 GA - k_3 GA - \beta_3 GA\\
\frac{SC_d}{dt} &= \gamma_{1d}(1-\frac{SC+SC_d}{SC_{max,t}})SC_d - \beta_{1d} In_A SC_d - k_{1sd}SC_d - \frac{k_p SC_d^2}{k_a^2+SC_d^2} + k_{1d}TA_d),\\
\frac{dTA_d}{dt} &= k_{1a,sd}SC_d + 2k_{1sd}SC_d + \gamma_{2d}TA_d + k_{2d}GA_d - \beta_{2d}In_A TA_d - k_{2sd}TA_d - k_{1d}TA_d,\\
\frac{dGA_d}{dt} &= (k_{2a,sd}+2k_{2sd})TA_d - k_{2d}GA_d - k_{3d}GA_d - \beta_{3d}GA_d.
\end{aligned}
\tag{5.2}
$$

The therapy involves a series of UVB irradiation episodes, which are simulated in the model by increasing In_A times the apoptosis rate constant for two cell types (stem cells and transit amplifying cells). The duration of each episode is 48 hours, followed by 8 hours of rest ($In_A = 1$), before the next irradiation can be started.

The efficacy of the therapy depends one the apoptosis rate (modified by In_A) and on the number of irradiation episodes. An insufficient number of treatments can lead to early psoriasis relapse: the deterministic variant of this model prognosticates psoriasis relapse for less than seven treatments [95]. Our model has *one random parameter* $In_A \sim N(\mu, 10,000)$, which depends on the uncertain parameter $\mu \in [55,000, 65,000]$. We compute the probability of a psoriasis relapse within a year following a seven-treatment therapy.

5.3 Confidence Interval Border Probability Cases

The true probability values which are shown in this Section were obtained via pseudo-random number generation that produces Boolean values according to a Bernoulli distribution.

5.3.1 Intervals Based on CLT and Bayesian Interval

The comparison of the different CIs estimation techniques for extreme probability cases (near 0 bound) with accuracy CI $\epsilon = 5 \times 10^{-3}$, which is presented in Figure 5.3, shows that all intervals except the Arcsin interval (see Equation 2.15) (see plot $c = 0.99$ of Figure 5.3 for probability=0.001) contain the true probability value within their bounds. The Bayesian method (see Subsection 2.4.1) tends to overestimate the true probability values according to their increase while CI_{CLT} tends to underestimate them. Also, it is interesting to note that the most accurate center value is returned by the Agresti-Coull interval (see Equation 2.12). The reason why CI_{CLT} tends to include the true probability value near the upper bound of the interval is directly related to the number of samples. As it is shown in Figure 5.3 for true probability values 0.007 - 0.01, the CI_{CLT} center is moving up evenly to the true probability value with the increase of the confidence value. It echoes the number of samples growth for obtaining the necessary confidence level. For the other true probability values (0.001-0.006), although this trend retained, it can not be seen from the Figure, because of the small difference in the number of samples for all confidence levels, which causes the CI center to move wave-like.

The results in Figure 5.3 also demonstrate that the CIs based on the standard interval (see Subsection 2.4.2) can have interval size smaller than its nominal value even for "large" sample sizes. It can be seen that every confidence level from 0.99 to 0.99999 displays further instances of the inadequacy of the CIs size. Also, Figure 5.3 shows that the size of the CI_B (Bayesian), CI_{CLT} (CLT) and CI_{AC_W} (Agresti-Coull) intervals decreases significantly as p moves toward 0. Also, CIs based on the standard interval have interval size changes because of two reasons: absence of a posteriori estimate and skewness of the underlying binomial distribution.

In Figure 5.4 I plot the number of samples that different CI estimation techniques used to return intervals with accuracy (size) $\epsilon = 5 \times 10^{-3}$ for different confidence levels. It can be clearly seen from the plots that with the increasing of the confidence all CIs based on the standard interval outperform the Bayesian CI (see Equation 2.9). The plot with $c = 0.99999$ in Figure 5.4 displays that the best techniques in the number of samples from the best to the worst are: CI_{CLT} (CLT), $Qint$ (Qint), CI_{Arc} (Arcsine), CI_W (Wilson), CI_L (Logit), CI_{Ans} (Anscombe), CI_{AC_W} (Agresti-Coull) and CI_B (Bayesian). The CI_L and CI_{Ans} techniques always show almost the same results near the bounds, because of the modification of the CI_L. Initially, CI_L is not able to deal with probability values near the bounds according to its λ formula (see Subsection 2.4.2). It has been modified to use the Anscombe estimation formula in cases when $\hat{p} = 0$ or $\hat{p} = 1$. It is also important to note that the difference in sample number between CI_{CLT}, CI_{Arc} and CI_B for extreme probability cases is significant. For example in the plot with $c = 0.9999$ of Figure 5.4 the number of samples used to obtain interval for $p = 0.005$ equals to 1,078 for CI_{CLT}, 2,662 for the CI_{Arc} and 4,440 for CI_B.

This trend is not preserved with the increase of the probability value from 0 to 0.5 and with the decrease from 1 to 0.5, respectively. Figure 5.5 shows that the difference in sample number between all CIs (except CI_{Arc}) is almost undetectable. At the same time, CI_{Arc} shows very "bad" results in comparison with the others, as opposed to its results for probability values at the extremes.

Summarising, for probability values near the bounds (0 or 1) the modified CLT method (see Section 3.3), named here CI_{CLT} achieves better results in number of samples in comparison with the others (see Figure 5.4). For probability values away from the bounds, CLT, Wilson, Agresti–Coull, Logit and Anscombe methods are all very similar (see Figure 5.5), and so for such probabilities we come to the conclusion that the CLT interval should be recommended, due to its simplest form. Meanwhile for smaller sample sizes, the CI_{CLT} is strongly preferable to the others and so might be the choice where sampling cost is paramount.

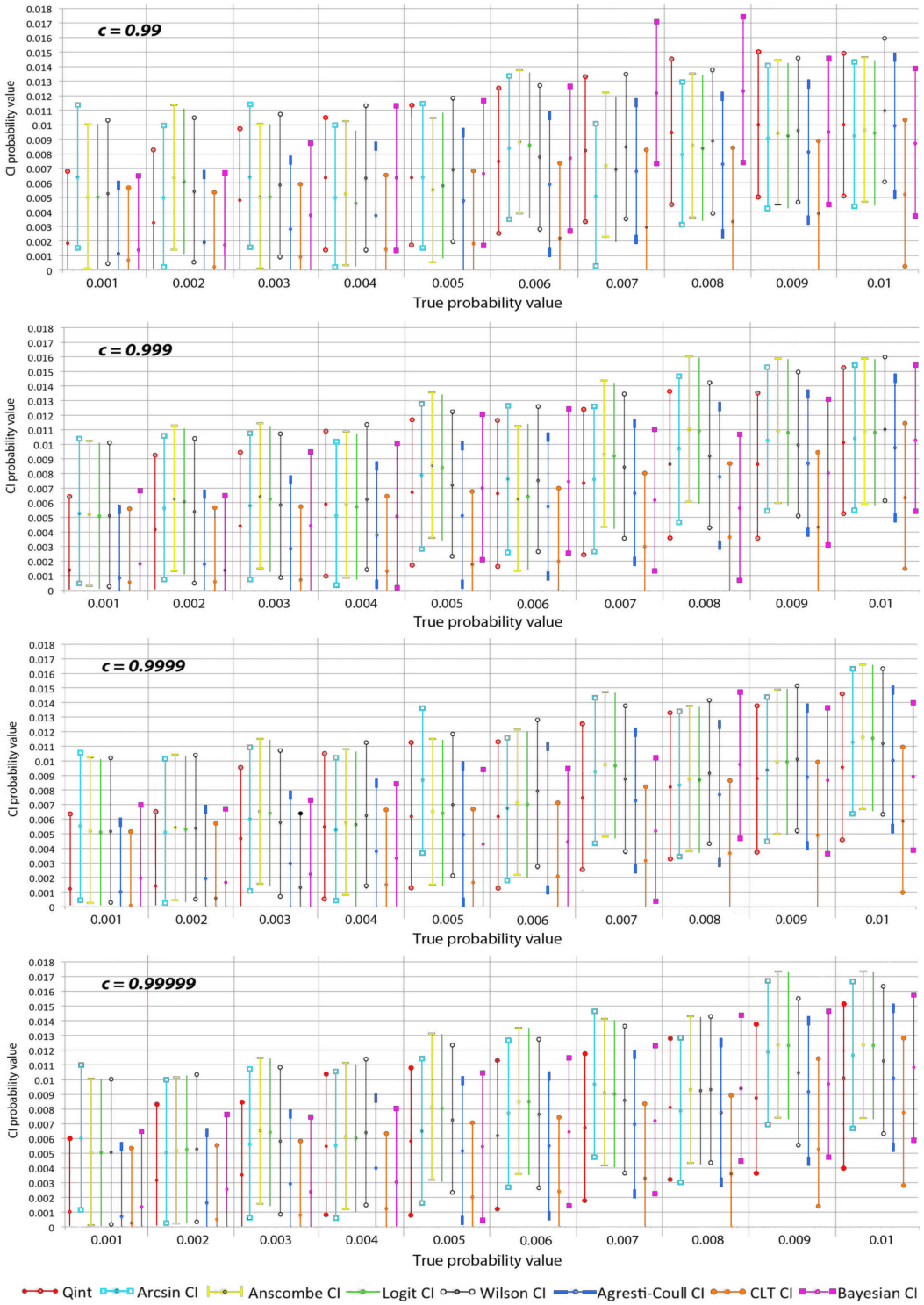


Figure 5.3: Comparison of confidence interval distribution for probability values near 0, interval size equal to 10^{-2} and $\boldsymbol{c}$ - confidence level.

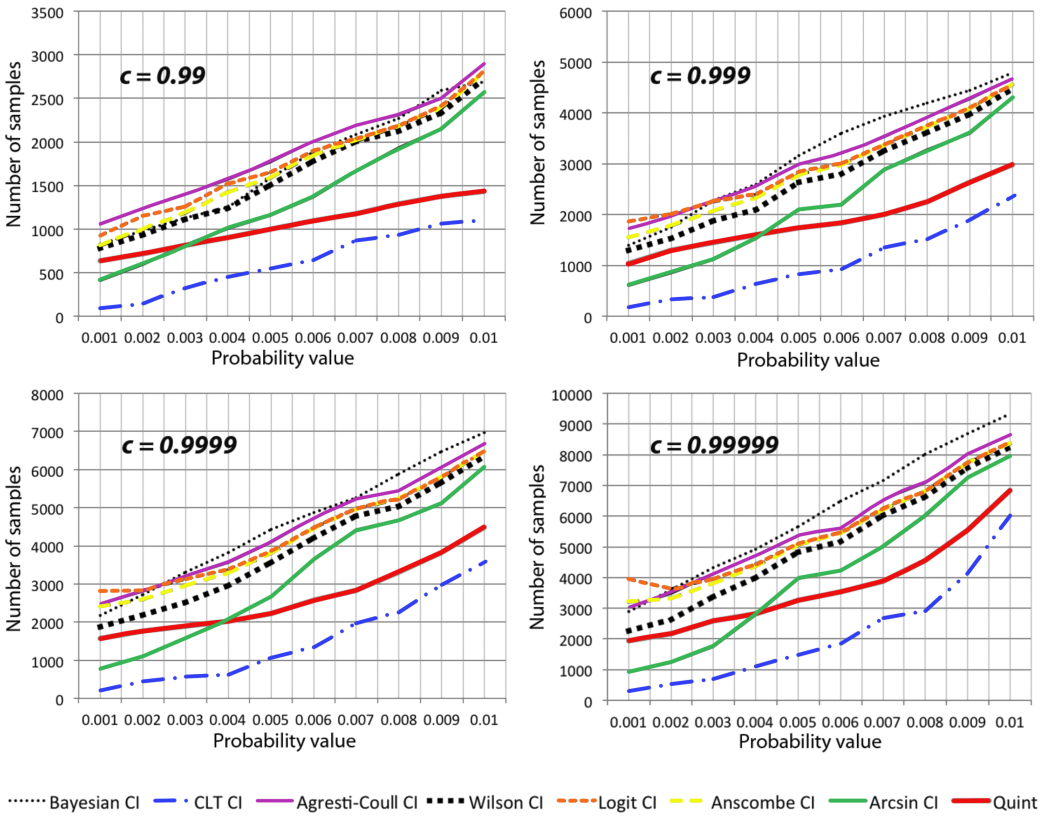


Figure 5.4: Comparison of sample size for probability values near 0, interval size equal to 10^{-2} and $\boldsymbol{c}$ - confidence level.

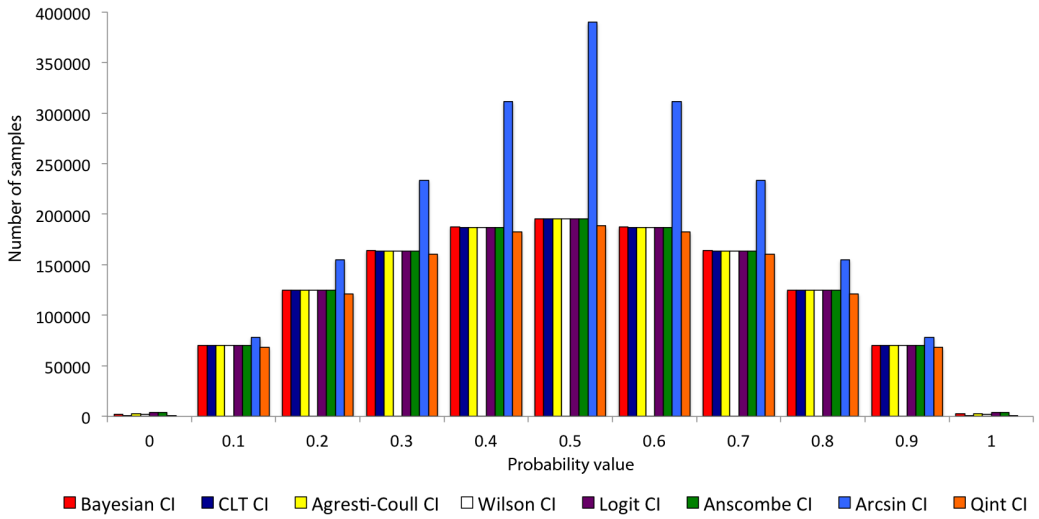


Figure 5.5: Comparison of sample size for probability values from 0 to 1, interval size equal to 10^{-2} and confidence level equal to 0.99999.

5.3.2 Qint Method Results

In Figure 5.3 and Figure 5.4, I also plotted the results of the recently developed Qint algorithm [1]. In my research we used Qint with $n = k \times 2^s$, where $k = 2$ (see Subsection 2.3.5). These parameters were used to form n points of the Sobol sequence x_i with numbers $i \in I_{k,s} = \{1, 2, ..., k \times 2^s\}$. These parameters were chosen on the basis of the original study of the Qint method as the most universal and reliable. As it was described earlier, Qint uses a cubature randomisation method and provides the integral estimation variance (see Equation 2.8). This formula is used to obtain a CI by calculating the standard interval (see Equation 3.5) with our modification.

In Figure 5.3, I display the Qint intervals distribution for border probability values. We can see from the plots that the Qint CI always contains the true probability value. At the same time for all confidence levels from 0.99 to 0.99999 and for true probability values 0.006-0.01, Qint shows better centration than CI_B and CI_{CLT}. The greatest differences between the Qint CLT center result and the true probability values are: 0.00245 for $c = 0.99$ (p=0.004), 0.00191 for $c = 0.999$ (p=0.004), 0.00168 for $c = 0.9999$ (p=0.003), 0.00141 for $c = 0.99999$ (p=0.004),

while for example this difference for CI_B reaches 0.00518 for $c = 0.99$ (p=0.007), 0.00235 for $c = 0.999$ (p=0.008), 0.00181 for $c = 0.9999$ (p=0.008), 0.00143 for $c = 0.99999$ (p=0.008).

We can see in Figure 5.4 that, as it was expected, Qint uses fewer samples than other CIs but CI_{CLT}. This modification allows the Qint algorithm to return intervals even if $n_s = 0$, which significantly decreases the final sample size for 10 runs. With the increase in n, which leads to further sampling and better reflects the behaviour of the underlying random process, the effectiveness of the method decreases, and the benefit no longer seems so significant.

The fact that with the chosen parameters Qint cannot outperform our modified CI_{CLT} leads us to the conclusion that the usage of the standard deviation formula with $\frac{1}{n^2}$ lower bound is a rather effective and simple solution. However, the deep range of the possible parameters variation as well as the novelty of the Qint algorithm suggest that further research towards their comparison is needed.

5.4 Monte Carlo and Quasi-Monte Carlo Error Comparison

Another key difference between the Bayesian CI (see Subsection 2.4.1) and the CIs based on CLT (see Subsection 2.4.2) is the use of MC and QMC techniques (see Section 2.3) for interval calculation. As it was described in Subsection 2.3.3, the QMC advantage in the error size holds for all of the tested models. In the cases where the true error rate could not be detected due to the probability value being extremely close to 0 ("Bad" model type min and Collision (Basic) model type min), we have that the MC absolute error line equals the true probability value, because $n_s = 0$ was obtained.

The chaotic coverage properties of the MC method (see Subsection 2.3.2) are far more persistent than they are appreciated. The chaotic behaviour does not disappear even when n is quite large and the true probability p is not near the boundaries. For instance, in Figure 5.6 it is visible that even when n is quite large (*i.e.*, tends to 10,000 samples) the actual absolute error value of the MC method reaches 5×10^{-3}. Hence we can conclude that CIs estimation techniques

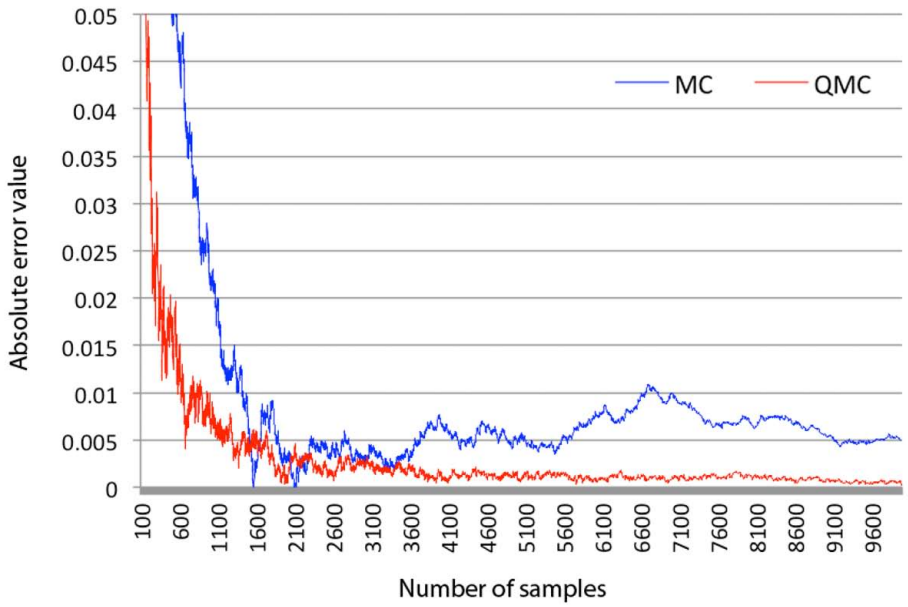


Figure 5.6: MC (blue line) and QMC (red line) absolute error with respect to the number of samples. Model: Collision advanced, type - max.

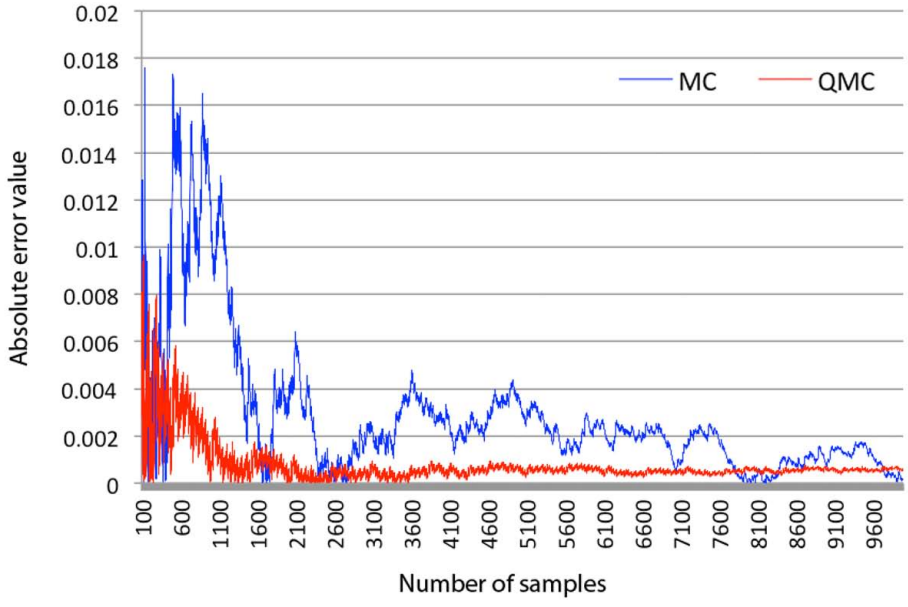


Figure 5.7: MC (blue line) and QMC (red line) absolute error with respect to the number of samples. Model: Collision advanced, type - max.

based on MC are misleading and defective in several respects and should not be trusted [16].

A phenomenon that was noticed for the MC and QMC probability calculation is that the actual coverage probability contains non-negligible oscillations as both p and n vary. There exist some "unlucky" pairs (p, n) such that the corresponding absolute error is much greater than the results for smaller n. The phenomenon of oscillation is both in n, for fixed p, and in p, for fixed n. Furthermore, drastic changes in coverage occur in nearby p for fixed n and in nearby n for fixed p [16]. We can see it on the simple example in Figure 5.7.

5.5 Confidence Interval Tested Models Results

The results in this Section were obtained via the ProbReach tool [73] for computing bounded reachability in stochastic parametric hybrid systems and the dReal solver [31] analyzing (standard) bounded reachability question.

5.5.1 Intervals Based on CLT and Bayesian Interval

Based on the model set (see Section 5.2), I provide in Tables 5.3, 5.4, 5.5 and 5.6 a comparison of the CIs described in Section 2.4, obtained via ProbReach with precision $\delta = 10^{-3}$, interval size 10^{-2} and where the true probability value $\boldsymbol{P}$ is either analytically computed single probability values or formally computed absolute (non-statistical) intervals. These parameters were chosen according to previous work [73] as a fine trade-off between the precision of the results and the CPU time. Each model was verified separately with different confidence levels from 0.99 to 0.99999. The lowest confidence level is often used in literature, while the highest can provide reasonable results for real-world complex models.

As it can be seen in Tables 5.3, 5.4, 5.5 and 5.6, all the intervals for the various techniques overlap. The modified CI_{CLT} approach (see Subsection 3.3) shows very similar results to the CI_B, which can be regarded as a successful implementation. The key difference in the interval sizes can be found in the results of the "Bad" model Type min and the Collision (Basic) model Type min (see Tables 5.3, 5.4, 5.5 and 5.6). From the results we can conclude that the true

probability value is very close to 0. This allows the Bayesian, CLT and Agresti-Coull methods to form intervals, which in reality are half of the proposed interval size 10^{-2}, while the other techniques return "fully" sized intervals. That happens because CI_B is using a posterior distribution to form the interval (interval size 10^{-2}). At the same time, the CI_{CLT} and CI_{AC_W} calculations of the mean value return the result, which is quite close to zero. Thus, the next step of the interval bounds computation cuts the negative part of the interval. This trend holds for all probability values within $[0, 0.001]$.

Tables 5.5 and 5.6 also shows that with the increase of the confidence level the interval's precision is growing, which in turn is directly related to the use of the inverse cumulative distribution function for normal random variable with given confidence level in formulas for CI_{CLT} (3.5), CI_W (2.11), CI_{AC_W} (2.13) and CI_{Arc} (2.15). It also results in the increase of the sample size n for CI_L and CI_{Anc}.

The comparison of the obtained intervals (see Table 5.3) with the true probability value or interval $\boldsymbol{P}$ shows that all CIs contain the single probability values but CI_{Acr} (see "Bad" type min model of Table 5.3), and all CIs overlap with the true probability intervals. We can also note that the true probability intervals of the Collision Extended, Collision Advanced, and Anaesthesia models contain all confidence intervals for all confidence levels (see Tables 5.3 and 5.5). The reason why Collision Basic and Deceleration models' true probability intervals do not contain CIs is their size (< 0.01).

Table 5.7 provides very interesting results with respect to the number of samples which were used to find CIs obtained via ProbReach with solver precision $\delta = 10^{-3}$ and interval size 10^{-2}. The number of samples varies for different models and types. As it was noted earlier, the number of samples needed for the computation grows from the bounds to the center of the [0,1] interval. The presented models show different behaviour and probability results. The most important outcome is that all CIs (except CI_{Arc}) show better result in number of points with respect to CI_B. The best result was shown by CI_{CLT}. It shows that the proposed CLT modification can provide reasonable results for RQMC calculation in comparison with the well-established Bayesian MC integral calculation.

Confidence level c=0.99						
Model	**Type**	***P***	CI_B	CI_{CLT}	CI_W	CI_{AC_W}
Good	max	0.1	[0.09671, 0.10671]	[0.09564, 0.10564]	[0.09632, 0.10632]	[0.09574, 0.10574]
	min	0.1	[0.09529, 0.10529]	[0.0956, 0.1056]	[0.09666, 0.10666]	[0.09679, 0.10679]
Bad	max	0.95001	[0.94416, 0.95416]	[0.94495, 0.95493]	[0.94422, 0.95422]	[0.94397, 0.95397]
	max2	0.88747	[0.8825, 0.8925]	[0.88028, 0.89028]	[0.88031, 0.88031]	[0.88019, 0.89019]
	min	4×10^{-7}	[0, 0.00525]	[0, 0.005]	[0, 0.00483]	[0, 0.00955]
Deceleration	max	[0.08404, 0.08881]	[0.08471, 0.09471]	[0.08802, 0.09802]	[0.08817, 0.09817]	[0.08685, 0.09685]
	min	[0.04085, 0.04275]	[0.03835, 0.04835]	[0.03861, 0.04861]	[0.03854, 0.04854]	[0.03884, 0.04884]
Collision (Basic)	max	[0.96567, 0.97254]	[0.96371, 0.97381]	[0.96873, 0.97873]	[0.9684, 0.9784]	[0.96851, 0.97851]
	min	[0, 0.00201]	[0, 0.00525]	[0, 0.005]	[0, 0.00483]	[0, 0.00955]
Collision (Extended)	max	[0.35751, 0.49961]	[0.42267, 0.43675]	[0.42418, 0.4342]	[0.42187, 0.43187]	[0.42345, 0.43345]
	min	[0.04296, 0.06311]	[0.0482, 0.0582]	[0.04772, 0.05772]	[0.04785, 0.05785]	[0.04823, 0.05823]
Collision (Advanced)	max	[0.14807, 0.31121]	[0.2072, 0.2172]	[0.20873, 0.21872]	[0.21872, 0.2185]	[0.20854, 0.21854]
	min	[0.02471, 0.05191]	[0.02631, 0.03631]	[0.03045, 0.04045]	[0.03016, 0.04016]	[0.03001, 0.04]
Anesthesia	n/a	[0.00916, 0.04222]	[0.01361, 0.02361]	[0.01339, 0.02332]	[0.01374, 0.02374]	[0.01373, 0.02373]
Confidence level c=0.999						
Model	**Type**	***P***	CI_B	CI_{CLT}	CI_W	CI_{AC_W}
Good	max	0.1	[0.09555, 0.10555]	[0.09559, 0.10559]	[0.09559, 0.10559]	[0.0957, 0.1057]
	min	0.1	[0.09393, 0.10393]	[0.0961, 0.1061]	[0.09613, 0.10613]	[0.0962, 0.1062]
Bad	max	0.95001	[0.94549, 0.95549]	[0.94544, 0.95544]	[0.94526, 0.95526]	[0.94504, 0.95504]
	max2	0.88747	[0.88165, 0.89165]	[0.88069, 0.89069]	[0.88071, 0.89071]	[0.8806, 0.8906]
	min	4×10^{-7}	[0, 0.00525]	[0, 0.005]	[0, 0.00489]	[0, 0.00972]
Deceleration	max	[0.08404, 0.08881]	[0.08695, 0.09695]	[0.08659, 0.09659]	[0.08656, 0.09656]	[0.0867, 0.0967]
	min	[0.04085, 0.04275]	[0.03785, 0.04785]	[0.0362, 0.0462]	[0.04, 0.05]	[0.0403, 0.0503]
Collision (Basic)	max	[0.96567, 0.97254]	[0.96397, 0.97052]	[0.96362, 0.97362]	[0.96392, 0.97392]	[0.96412, 0.97412]
	min	[0, 0.00201]	[0, 0.00525]	[0, 0.005]	[0, 0.00489]	[0, 0.00972]
Collision (Extended)	max	[0.35751, 0.49961]	[0.42673, 0.43676]	[0.42477, 0.43477]	[0.42634, 0.42634]	[0.42441, 0.42441]
	min	[0.04296, 0.06311]	[0.05004, 0.06004]	[0.05173, 0.06173]	[0.04189, 0.05189]	[0.04244, 0.05244]
Collision (Advanced)	max	[0.14807, 0.31121]	[0.20451, 0.21456]	[0.20639, 0.21643]	[0.20623, 0.21623]	[0.20617, 0.21617]
	min	[0.02471, 0.05191]	[0.02626, 0.03633]	[0.03256, 0.04256]	[0.02666, 0.03666]	[0.03212, 0.04212]
Anesthesia	n/a	[0.00916, 0.04222]	[0.01285, 0.02285]	[0.01497, 0.02495]	[0.01574, 0.02574]	[0.01492, 0.02492]

Table 5.3: Results for CI computation obtained via ProbReach, with solver precision δ=10^{-3} and interval size equal to 10^{-2}, **Type** - extremum type and ***P*** - true probability value.

Confidence level c=0.99						
Model	**Type**	***P***	CI_L	CI_{Ans}	CI_{Arc}	$Qint$
Good	max	0.1	[0.09575, 0.10575]	[0.09577, 0.10577]	[0.09559, 0.10559]	[0.09147, 0.10147]
	min	0.1	[0.09678, 0.10678]	[0.0968, 0.1068]	[0.09639, 0.10639]	[0.09164, 0.10164]
Bad	max	0.95001	[0.94396, 0.95396]	[0.94392, 0.95392]	[0.94735, 0.95735]	[0.94459, 0.95459]
	max2	0.88747	[0.8803, 0.8902]	[0.88019, 0.89019]	[0.88325, 0.89325]	[0.88136, 0.89136]
	min	4×10^{-7}	[0.00005, 0.00959]	[0.00005, 0.00959]	[0.00131, 0.00959]	[0,0.005]
Deceleration	max	[0.08404, 0.08881]	[0.08614, 0.09614]	[0.0863, 0.0963]	[0.08963, 0.09932]	[0.08852, 0.09852]
	min	[0.04085, 0.04275]	[0.03886, 0.04886]	[0.0389, 0.0489]	[0.03873, 0.04873]	[0.03337, 0.04337]
Collision (Basic)	max	[0.96567, 0.97254]	[0.96875, 0.97875]	[0.96853, 0.97853]	[0.96851, 0.97851]	[0.96301, 0.97301]
	min	[0, 0.00201]	[0.00005, 0.00959]	[0.00005, 0.00959]	[0.00131, 0.00959]	[0,0.005]
Collision (Extended)	max	[0.35751, 0.49961]	[0.42463, 0.43463]	[0.42457, 0.43457]	[0.42385, 0.43385]	[0.42342, 0.43342]
	min	[0.04296, 0.06311]	[0.04812, 0.05812]	[0.0481, 0.0581]	[0.04757, 0.05772]	[0.04618, 0.05618]
Collision (Advanced)	max	[0.14807, 0.31121]	[0.20854, 0.21854]	[0.20855, 0.21855]	[0.20111, 0.21111]	[0.20167, 0.21166]
	min	[0.02471, 0.05191]	[0.03001, 0.04]	[0.03016, 0.04016]	[0.03164, 0.04164]	[0.0304, 0.0404]
Anesthesia	n/a	[0.00916, 0.04222]	[0.01318, 0.02318]	[0.01311, 0.02311]	[0.01592, 0.02592]	[0.01815, 0.02815]
Confidence level c=0.999						
Model	**Type**	***P***	CI_L	CI_{Ans}	CI_{Arc}	$Qint$
Good	max	0.1	[0.09572, 0.10572]	[0.09571, 0.10571]	[0.09528, 0.10528]	[0.09444, 0.10444]
	min	0.1	[0.09619, 0.10619]	[0.0962, 0.1062]	[0.09637, 0.10637]	[0.09263, 0.10263]
Bad	max	0.95001	[0.94502, 0.95502]	[0.94499, 0.95499]	[0.94735, 0.95735]	[0.94564, 0.95564]
	max2	0.88747	[0.88061, 0.89061]	[0.88061, 0.89061]	[0.88325, 0.89325]	[0.88059, 0.89059]
	min	4×10^{-7}	[0.00005, 0.00978]	[0.00005, 0.00978]	[0.00024, 0.01173]	[0,0.005]
Deceleration	max	[0.08404, 0.08881]	[0.08675, 0.09675]	[0.08683, 0.09683]	[0.0868, 0.0968]	[0.08825, 0.08925]
	min	[0.04085, 0.04275]	[0.0402, 0.0502]	[0.0403, 0.0503]	[0.04001, 0.05001]	[0.03495, 0.04495]
Collision (Basic)	max	[0.96567, 0.97254]	[0.96521, 0.97521]	[0.96516, 0.97516]	[0.96851, 0.97851]	[0.96305, 0.97305]
	min	[0, 0.00201]	[0.00005, 0.00978]	[0.00005, 0.00978]	[0.00024, 0.01173]	[0,0.005]
Collision (Extended)	max	[0.35751, 0.49961]	[0.42448, 0.42448]	[0.42434, 0.42434]	[0.42345, 0.43345]	[0.43553, 0.45553]
	min	[0.04296, 0.06311]	[0.04385, 0.05385]	[0.04382, 0.05382]	[0.04483, 0.05483]	[0.04499, 0.05499]
Collision (Advanced)	max	[0.14807, 0.31121]	[0.20647, 0.21647]	[0.20643, 0.21643]	[0.20111, 0.21111]	[0.20469, 0.21469]
	min	[0.02471, 0.05191]	[0.03301, 0.04301]	[0.03301, 0.04301]	[0.03364, 0.04364]	[0.03112, 0.04112]
Anesthesia	n/a	[0.00916, 0.04222]	[0.01446, 0.02446]	[0.01442, 0.02442]	[0.01592, 0.02592]	[0.01774, 0.02774]

Table 5.4: Results for CI computation obtained via ProbReach, with solver precision δ=10^{-3} and interval size equal to 10^{-2}, **Type** - extremum type and ***P*** - true probability value.

Confidence level c=0.9999						
Model	**Type**	***P***	CI_B	CI_{CLT}	CI_W	CI_{AC_W}
Good	max	0.1	[0.09621, 0.10621]	[0.09465, 0.10465]	[0.09464, 0.10464]	[0.09478, 0.10478]
	min	0.1	[0.09352, 0.10352]	[0.09662, 0.10662]	[0.09669, 0.10669]	[0.09672, 0.10672]
Bad	max	0.95001	[0.94477, 0.95477]	[0.94595, 0.95595]	[0.94598, 0.95598]	[0.94574, 0.95574]
	max2	0.88747	[0.88208, 0.89208]	[0.88058, 0.89058]	[0.88061, 0.89061]	[0.88049, 0.89049]
	min	4×10^{-7}	[0, 0.00525]	[0, 0.005]	[0, 0.00492]	[0, 0.00979]
Deceleration	max	[0.08404, 0.08881]	[0.08593, 0.09593]	[0.08631, 0.09631]	[0.08629, 0.09629]	[0.08642, 0.09642]
	min	[0.04085, 0.04275]	[0.03764, 0.04764]	[0.0394, 0.0494]	[0.03438, 0.04438]	[0.03972, 0.04972]
Collision (Basic)	max	[0.96567, 0.97254]	[0.96371, 0.97373]	[0.96285, 0.97285]	[0.96799, 0.97799]	[0.96792, 0.97792]
	min	[0, 0.00201]	[0, 0.00525]	[0, 0.005]	[0, 0.00492]	[0, 0.00979]
Collision (Extended)	max	[0.35751, 0.49961]	[0.42492, 0.43495]	[0.42799, 0.43804]	[0.42157, 0.43157]	[0.42356, 0.43356]
	min	[0.04296, 0.06311]	[0.04933, 0.05933]	[0.04708, 0.05708]	[0.04864, 0.05864]	[0.04848, 0.05848]
Collision (Advanced)	max	[0.14807, 0.31121]	[0.20531, 0.21537]	[0.20617, 0.21621]	[0.20633, 0.21633]	[0.20631, 0.21631]
	min	[0.02471, 0.05191]	[0.02895, 0.03895]	[0.02937, 0.03937]	[0.02984, 0.03984]	[0.03823, 0.04823]
Anesthesia	n/a	[0.00916, 0.04222]	[0.01388, 0.02388]	[0.01428, 0.02427]	[0.01399, 0.02399]	[0.01425, 0.02425]
Confidence level c=0.99999						
Model	**Type**	***P***	CI_B	CI_{CLT}	CI_W	CI_{AC_W}
Good	max	0.1	[0.09499, 0.10499]	[0.09378, 0.10378]	[0.09386, 0.10386]	[0.09389, 0.10389]
	min	0.1	[0.09419, 0.10419]	[0.09667, 0.10667]	[0.09668, 0.10668]	[0.09677, 0.10677]
Bad	max	0.95001	[0.94525, 0.95525]	[0.94579, 0.95579]	[0.94564, 0.95564]	[0.94548, 0.95548]
	max2	0.88747	[0.88215, 0.89215]	[0.88055, 0.89055]	[0.88057, 0.89057]	[0.88046, 0.89046]
	min	4×10^{-7}	[0, 0.00517]	[0, 0.00319]	[0, 0.00494]	[0, 0.00984]
Deceleration	max	[0.08404, 0.08881]	[0.08613, 0.09613]	[0.08624, 0.09624]	[0.08312, 0.09312]	[0.08725, 0.09725]
	min	[0.04085, 0.04275]	[0.03514, 0.04514]	[0.03919, 0.04919]	[0.03918, 0.04918]	[0.03942, 0.04942]
Collision (Basic)	max	[0.96567, 0.97254]	[0.96359, 0.97359]	[0.96241, 0.97241]	[0.96767, 0.9767]	[0.96892, 0.96892]
	min	[0 , 0.00201]	[0, 0.00517]	[0, 0.00319]	[0, 0.00494]	[0, 0.00984]
Collision (Extended)	max	[0.35751, 0.49961]	[0.42651, 0.43652]	[0.42719, 0.43724]	[0.42757, 0.43757]	[0.42656, 0.43656]
	min	[0.04296, 0.06311]	[0.04979, 0.05979]	[0.04766, 0.05766]	[0.04764, 0.05764]	[0.04748, 0.05748]
Collision (Advanced)	max	[0.14807, 0.31121]	[0.20515, 0.21519]	[0.20558, 0.21563]	[0.20533, 0.21533]	[0.20531, 0.21531]
	min	[0.02471, 0.05191]	[0.03011, 0.04015]	[0.02902, 0.03902]	[0.02954, 0.03945]	[0.03956, 0.04956]
Anesthesia	n/a	[0.00916, 0.04222]	[0.01284, 0.02284]	[0.01513, 0.02511]	[0.01623, 0.02623]	[0.01545, 0.02545]

Table 5.5: Results for CI computation obtained via ProbReach, with solver precision δ=10^{-3} and interval size equal to 10^{-2}, **Type** - extremum type and ***P*** - true probability value.

Confidence level c=0.9999						
Model	**Type**	***P***	CI_L	CI_{Ans}	CI_{Arc}	$Qint$
Good	max	0.1	[0.09477, 0.10477]	[0.09424, 0.10424]	[0.09423, 0.10423]	[0.09415, 0.10415]
	min	0.1	[0.09674, 0.10674]	[0.09674, 0.10674]	[0.09684, 0.10684]	[0.09489, 0.10489]
Bad	max	0.95001	[0.94576, 0.95576]	[0.94574, 0.95574]	[0.95377, 0.96377]	[0.94658, 0.95658]
	max2	0.88747	[0.88051, 0.89051]	[0.88053, 0.89053]	[0.88325, 0.89325]	[0.88051, 0.89051]
	min	4×10^{-7}	[0, 0.00987]	[0, 0.00987]	[0.00349, 0.01287]	[0,0.005]
Deceleration	max	[0.08404, 0.08881]	[0.08643, 0.09643]	[0.08644, 0.09644]	[0.08672, 0.09672]	[0.08799, 0.09799]
	min	[0.04085, 0.04275]	[0.03969, 0.04969]	[0.03971, 0.04971]	[0.03938, 0.04938]	[0.03359, 0.04359]
Collision (Basic)	max	[0.96567, 0.97254]	[0.96775, 0.97775]	[0.96772, 0.97772]	[0.96851, 0.97851]	[0.96411, 0.97411]
	min	[0, 0.00201]	[0, 0.00987]	[0, 0.00987]	[0.00349, 0.01287]	[0,0.005]
Collision (Extended)	max	[0.35751, 0.49961]	[0.4177, 0.42783]	[0.42187, 0.43187]	[0.42345, 0.43345]	[0.42656, 0.43656]
	min	[0.04296, 0.06311]	[0.04845, 0.05845]	[0.04845, 0.05845]	[0.04254, 0.05254]	[0.04463, 0.05463]
Collision (Advanced)	max	[0.14807, 0.31121]	[0.20647, 0.21647]	[0.20646, 0.21646]	[0.20834, 0.21834]	[0.20496, 0.21496]
	min	[0.02471, 0.05191]	[0.03971, 0.04971]	[0.03971, 0.04971]	[0.03473, 0.04473]	[0.03062, 0.04062]
Anesthesia	n/a	[0.00916, 0.04222]	[0.01483, 0.02483]	[0.01493, 0.02493]	[0.01623, 0.02623]	[0.01847, 0.0284]
Confidence level c=0.99999						
Model	**Type**	***P***	CI_L	CI_{Ans}	CI_{Arc}	$Qint$
Good	max	0.1	[0.09391, 0.10391]	[0.09392, 0.10392]	[0.09405, 0.10405]	[0.09512, 0.10512]
	min	0.1	[0.09671, 0.10671]	[0.09679, 0.10679]	[0.09675, 0.10675]	[0.09525, 0.10525]
Bad	max	0.95001	[0.94545, 0.95545]	[0.94543, 0.95543]	[0.94735, 0.95735]	[0.94543, 0.95543]
	max2	0.88747	[0.88046, 0.89046]	[0.88046, 0.89046]	[0.88325, 0.89325]	[0.88052, 0.89052]
	min	4×10^{-7}	[0, 0.00992]	[0, 0.00992]	[0.00445, 0.0139]	[0,0.005]
Deceleration	max	[0.08404, 0.08881]	[0.08725, 0.09725]	[0.08726, 0.09726]	[0.08746, 0.09746]	[0.08737, 0.09735]
	min	[0.04085, 0.04275]	[0.03943, 0.04943]	[0.03944, 0.04944]	[0.039, 0.049]	[0.03377, 0.04377]
Collision (Basic)	max	[0.96567, 0.97254]	[0.96689, 0.97589]	[0.96683, 0.97583]	[0.96863, 0.97863]	[0.96462, 0.97462]
	min	[0 , 0.00201]	[0, 0.00992]	[0, 0.00992]	[0.00445, 0.0139]	[0,0.005]
Collision (Extended)	max	[0.35751, 0.49961]	[0.41774, 0.42774]	[0.41779, 0.42779]	[0.42745, 0.43745]	[0.42875, 0.43875]
	min	[0.04296, 0.06311]	[0.04745, 0.05745]	[0.04776, 0.05776]	[0.05776, 0.05673]	[0.04576, 0.05576]
Collision (Advanced)	max	[0.14807, 0.31121]	[0.20547, 0.21547]	[0.20547, 0.21547]	[0.20385, 0.21385]	[0.20453, 0.21453]
	min	[0.02471, 0.05191]	[0.03861, 0.04861]	[0.03887, 0.04887]	[0.0363, 0.04363]	[0.03031, 0.04031]
Anesthesia	n/a	[0.00916, 0.04222]	[0.01557, 0.02557]	[0.01562, 0.02562]	[0.01385, 0.02385]	[0.01852, 0.02852]

Table 5.6: Results for CI computation obtained via ProbReach, with solver precision δ=10^{-3} and interval size equal to 10^{-2}, **Type** - extremum type and $\boldsymbol{P}$ - true probability value.

Model	**Type**	$\boldsymbol{c}$	CI_B	CI_{CLT}	CI_{AC_W}	CI_W	CI_L	CI_{Ans}	CI_{Arc}	$Qint$
Good	max	0.99	24252	24025	24038	24027	24035	24034	26681	**23136**
	min	0,99	23451	23248	24256	24250	24253	24252	26894	**23245**
Bad	max	0.99	13118	12670	12841	12817	12833	12832	23006	**11726**
	max2	0.99	27498	26954	26960	26955	26958	26958	40442	**25734**
	min	0.99	2590	**96**	961	688	680	680	347	n/a
Deceleration	max	0.99	22842	22393	22673	22517	22628	22623	24365	**20318**
	min	0.99	11224	11073	11114	11086	11104	11104	11570	**9798**
Collision	max	0.99	9581	9318	9653	9463	9386	9381	10643	**8222**
(Basic)	min	0.99	2590	**96**	961	688	680	680	347	n/a
Collision	max	0.99	65109	64804	64854	64841	64932	64930	104637	**62485**
(Extended)	min	0.99	13624	13257	13486	13375	13326	13320	14737	**12869**
Collision	max	0.99	44370	43602	43645	43640	43644	43643	51734	**43524**
(Advanced)	min	0.99	9500	9081	9094	9085	9090	9089	9282	**9080**
Anesthesia	n/a	0.99	5801	4847	5024	4952	4928	4919	5522	**4804**
Good	max	0.999	39211	39187	39215	39196	39210	39200	43407	**38094**
	min	0,999	39650	39364	39401	39368	39373	39373	43848	**38204**
Bad	max	0.999	20717	20401	20550	20497	20527	20562	32006	**20322**
	max2	0.999	44557	43848	43863	43848	43855	43855	56442	**42888**
	min	0.999	3950	**107**	1549	1103	1362	1362	434	n/a
Deceleration	max	0.999	36609	36039	36061	36044	36132	36130	39524	**33068**
	min	0.999	18727	18629	18709	18671	18628	18682	19438	**16618**
Collision	max	0.999	13795	13222	13341	13286	13311	13397	15385	**13098**
(Basic)	min	0.999	3950	**107**	1549	1103	1362	1362	434	n/a
Collision	max	0.999	106252	106099	106243	106147	106224	106224	166345	**104531**
(Extended)	min	0.999	22887	21860	22196	21935	22041	22038	24742	**20862**
Collision	max	0.999	71746	70435	70646	70636	70642	70640	143390	**69642**
(Advanced)	min	0.999	15833	15679	15746	15723	15748	15746	18354	**15086**
Anesthesia	n/a	0.999	9017	8516	8827	8628	8593	8592	9284	**8430**

Table 5.7: Samples size comparison for confidence interval computation obtained via ProbReach, with solver δ precision equal to 10^{-3} and interval size equal to 10^{-2}, **Type** - extremum type and $\boldsymbol{c}$ - confidence level. Min result between all CIs results reported in **bold**.

Model	**Type**	$\boldsymbol{c}$	CI_B	CI_{CLT}	CI_{AC_W}	CI_W	CI_L	CI_{Ans}	CI_{Arc}	$Qint$
Good	max	0.9999	55187	54327	54361	54347	54355	54362	60104	**52990**
	min	0,9999	55885	55281	55231	55286	55307	55307	61631	**53411**
Bad	max	0.9999	29147	28240	28339	28276	28289	28289	42463	**27944**
	max2	0.9999	62735	61139	61364	61152	61359	61358	86442	**59012**
	min	0.9999	4849	**116**	2153	1530	2458	2458	508	n/a
Deceleration	max	0.9999	50476	50243	50277	50250	50269	50268	55495	**46084**
	min	0.9999	25741	25695	25817	25779	25794	25794	26790	**22466**
Collision	max	0.9999	19476	18907	19084	18984	19035	19032	21537	**18128**
(Basic)	min	0.9999	4849	**116**	2153	1530	2458	2458	508	n/a
Collision	max	0.9999	148388	147675	147834	147746	147786	147635	236423	**145974**
(Extended)	min	0.9999	31528	29894	30420	30023	30423	30420	34736	**28588**
Collision	max	0.9999	100592	100143	100275	100174	100196	100195	168345	**99456**
(Advanced)	min	0.9999	20497	20130	20412	20312	20384	20383	23864	**19788**
Anesthesia	n/a	0.9999	13131	11462	11683	11658	11724	11722	13948	**11288**
Good	max	0.99999	70422	69484	69582	69496	69530	69529	77262	**68456**
	min	0,99999	71898	71286	71339	71293	71321	71321	79369	**68994**
Bad	max	0.99999	37388	36518	36771	36629	36687	36868	37006	**36164**
	max2	0.99999	79306	79097	79125	79101	79118	79118	96442	**77892**
	min	0.99999	5797	**124**	2766	1963	4136	4136	572	n/a
Deceleration	max	0.99999	65248	65233	65330	65299	65320	65319	72114	**59882**
	min	0.99999	33147	32969	33133	33018	33060	33060	34231	**29096**
Collision	max	0.99999	25279	24711	24834	24789	24934	24933	26045	**23016**
(Basic)	min	0.99999	5797	**124**	2766	1963	4136	4136	572	n/a
Collision	max	0.99999	191466	190776	191253	190894	191485	191472	376294	**185456**
(Extended)	min	0.99999	41153	38942	39745	39473	39537	39541	47923	**37608**
Collision	max	0.99999	131517	129746	131185	129845	129934	129933	183405	**127486**
(Advanced)	min	0.99999	27305	25657	25835	25736	25792	25791	29362	**24569**
Anesthesia	n/a	0.99999	16197	15453	15834	15634	15734	15733	17845	**15314**

Table 5.8: Samples size comparison for confidence interval computation obtained via ProbReach, with solver δ precision equal to 10^{-3} and interval size equal to 10^{-2}, **Type** - extremum type and $\boldsymbol{c}$ - confidence level. Min result between all CIs results reported in **bold**.

5.5.2 Qint Method Results

A comparison of the Qint method's confidence intervals is also presented in Tables 5.3, 5.4, 5.5 and 5.6. All of the Qint intervals also contain single probability values and overlap with true probability intervals. The original Qint algorithm is not able to provide results for "Bad" type min and Collision Basic type min models, because for very small probability values like 4×10^{-7} and [0, 0.00201] it could not detect $n_s > 0$ for the chosen confidence levels and interval size. Due to this reason the original Qint algorithm was changed by modifying the CLT method described in Section 3.3. From the results we see that the Qint algorithm shows great potential, which is connected with the very fast convergence rate of the QMC method and with finding an appropriate partition (in terms of the parameters k, s).

Table 5.7 allows us to compare Qint's sample sizes with those of other CIs. However, CI_{CLT} had an advantage in the number of samples for small probability values near the border (see Figure 5.4), where we can clearly see that for bigger true probability values, which is presented in the tested models except "Bad" type min and Collision Basic type min, this trend is not preserved. On the contrary, Qint uses fewer number of samples than other CIs and CI_{CLT} in particular (see also Figure 5.5). For the tested models set with confidence $\boldsymbol{c}$=0.99999, Qint used on average between 1,850 and 24,802 fewer samples than other CIs techniques.

5.6 Gaussian Process Estimation Results

In my thesis I apply GP estimation techniques (see Section 2.5) based on the EP algorithm (see Subsection 2.5.5), and statistical model checking (SMC) CI estimation based on the standard Clopper-Pearson technique with standard MC sampling (see Subsection 2.4.1). In the experiments, the parameter types used in each model were:

- "Good" and "Bad": random parameter r uniformly distributed over $[\gamma, \theta]$, where γ and θ are uncertain parameters;
- Deceleration: random parameter β and two uncertain parameters μ and σ for the normal distribution of β;

- Collision basic: one random a_{d1} parameter and two uncertain parameters μ and σ for the normal distribution of a_{d1};
- Collision extended: random parameters a_{d1}, a_{d2} and four uncertain parameters μ_1, σ_1 and μ_2, σ_2 for the normal distribution of a_{d1} and of a_{d2};
- Psoriasis treatment: random parameter In_A and one uncertain parameter μ of the normal distribution of In_A.

Next, I provide a comparison of the CIs obtained via ProbReach and true probability values that are either analytically calculated or absolute (non-statistical) enclosures computed using ProbReach's formal approach, which provides rigorous guarantees.

5.6.1 Accuracy of the Expectation Propagation Method

I estimate the accuracy of the GP approach using the average CI interval size and root mean squared error (RMSE) of our estimates across all input points.

As it can be seen in Table 5.9, where the average CI size comparison $\pm$ standard deviation is presented, the GP approach shows much better results in comparison with SMC using the same number of samples. We can see that GP offers not only tighter intervals but also smaller standard deviation values for all the tested models over a different number of points and samples with 0.99 confidence. Table 5.10 also shows that with the increase of the confidence level to 0.99999 the precision of the interval is obviously decreasing, but GPs still show better results for all the values. It is important to note that the GP superiority in terms of average CI size can be quite significant with respect to SMC. For example, for the Psoriasis model (see Section 5.2) with 20 points and 200 samples GP has 0.0727 average interval size while SMC has as much as 0.4014 (see Table 5.9). For instance, in Figure 5.8 (a) it is clearly visible that even when the number of samples is quite small and equal to 20, the GP method provides thin CI bounds. It is also important to note that GP presents much smoother mean curve in comparison with SMC, which follows directly from the GP construction procedure (see Section 2.5). The same trend is retained for 100 samples (see Figure 5.8 (b)).

Model	n	*S=20*		*S=50*		*S=100*		*S=200*	
		SMC	**GP**	**SMC**	**GP**	**SMC**	**GP**	**SMC**	**GP**
Good	20	0.3999±0.0457	**0.0797**±0.0265	0.2304±0.0268	**0.0575**±0.0171	0.1591±0.0207	**0.0572**±0.0184	0.1087±0.0147	**0.0561**±0.0181
	100	0.3854±0.0452	**0.0378**±0.0127	0.2264±0.0271	**0.0314**±0.0106	0.1572±0.0192	**0.0293**±0.0097	0.1084±0.0136	**0.0283**±0.0094
	200	0.3835±0.0470	**0.0301**±0.0099	0.2263±0.0274	**0.0226**±0.0082	0.1575±0.0194	**0.0221**±0.0076	0.1083±0.0138	**0.0214**±0.0073
Bad	20	0.5431±0.0232	**0.1224**±0.0179	0.3580±0.1183	**0.1183**±0.0171	0.2536±0.0108	**0.1178**±0.0162	0.1794±0.0721	**0.1173**±0.0146
	100	0.5381±0.0297	**0.0586**±0.0150	0.3545±0.0169	**0.0585**±0.0148	0.2514±0.0128	**0.0584**±0.0145	0.1778±0.0090	**0.0581**±0.0143
	200	0.5371±0.0294	**0.0442**±0.0119	0.3543±0.0167	**0.0428**±0.0113	0.2512±0.0129	**0.0426**±0.0112	0.1776±0.0114	**0.0425**±0.0105
Deceleration	20	0.3634±0.1169	**0.0812**±0.0626	0.2206±0.0946	**0.0802**±0.0601	0.1502±0.0746	**0.0801**±0.0588	0.1023±0.0568	**0.0795**±0.0587
	100	0.3571±0.1194	**0.0468**±0.0436	0.2141±0.0968	**0.0424**±0.0366	0.1443±0.0743	**0.0411**±0.0338	0.0991±0.0556	**0.0405**±0.0332
	200	0.3563±0.1191	**0.0396**±0.0370	0.2132±0.0967	**0.0317**±0.0275	0.1441±0.0741	**0.0308**±0.0253	0.0991±0.0556	**0.0306**±0.0247
Collision (Basic)	20	0.4591±0.0749	**0.1023**±0.0390	0.3107±0.0425	**0.0994**±0.0338	0.2182±0.0300	**0.0955**±0.0323	0.1542±0.0212	**0.0944**±0.0316
	100	0.4551±0.0885	**0.0571**±0.0261	0.3069±0.0544	**0.0511**±0.0196	0.2148±0.0397	**0.0501**±0.0191	0.1518±0.0284	**0.0498**±0.0189
	200	0.4541±0.0872	**0.0425**±0.0197	0.3061±0.0554	**0.0393**±0.0154	0.2146±0.0402	**0.0363**±0.0142	0.1516±0.0287	**0.0347**±0.0130
Collision (Extended)	20	0.7456±0.0845	**0.1545**±0.0479	0.6845±0.0843	**0.1541**±0.0442	0.5375±0.0841	**0.1532**±0.0434	0.5347±0.0835	**0.1481**±0.0421
	100	0.6023±0.0824	**0.1056**±0.0336	0.5342±0.0784	**0.0942**±0.0307	0.5002±0.0756	**0.0938**±0.0302	0.4835±0.0734	**0.0936**±0.0284
	200	0.4756±0.0764	**0.0945**±0.0291	0.4385±0.0746	**0.0759**±0.0273	0.3957±0.0721	**0.0728**±0.0259	0.3528±0.0684	**0.0725**±0.0247
Psoriasis	20	0.5471±0.0179	**0.0871**±0.0162	0.4832±0.0171	**0.0734**±0.0159	0.4627±0.0165	**0.0731**±0.0157	0.4014±0.0162	**0.0727**±0.0156
	100	0.4827±0.0162	**0.0673**±0.0158	0.4173±0.0159	**0.0583**±0.0156	0.3854±0.0157	**0.0565**±0.0155	0.3365±0.0156	**0.0525**±0.0154
	200	0.4013±0.0157	**0.0495**±0.0156	0.3531±0,0156	**0.0417**±0.0154	0.3264±0.0153	**0.0384**±0.0153	0.3245±0.0153	**0.0371**±0.0152

Table 5.9: Average CI size ± standard deviation for SMC *vs.* GP, obtained via ProbReach, with solver δ precision equal to 10^{-3} and 0.99 confidence level for 10 independent runs of the experiment, **Model** - model type, $\boldsymbol{n}$ - number of points and $\boldsymbol{S}$ - number of samples per point. Min between SMC and GP results reported in **bold**.

Model	n	***S=20***		***S=50***		***S=100***		***S=200***	
		SMC	**GP**	**SMC**	**GP**	**SMC**	**GP**	**SMC**	**GP**
Good	20	0.6201±0.0486	**0.1370**±0.0450	0.3781±0.0376	**0.0992**±0.0294	0.2647±0.0324	**0.0991**±0.0216	0.1827±0.0242	**0.0970**±0.0210
	100	0.6051±0.0468	**0.0650**±0.0218	0.3726±0.0377	**0.0524**±0.0208	0.2618±0.0329	**0.0503**±0.0209	0.1825±0.0307	**0.0488**±0.0198
	200	0.6033±0.0484	**0.0517**±0.0171	0.3725±0.0383	**0.0371**±0.0134	0.2619±0.0302	**0.0365**±0.0131	0.1818±0.0226	**0.0362**±0.0126
Bad	20	0.7819±0.0279	**0.2081**±0.0464	0.5647±0.0208	**0.2013**±0.0454	0.4149±0.0175	**0.2004**±0.0453	0.2993±0.0125	**0.1996**±0.0451
	100	0.7758±0.0354	**0.1620**±0.0256	0.5595±0.0253	**0.1003**±0.0257	0.4114±0.0206	**0.1001**±0.0255	0.2967±0.0149	**0.0999**±0.0252
	200	0.7746±0.0350	**0.0758**±0.0204	0.5592±0.0251	**0.0733**±0.0193	0.4109±0.0207	**0.0730**±0.0192	0.2965±0.0150	**0.0728**±0.0192
Deceleration	20	0.5881±0.1298	**0.1453**±0.0992	0.3723±0.1263	**0.1432**±0.0949	0.2568±0.1097	**0.1428**±0.0928	0.1760±0.0883	**0.1426**±0.0825
	100	0.5822±0.1223	**0.0819**±0.0732	0.3617±0.1189	**0.0737**±0.0618	0.2482±0.1094	**0.0711**±0.0572	0.1705±0.0869	**0.0704**±0.0543
	200	0.5814±0.1320	**0.0744**±0.0798	0.3612±0.1288	**0.0549**±0.0467	0.2462±0.1091	**0.0531**±0.0431	0.1701±0.0869	**0.0528**±0.0421
Collision (Basic)	20	0.6865±0.0811	**0.1748**±0.0650	0.4945±0.0625	**0.1696**±0.0566	0.3582±0.0478	**0.1631**±0.0542	0.2576±0.0351	**0.1612**±0.0531
	100	0.6833±0.0942	**0.0980**±0.0441	0.4895±0.0785	**0.0874**±0.0333	0.4363±0.0789	**0.0861**±0.0331	0.2538±0.0468	**0.0859**±0.0321
	200	0.6838±0.0930	**0.0729**±0.0334	0.4886±0.0796	**0.0674**±0.0263	0.3527±0.0633	**0.0623**±0.0254	0.2535±0.0472	**0.0596**±0.0223
Collision (Extended)	20	0.7834±0.0984	**0.2624**±0.0776	0.6134±0.0975	**0.2622**±0.0773	0.5384±0.0846	**0.2600**±0.0712	0.4384±0.0824	**0.2513**±0.0691
	100	0.7823±0.0964	**0.1804**±0.0559	0.6128v 0.0945	**0.1734**±0.0523	0.5364±0.0824	**0.1621**±0.0491	0.4365±0.0786	**0.1599**±0.0477
	200	0.7754±0.0864	**0.1403**±0.0488	0.6084±0.0834	**0.1299**±0.0459	0.5328±0.0724	**0.1145**±0.3146	0.4325±0.0713	**0.1035**±0.2835
Psoriasis)	20	0.7867±0.0246	**0.1486**±0.0212	0.6134±0.0215	**0.1273**±0.0210	0.5633±0.0214	**0.1135**±0.0197	0.4726±0.0211	**0.1034**±0.0196
	100	0.7694±0.0214	**0.0122**±0.0189	0.5935±0.0196	**0.0118**±0.0187	0.5464±0.0193	**0.0115**±0.0185	0.4964±0.0191	**0.0113**±0.0184
	200	0.7602±0.0158	**0.0114**±0.0147	0.5424±0.0153	**0.0111**±0.0144	0.5325±0.0152	**0.0109**±0.0144	0.4867±0.0151	**0.0108**±0.0143

Table 5.10: Average CI size ± standard deviation for SMC *vs.* GP, obtained via ProbReach, with solver δ precision equal to 10^{-3} and 0.99999 confidence level for 10 independent runs of the experiment, **Model** - model type, $\boldsymbol{n}$ - number of points and $\boldsymbol{S}$ - number of samples per point. Min between SMC and GP results reported in **bold**.

Model	n	S=20		S=100		S=300	
		SMC	GP	SMC	GP	SMC	GP
Good	20	0.0864±0.005	**0.0544**±0.004	0.0573±0.003	**0.0395**±0.002	0.0418±0.002	**0.0325**±0.001
	100	0.0532±0.003	**0.0426**±0.003	0.0496±0.003	**0.0366**±0.002	0.0356±0.001	**0.0268**±0.001
	300	0.0328±0.002	**0.0284**±0.002	0.0289±0.002	**0.0258**±0.001	0.0255±0.001	**0.0208**±0.0008
Bad	20	0.0734±0.031	**0.0628**±0.025	0.0625±0.018	**0.0536**±0.018	0.0511±0.011	**0.0435**±0.011
	100	0.0657±0.024	**0.0574**±0.022	0.0553±0.015	**0.0467**±0.015	0.0385±0.008	**0.0342**±0.008
	300	0.0577±0.016	**0.0486**±0.014	0.0472±0.013	**0.0384**±0.012	0.0336±0.009	**0.0358**±0.008
Deceleration	20	0.0637±0.023	**0.0431**±0.021	0.0346±0.014	**0.0243**±0.004	0.0237±0.007	**0.0164**±0.005
	100	0.0482±0.021	**0.0364**±0.009	0.0294±0.004	**0.0189**±0.003	0.0216±0.003	**0.0113**±0.002
	300	0.0421±0.015	**0.0294**±0.011	0.0259±0.003	**0.0119**±0.001	0.0146±0.002	**0.0102**±0.002
Collision (Basic)	20	0.0815±0.014	**0.0647**±0.012	0.0619±0.011	**0.0528**±0.010	0.0428±0.008	**0.0325**±0.006
	100	0.0684±0.012	**0.0627**±0.011	0.0519±0.011	**0.0426**±0.008	0.0317±0.007	**0.0234**±0.005
	300	0.0412±0.008	**0.0386**±0.008	0.0295±0.009	**0.0254**±0.006	0.0158±0.007	**0.0128**±0.005
Collision (Extended)	20	0.0574±0.009	**0.0462**±0.008	0.0357±0.006	**0.0346**±0.004	0.0173±0.005	**0.0127**±0.003
	100	0.0453±0.008	**0.0391**±0.007	0.0276±0.005	**0.0258**±0.005	0.0147±0.003	**0.0105**±0.004
	300	0.0319±0.006	**0.0208**±0.005	0.0185±0.004	**0.0113**±0.004	0.0093±0.004	**0.0085**±0.002
Psoriasis	20	0.0756±0.017	**0.0593**±0.016	0.0584±0.014	**0.0485**±0.012	0.0388±0.011	**0.0347**±0.011
	100	0.0566±0.015	**0.0416**±0.015	0.0496±0.013	**0.0295**±0.011	0.0352±0.012	**0.0208**±0.011
	300	0.0393±0.012	**0.0317**±0.012	0.0279±0.011	**0.0218**±0.009	0.0218±0.009	**0.0159**±0.006

Table 5.11: Root-Mean-Square Error ± standard deviation for SMC *vs.* GP, obtained via ProbReach, with solver δ precision equal to 10^{-3} and 0.99 confidence level for 10 independent runs of the experiment, **Model** - model type, $\boldsymbol{n}$ - number of points and $\boldsymbol{S}$ - number of samples per point. Min between SMC and GP results reported in **bold**.

The comparison of RMSE ± standard deviation (see Table 5.11) shows that in all cases GPs are more accurate than SMC. In this table, the true probability values used to compute the RMSE for the Good and Bad models (see Section 5.2) were calculated analytically, while for all the other models 10,000 SMC simulations were used. We can also note that the RMSE for both GP and SMC approaches do not depend on the confidence level because that is used only to construct CI bounds, so for all confidence levels the result of RMSE depends only on the number of points, samples, and parameters of the chosen model.

Figure 5.9 also provides very interesting results with respect to the number of samples. The true probability function values for these figures were obtained analytically. The presented GP mean shows slightly different behaviour and probability results for the 20 samples case (see Figure 5.9 (a)). The GP CI does not include the true probability function for σ values approximately in $[1.6, 2.0]$. However, when the number of samples needed for the computation grows to 100 (see

Figure 5.9 (b)), the GP approach shows much better results and the 99% GP CIs fully include the true probability function. A similar behaviour is illustrated in Figures 5.10 and 5.11, where the true probability function is not available so we computed rigorous enclosures (boxes) with ProbReach's formal approach. We can see that the GP CIs intersect with all the probability enclosures for 100 samples (see Figure 5.10 (b)) but not for 20 samples (Figure 5.10 (a)).

These results also hold with the increase of the number of parameters, *e.g.*, a two-parameter case comparison. Figures 5.14 and 5.15 show 3D a comparison of SMC (see Figure 5.14 (a)) and GP (see Figure 5.14 (b)) CIs constructed for 20 samples and SMC (see Figure 5.15 (a)) and GP (see Figure 5.14 (b)) CIs constructed for 100 samples from the Deceleration model (see Section 5.2). We can clearly see an advantage of the GP method for both cases. At the same time it is also visible that the CI borders contract with the increase of the number of samples as for SMC approach (from 20 samples 5.14 (a) to 100 samples 5.15 (a)) as well for GP approach from 20 samples 5.14 (b) to 100 samples (b)). The same trend is preserved on the other models, including the Bad model, whose results are presented in Figures 5.16 and 5.17. An interesting fact here is that CI shrink with respect to the number of samples faster for SMC method is much higher than for the GP approach. It can be seen that CI borders contract much quickly for SMC approach (from 20 samples 5.16 (a) to 100 samples 5.17 (a)) then for GP approach from 20 samples 5.17 (b) to 100 samples (b).

It is also important to note that there are cases where the true probability function is not available and probability enclosures can not be computed via the formal approach as well even for the 2D case because of the complexity of the models. For my chosen models this problem presents itself for Collision model type extended (see Figure 5.12) and Psoriasis model (see Figure 5.13).

Hence, we conclude that GP estimation with EP is generally very accurate and more accurate than Clopper-Pearson SMC, and thus it can be used for the verification of SnPHS.

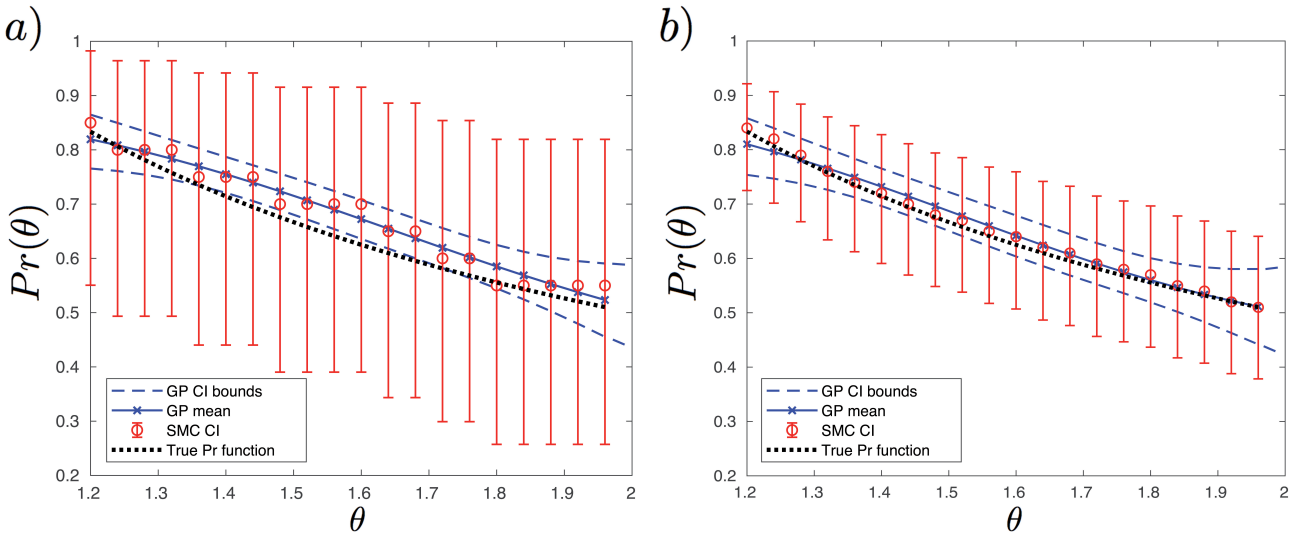


Figure 5.8: True probability function, GP and SMC 0.99 confidence CI comparison with respect to one uncertain parameter θ for 20 points and a) 20 samples and b) 100 samples per point. Model: Bad.

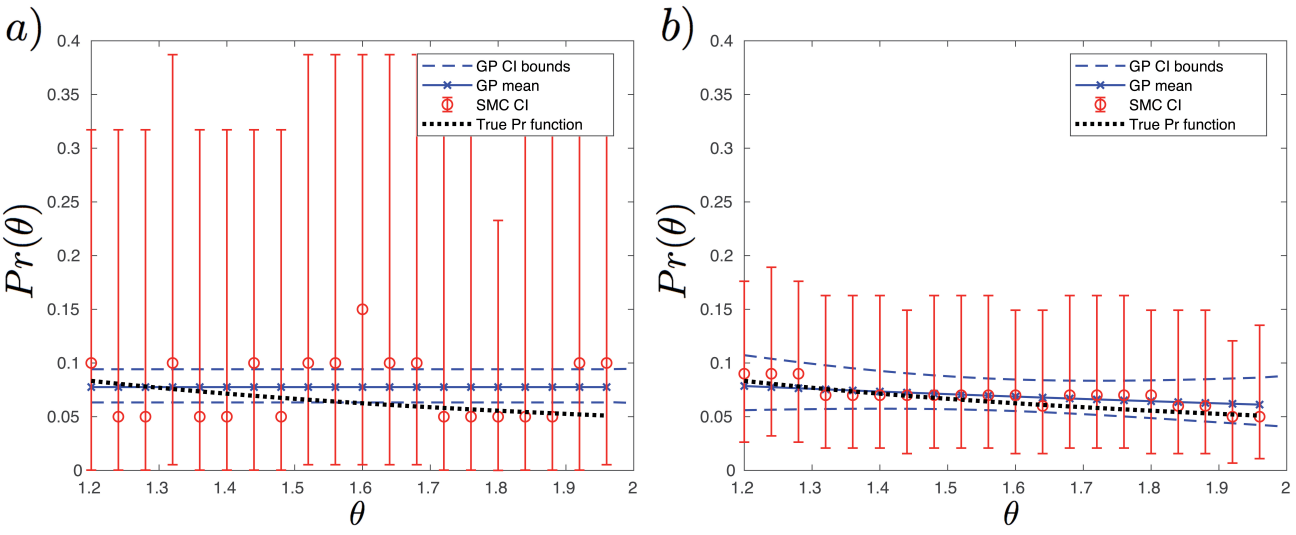


Figure 5.9: True probability function, GP and SMC 0.99 confidence CI comparison with respect to one uncertain parameter θ for 20 points and a) 20 samples and b) 100 samples per point. Model: Good.

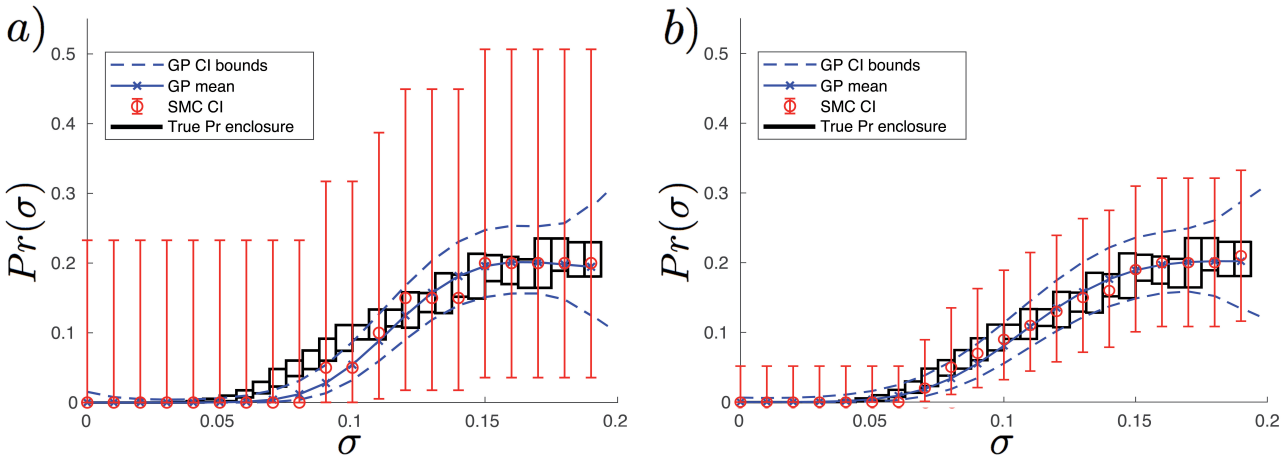


Figure 5.10: Formal enclosure, GP and SMC 0.99 confidence CI comparison with respect to one uncertain parameter σ for 20 points and a) 20 samples and b) 100 samples per point. Model: Deceleration.

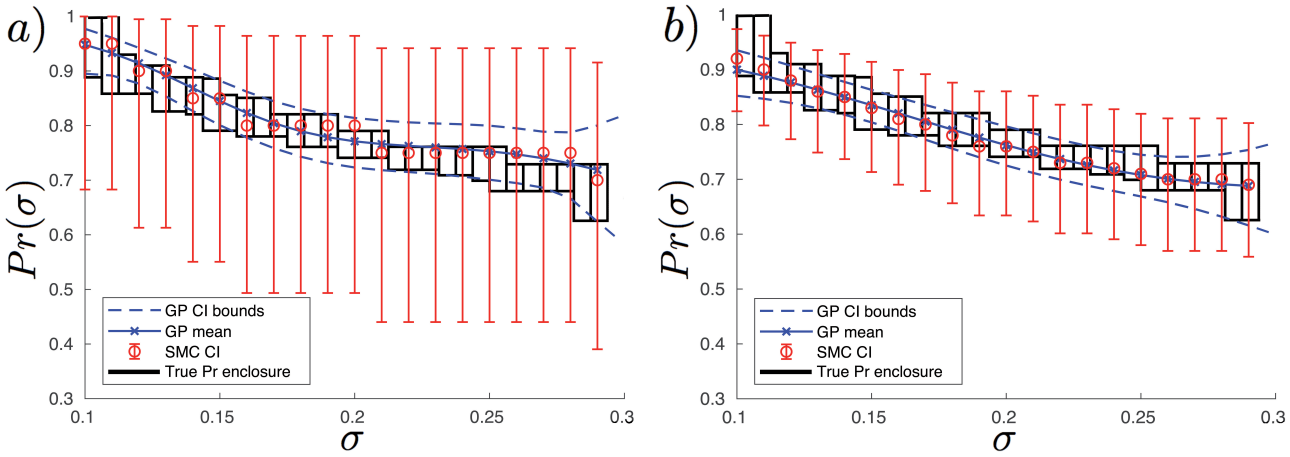


Figure 5.11: Formal enclosure, GP and SMC 0.99 confidence CI comparison with respect to one uncertain parameter σ for 20 points and a) 20 samples and b) 100 samples per point. Model: Collision, type: basic.

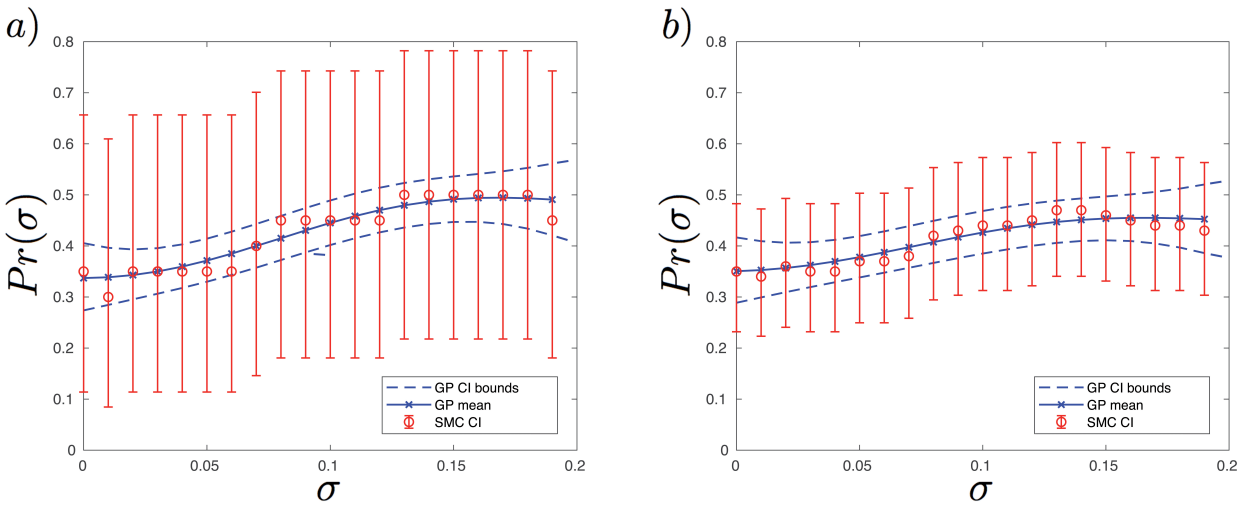


Figure 5.12: Formal enclosure, GP and SMC 0.99 confidence CI comparison with respect to one uncertain parameter σ for 20 points and a) 20 samples and b) 100 samples per point. Model: Collision, type: extended.

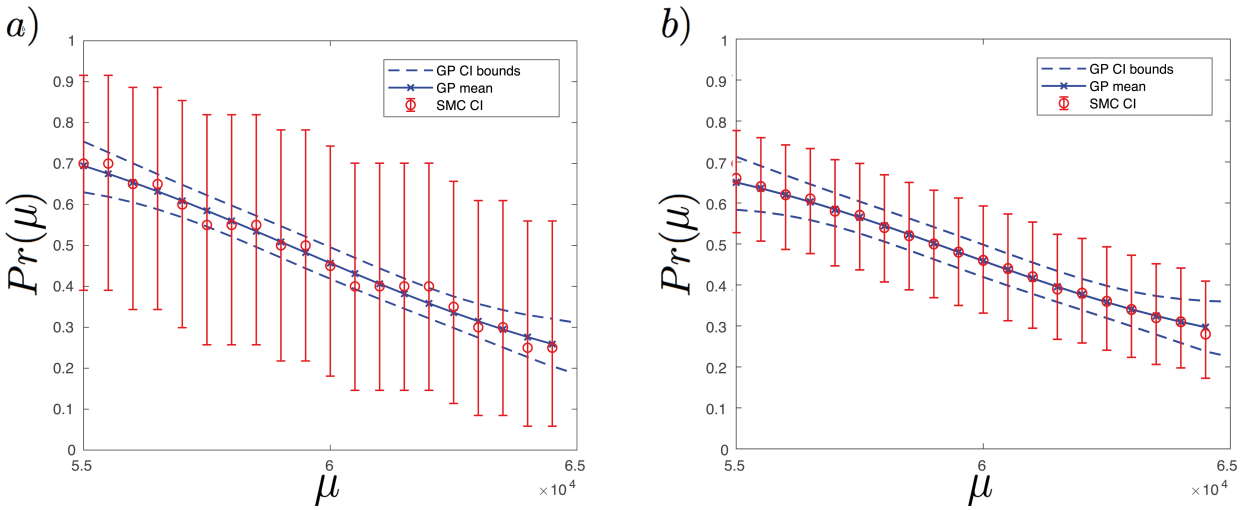


Figure 5.13: Formal enclosure, GP and SMC 0.99 confidence CI comparison with respect to one uncertain parameter σ for 20 points and a) 20 samples and b) 100 samples per point. Model: Psoriasis.

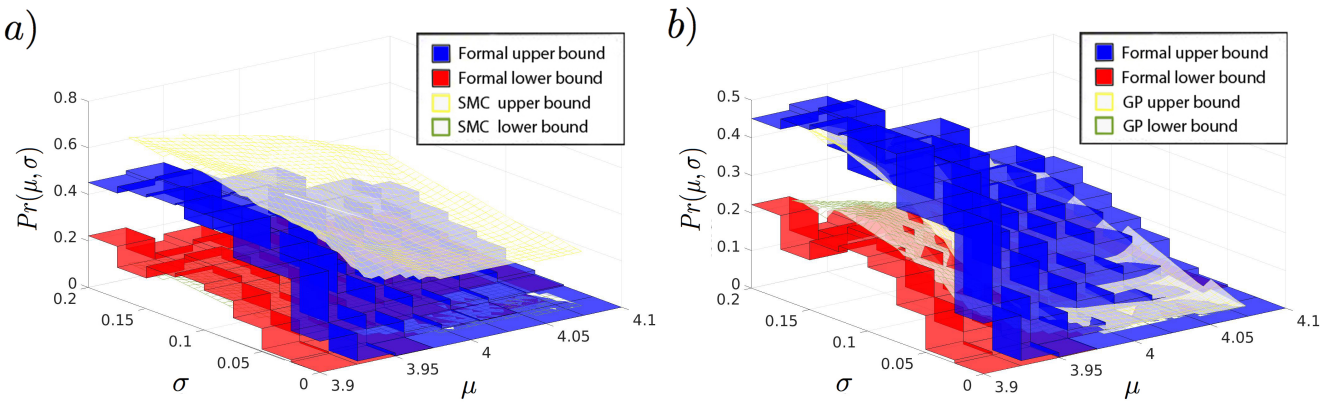


Figure 5.14: a) SMC and b) GP CI and formal bounds comparison with respect to two uncertain parameters: μ and σ for 20 points and 20 samples per point with 0.99 confidence. Model: Deceleration.

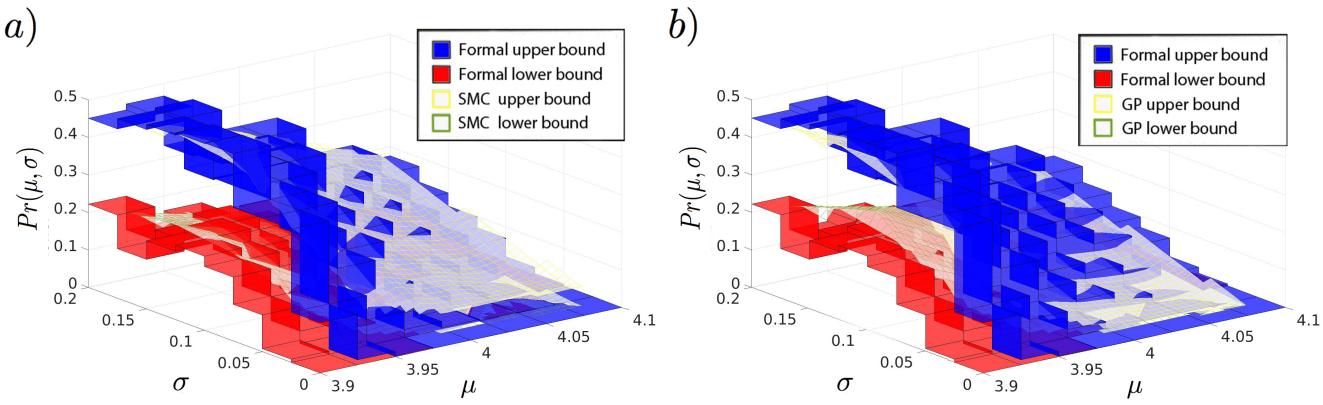


Figure 5.15: a) SMC and b) GP CI and formal bounds comparison with respect to two uncertain parameters: μ and σ for 20 points and 100 samples per point with 0.99 confidence. Model: Deceleration.

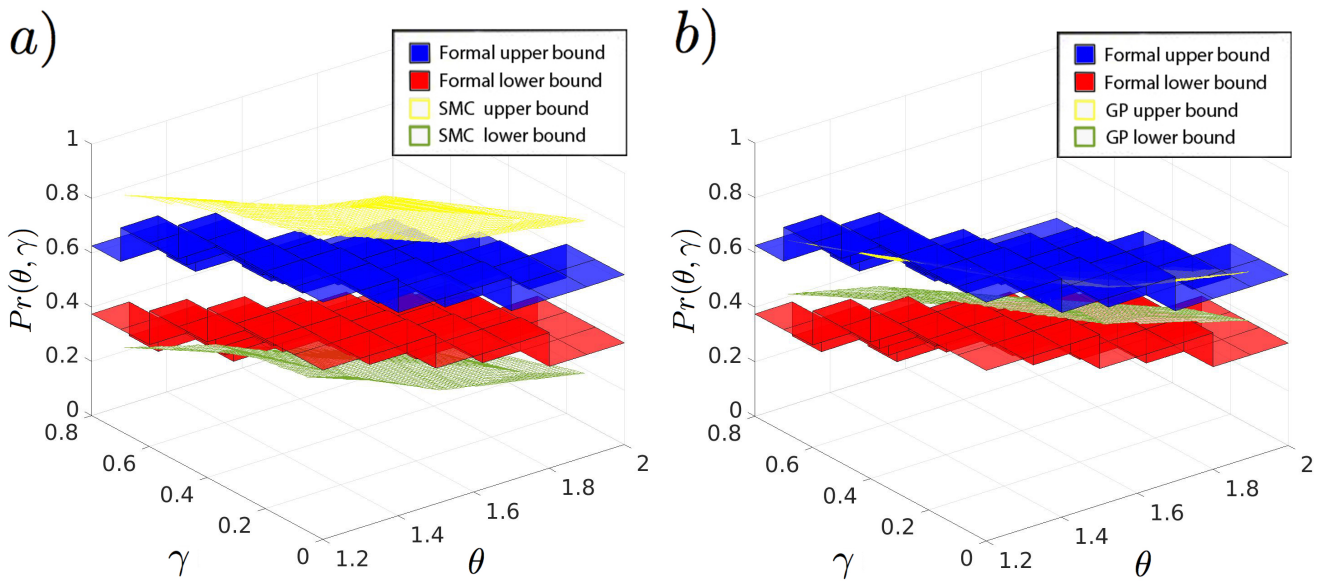


Figure 5.16: a) SMC and b) GP CI and formal bounds comparison with respect to two uncertain parameters: μ and σ for 20 points and 20 samples per point with 0.99 confidence. Model: Bad.

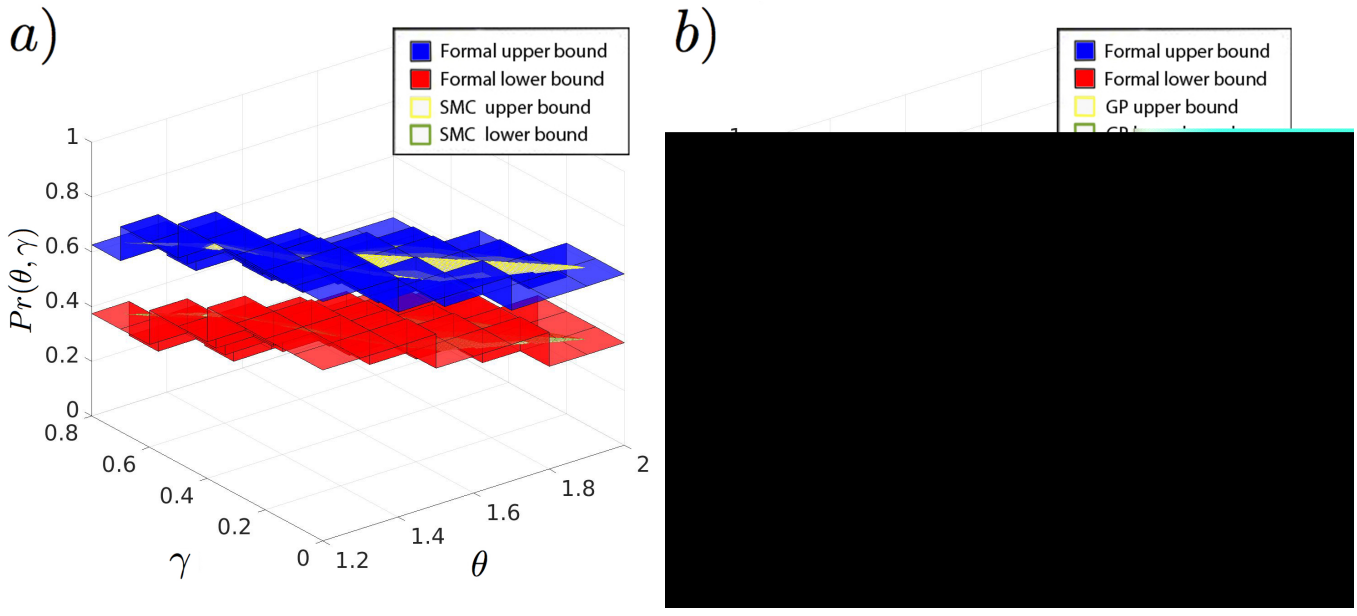


Figure 5.17: a) SMC and b) GP CI and formal bounds comparison with respect to two uncertain parameters: μ and σ for 20 points and 100 samples per point with 0.99 confidence. Model: Bad.

5.6.2 Cost of the Expectation Propagation Method

An important aspect to consider is the CPU time cost of the GP and SMC techniques. I measured CPU time in seconds, and I also compared the sample size needed by both techniques to produce results of similar accuracy.

As it was described in Section 2.5, the GP method has a significant advantage after the training process, because all the subsequent test inputs (with different points' values) can be calculated using a simple regression process, which is relatively fast and does not require further sampling. This feature reveals also one of the most important advantages of the GP method - construction of a smooth probability function, while SMC provides only a pointwise approximation.

Model	***n***	***S=20***		***S=50***		***S=100***		***S=200***	
		SMC	**GP**	**SMC**	**GP**	**SMC**	**GP**	**SMC**	**GP**
Good	20+20	**24**	25	33	**31**	41	**35**	71	**51**
	100+100	**109**	137	**151**	159	196	**180**	360	**256**
	200+200	**221**	459	**321**	571	**429**	681	**708**	823
Bad	20+20	28	**27**	37	**36**	75	**57**	140	**87**
	100+100	**126**	148	181	**164**	394	**260**	690	**439**
	200+200	**224**	463	**363**	600	**736**	745	1,254	**1,043**
Deceleration	20+20	30	**28**	41	**36**	73	**59**	183	**95**
	100+100	124	**121**	250	**184**	460	**293**	817	**476**
	200+200	**234**	474	**415**	613	892	**806**	1,496	**1,139**
Collision (Basic)	20+20	259	**140**	394	**229**	1,070	**518**	1,721	**989**
	100+100	1,322	**692**	2,181	**1,126**	5,304	**2,791**	8,902	**5,047**
	200+200	2,491	**1,497**	3,924	**2,481**	10,058	**5,817**	17,244	**11,328**
Collision (Extended)	20+20	328	**179**	834	**428**	1,286	**674**	3,804	**1,255**
	100+100	1,682	**896**	4,108	**2,112**	6,311	**3,307**	1,696	**6,284**
	200+200	3,246	**1,931**	8,319	**4,470**	13,305	**7,138**	23,133	**12,760**
Psoriasis	20+20	4,628	**2,396**	10,027	**5,577**	20,168	**11,473**	39,208	**19,808**
	100+100	23,879	**11,973**	53,190	**27,890**	118,350	**57,398**	172,908	**98,145**
	200+200	46,284	**24,371**	114,326	**56,150**	237,364	**115,065**	391,879	**198,075**

Table 5.12: Total CPU time (sec) for SMC and GP. This table shows time needed to construct a CI by GP (one training and two testing) and by SMC for the first $\boldsymbol{n}$ randomly chosen points + second randomly $\boldsymbol{n}$ chosen points. **Model** - model type, $\boldsymbol{n}$ - number of points and $\boldsymbol{S}$ - number of samples per point. All values were obtained via 10 independent runs of the experiment. Min between SMC and GP results reported in **bold**.

Table 5.12 demonstrates the total CPU time in seconds needed to construct CIs by GP and SMC for the first n randomly chosen points (training) + second n randomly chosen points (testing). As it was noted above, the GP testing process is relatively fast, however, it takes time to recompute all the covariances according to the new testing points input, while for SMC almost all the time

Model	n	S=20		S=50		S=100		S=200	
		SMC	**GP**	**SMC**	**GP**	**SMC**	**GP**	**SMC**	**GP**
Good	20	64 (7242)	**16** (400)	104 (13,663)	**23** (1,000)	119 (13,746)	**27** (2,000)	131 (13,868)	**40** (4,000)
	100	1,083 (127,442)	**106** (2,000)	1,108 (137,585)	**125** (5,000)	1,147 (138,773)	**131** (10,000)	1,215 (139,943)	**220** (20,000)
	200	2,644 (316,464)	**310** (4,000)	2,717 (328,419)	**369** (10,000)	2,757 (329,837)	**413** (20,000)	2,802 (330,494)	**570** (40,000)
Bad	20	160 (9,061)	**18** (400)	183 (9,729)	**26** (1,000)	197 (10,405)	**44** (2,000)	223 (11,266)	**78** (4,000)
	100	3,320 (194,455)	**114** (2,000)	3,374 (195,834)	**147** (5,000)	3,364 (196,128)	**243** (10,000)	2,837 (196,835)	**410** (20,000)
	200	8,868 (566,835)	**328** (4,000)	9,093 (578,673)	**392** (10,000)	9,230 (586,837)	**584** (20,000)	9,317 (592,844)	**927** (40,000)
Deceleration	20	222 (11,105)	**21** (400)	239 (11,426)	**30** (1,000)	270 (11,648)	**51** (2,000)	286 (12,285)	**90** (4,000)
	100	4,771 (238,485)	**117** (2,000)	4,965 (241,635)	**162** (5,000)	5,084 (248,551)	**277** (10,000)	5,118 (248,734)	**457** (20,000)
	200	19,536 (964,611)	**338** (4,000)	20,540 (982,463)	**432** (10,000)	20,606 (1,000,683)	**642** (20,000)	21,823 (1,034,434)	**1,033** (40,000)
Collision (Basic)	20	2,745 (10,266)	**133** (400)	2,851 (10,941)	**220** (1,000)	2,974 (11,845)	**548** (2,000)	16,666 (12,283)	**992** (4,000)
	100	70,236 (264,727)	**678** (2,000)	70,482 (269,433)	**1,134** (5,000)	71,094 (274,862)	**2,783** (10,000)	71,190 (275,423)	**4,996** (20,000)
	200	145,688 (569,809)	**1,451** (4,000)	146,354 (576,294)	**2,341** (10,000)	147,410 (582,828)	**5,620** (20,000)	149,061 (587,483)	**10,019** (40,000)
Collision (Extended)	20	7,232 (5,641)	**170** (400)	7,280 (5,692)	**417** (1,000)	7,352 (5,767)	**670** (2,000)	7,402 (6,144)	**1,251** (4,000)
	100	43,600 (59,193)	**883** (2,000)	47,044 (74,314)	**2,126** (5,000)	50,211 (74,563)	**3,361** (10,000)	51,148 (75,125)	**6,257** (20,000)
	200	105,122 (147,606)	**1,851** (4,000)	118,680 (226,846)	**4,331** (10,000)	123,520 (246,822)	**6,850** (20,000)	126,672 (249,474)	**12,604** (40,000)
Psoriasis	20	10,527 (17,879)	**2,389** (400)	24,651 (24,781)	**5,568** (1,000)	29,364 (25,184)	**11,460** (2,000)	36,071 (25,385)	**14,782** (4,000)
	100	62,184 (147,323)	**11,824** (2,000)	68,136 (195,674)	**27,876** (5,000)	69,045 (22,784)	**35,196** (10,000)	75,931 (241,846)	**48,926** (20,000)
	200	162,406 (544,884)	**23,965** (4,000)	191,761 (632,116)	**55,821** (10,000)	243,504 (634,808)	**114,713** (20,000)	284,397 (637,466)	**197,933** (40,000)

Table 5.13: CPU time in seconds and total number of samples (written in parentheses) comparison for confidence interval match between SMC and GP. This table presents time and samples needed for SMC to provide the same interval width per every point like GP; and GP CPU time and samples according to the $\boldsymbol{n}$ and $\boldsymbol{S}$ table values, **Model** - model type, $\boldsymbol{n}$ - number of points and $\boldsymbol{S}$ - number of samples per point. All values were obtained via 10 independent runs of the experiment. Min between SMC and GP results reported in **bold**.

is spent on sampling. That is why for “lightweight” models (“Good”, “Bad” and Deceleration models presented in Section 5.2, GPs show worse results for small number of points and samples. For example, the “Bad" model shows that SMC has an advantage for 100+100 points over 20 samples and 200+200 points over 20, 50 and 100 samples, but by increasing the number of samples this trend does not hold and GPs compute results faster.

The advantage of the GP method in total CPU time also depends on the probability values obtained from a certain model for a chosen goal set. Even if we deal with “lightweight” models, which could however returns an average probability values around 0.5, the GP approach can show better results because SMC will require longer probability computation.

At the same time it can be seen that with the increase of the number of testing sessions from two, as in Table 5.12, to 3 and more the advantage of GP will be more and more dramatic. This time will be spent only on GP testing process, based on the initial training set obtained from the very beginning, while SMC will engage the probability evaluation procedure again and again, which requires much more computation costs.

In Table 5.13 I show CPU time and number of samples needed for SMC to provide the GP interval width with 99% confidence per every point in accordance with the number of points and samples. It can be seen that SMC requires much longer CPU time and higher number of samples to achieve the CI results of GP. In this case, even lightweight models cannot prevent a clear advantage of the GP method.

However, with the increase of the number of observations for all models the CPU time difference decreases and the SMC method starts to narrow the gap. It can be especially seen for the larger number of points (*e.g.* for 100 points). For example, for the Psoriasis model the CPU computation time for 100 points and 20 samples, the difference ratio between SMC and GP is 5.25, while for the CPU computation time for 200 points and 200 samples the difference ratio is 1.55 (see Table 5.13).

In general, for all the tested models the GP approach spends between 2-56 times less CPU time for calculating results, it needs between 3-135 times less number of samples and it shows between 2-67 times smaller confidence interval

average size, depending on the confidence level. Therefore, it is possible to highlight a very important advantage of the GP method in both CPU time and the number of samples with respect to SMC.

5.7 Combined Approach Results

In this section, I show an application of the new combined approach, consisting of the formal approach, based on formal reasoning which returns results with absolute numerical guarantees, and the GP regression method, which offers statistical guarantees only (see Section 3.5).

In particular, I compare the combined approach with the GPEP method (see Subsection 2.5.5), whose results have been presented in Section 5.6. Like the GPEP method, where we obtain CIs for the whole parameter's domain, simple GP regression also provides CIs not only for the chosen parameter's testing points but also for the full parameter's range. The training points dataset for the combined approach produced by the formal approach which returns interval enclosures for a selection of points in the nondeterministic parameters range (see Subsection 3.5.1). GP then simply considers two rows of upper and lower points of these intervals as input training data and constructs two regression approximations (latent **Pr** function). A GP regression output provides estimated mean approximation and two CIs (lower and upper) per every function. As it was discussed in Section 3.5 the final CI returned by the combined approach is formed by the upper bound of the CI of the upper latent **Pr** function and lower bound of the CI of the lower latent **Pr** function. In this section I discuss how probability enclosures size obtained from the formal approach may affect the formation of initial data for GP training and the final size of confidence intervals returned by the combined approach.

In Subsection 5.7.1 I also consider the accuracy of the GPEP approach with respect to the combined approach and formal probability enclosures, which as it was mentioned above returns probability intervals with absolute numerical guarantees. The results in this Section are based on the Deceleration model (see Section 5.2), a distinctive feature of which is the difficulty of constructing an analytically calculated function in view of the complexity of the model.

In Subsection 5.7.2 I discuss the efficiency of the combined approach in terms of CPU time costs. As discussed in Section 3.5, the formal approach could spend a lot of CPU time on the computation of probability boxes to cover all the parameter's space, while the combined approach does not need to have so much information and does not need to spend time on extra computation. That is why I instead compute probability enclosure intervals for certain points, chosen over the parameter's space (see Figure 3.3). As well as for the GPEP method these points were chosen by using the QMC method.

One of the most interesting features considered in this Section is the ability of the combined approach to outperform the GPEP approach in terms of total computation CPU time with respect to the number of samples of GPEP and probability enclosures precision of the formal part of the combined approach. In the combined approach we deal with GP regression (see Subsection 2.5.1), which is computed on the basis of information obtained from the formal data. Obviously this fact doubles CPU time on GP regression computation as we deal with two input points' rows - one for the upper formal interval points and the other one for the lower formal interval points (see Figure 3.3).

The efficacy of the combined approach, as well as GPEP approach with respect to the classical formal and SMC methods, is based on the major advantage of GP regression. It allows us to provide information about CIs that include the true probability function over the whole model parameter's domain. At the same time, in this Section I raise a problem of finding a trade-off between accuracy and computational costs for the combined method and discuss the question of choosing the most efficient method between GPEP and combined approaches.

5.7.1 Accuracy of the Combined Method

I estimate the accuracy of the GP approach using the average CI size across all input points. In particular, the results presented in this Section are based on 20 input points of the σ parameter for the Deceleration model.

I compared two versions of the Combined approach (with formal probability enclosures precision is equal to 0.1 and 0.001) with GPEP approach with different number of samples obtained per every point. As it can be seen in Table 5.14,

where the average CI size comparison is presented. The combined approach benefits in 50% of the cases. However, I need to highlight that the decisive factor affecting the CI size, in this case, is the precision of the formal part of the combined approach. Moreover, the results for 0.001, shown in Table 5.14 serve as a border line in the sense of the combined approach advantage. It means that with the increase of the precision level the combined approach outperforms the GPEP approach in full for all the numbers of samples per point in the provided Table.

Here we can raise a question of finding a number of samples which allow us to receive the same size of CI for GPEP as for the combined approach. However, it can be concluded intuitively that for example in the case of 0.001 precision (see Table 5.14) this number will be about 8,000-10,000 samples according to the decreasing combined approach advantage in CI size from 0.07446 for 20 samples to 0.04698 for 3,000 samples. The ineffectiveness of computing such a big number of samples per input point for GPEP approach will be discussed in the next Subsection. At the same time for the small precision values like 0.1 (see Table 5.14) the advantage of the GPEP method also grows with the increase in the number of samples.

It is also important to note that the GP methods superiority in terms of average CI size is quite significant with respect to all SMC methods. This becomes apparent on the basis of the fact that the combined approach shows approximately equal CIs compared to GPEP approach with different sample sizes, while GPEP approach, as discussed in Section 5.6 is superior to all SMC methods. For example, for the Deceleration model (see Section 5.2) with 20 points and 50 samples, GPEP has 0.0802 average interval size, SMC has as much as 0.2206 (see Table 5.9), while the combined method provides 0.15871 $(0.0802 + 0.07782)$ CI size for 0.1 precision and 0.00673 $(0.0802 - 0.07347)$ CI size for 0.001 precision (see Table 5.14).

Another important aspect in terms of accuracy of the combined and GPEP approaches is the placing of the mean lines and CIs borders with respect to the true probability formal enclosures. For the chosen Deceleration model (see Section 5.2) the formal approach can return arbitrarily tight enclosures because this model features one continuous random parameter, which depends on the only

Combined Approach Precision	GPEP					
	S=20	***S=50***	***S=100***	***S=200***	***S=1000***	***S=3000***
0.1	-0.07683	-0.07782	-0.07794	-0.07851	-0.08674	-0.10431
0.001	**0.07446**	**0.07347**	**0.07335**	**0.07278**	**0.06455**	**0.04698**

Table 5.14: Average CI size difference between GPEP and Combined approaches (GPEP - Combined), obtained for 20 points with 0.99 confidence and $\boldsymbol{S}$ - number of samples per point for the Deceleration model. Results reported in **bold** show advantage for the combined approach.

one nondeterministic parameter (σ). As it was discussed earlier, it is possible to specify manually a particular precision, which we want to obtain. In other words, the mean value of the combined approach is represented by the center of the probability enclosures obtained from the formal part, while for GPEP method the mean function is represented by the regression process on the training data.

In Figure 5.18 (a) it is shown that the GPEP mean for 20 points and 20 samples (shown in black solid line) does not fit in between the two GP regression (GPR) mean lines for upper (shown in green solid line) and lower bound (shown in red solid line) constructed using the formal enclosures data with precision 0.1. However, when using 100 samples as presented in Figure 5.18 (b) the GPEP mean is within the GPR lines remains true for the 1,000 samples and 3,000 samples per point (see Figures 5.19 (a) and (b)). We can see that the increase in the number of samples helps GPEP to fit in between the two GPR lines.

The comparison of Figures 5.18 (a),(b) and 5.19 (a),(b) also shows that the CI borders of the GPEP method (shown in black dashed line) also change their shape with the increase of the number of samples. Starting with larger interval sizes around $\sigma = 0.15$ for 20 samples, which lies outside of the GPR means (see Figure 5.18 (a)) GPEP gradually changes its location and reduces the CI size with the increase of the number of samples. So we can see in Figure 5.19 (b) that around $\sigma = 0.15$ parameter's value not only the GPEP mean is included in between GPR means but also almost all CI is lying within the GPR means.

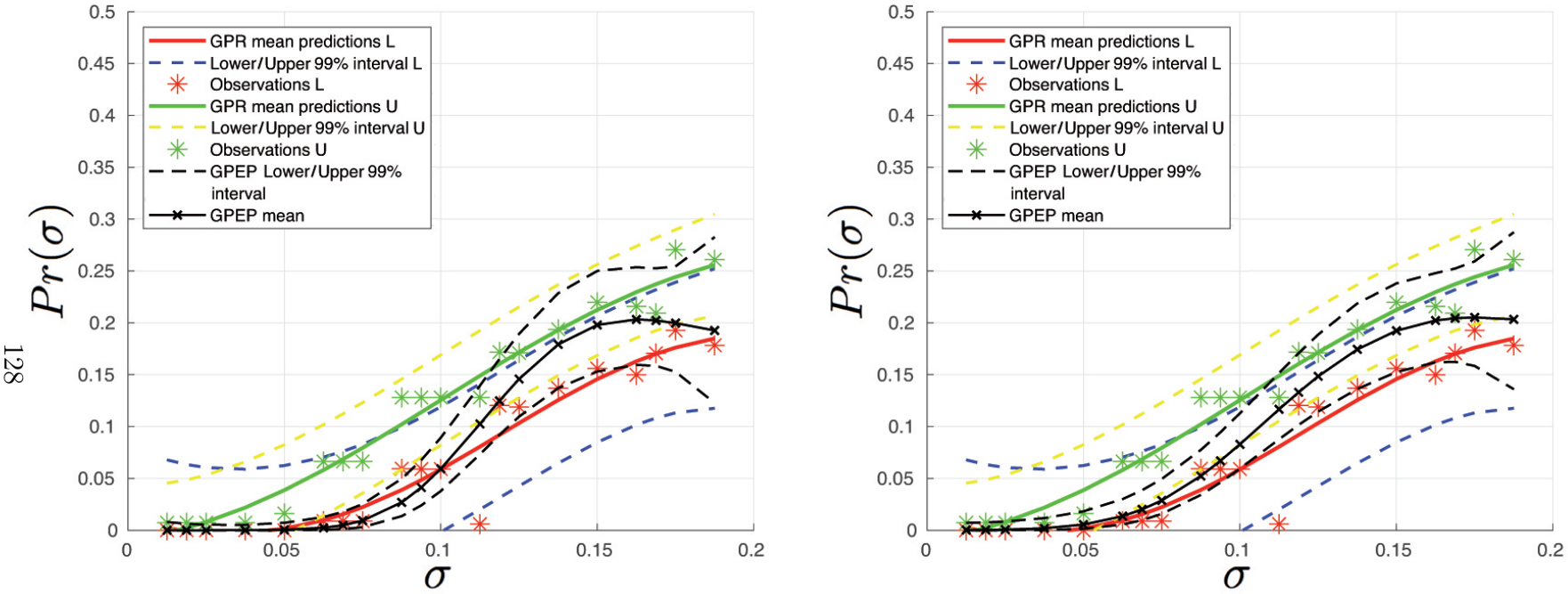
An interesting feature of the Combined approach, which can be noted for the results with 0.1 precision (see Figures 5.18 (a),(b) and 5.19 (a),(b)) is that the obtained GPR means are presented by almost flat lines, which as we will see later does very well reflect the actual latent true probability function shape.

In Figure 5.20 (a) it is shown that GPEP mean for 20 points and 20 samples (shown in black solid line) also does not fit in between the two GP regression (GPR) mean lines for upper (shown in green solid line) and lower bound (shown in ref solid line) constructed using the formal enclosures data with precision 0.001. However, with 100 samples as presented in Figure 5.20 (b) this fact does not change at all and remains the same for the 1,000 samples and 3,000 samples per point case (see Figures 5.22 (a) and (b)). We can see that the increase in the number of samples now does not help GPEP to fit in between the two GPR lines. This fact simply follows from the very small size (0.001) of the probability enclosures returned from the formal part.

Figures 5.20 (a),(b) and 5.22 (a),(b) represent the GPEP mean shape changes with the increase of the number of samples. Starting far from the combined approach means especially for σ between 0.08 and 0.15 for 20 samples (see Figure 5.20 (a)) the GPEP mean gradually changes its location and almost follows the combined approach mean with the increase of the number of samples. We can see in Figure 5.22 (b) that for σ between 0.00 and 0.15 and 3,000 samples the GPEP mean is very close to the around GPR means but unfortunately still does not lie inside the GPR lines.

We now consider the question of the GPEP mean placing a bit closer on the example of the same Figures 5.20 (a),(b) and 5.22 (a),(b). Figures 5.21 (a),(b) and 5.23 (a),(b) zooming in around σ=0.1 point with the same formal precision of 0.001 and number of samples from 20 to 3,000. It is interesting to note how the GPEP mean and GPEP CI border change with respect to the GPR means. In Figure 5.21 (a) the GPEP mean for 20 samples lies very far away from the actual combined approach means. We can see that even GPEP upper border crosses the combined approach means. For the 100 samples case represented in Figure 5.21 (b) the GPEP mean is much closer to the GPR means and the upper GPEP CI bound now lies above the GPR means and GPR CI lines. With the increase of the number of samples to 1,000 and 3,000 (see Figures 5.23 (a) and (b)) the GPEP means move closer to the GPR means and the GPEP CI bounds are almost parallel to the GPR CI bounds. However, the GPEP mean still lies outside of the GPR means and even outside the GPR CI bound.

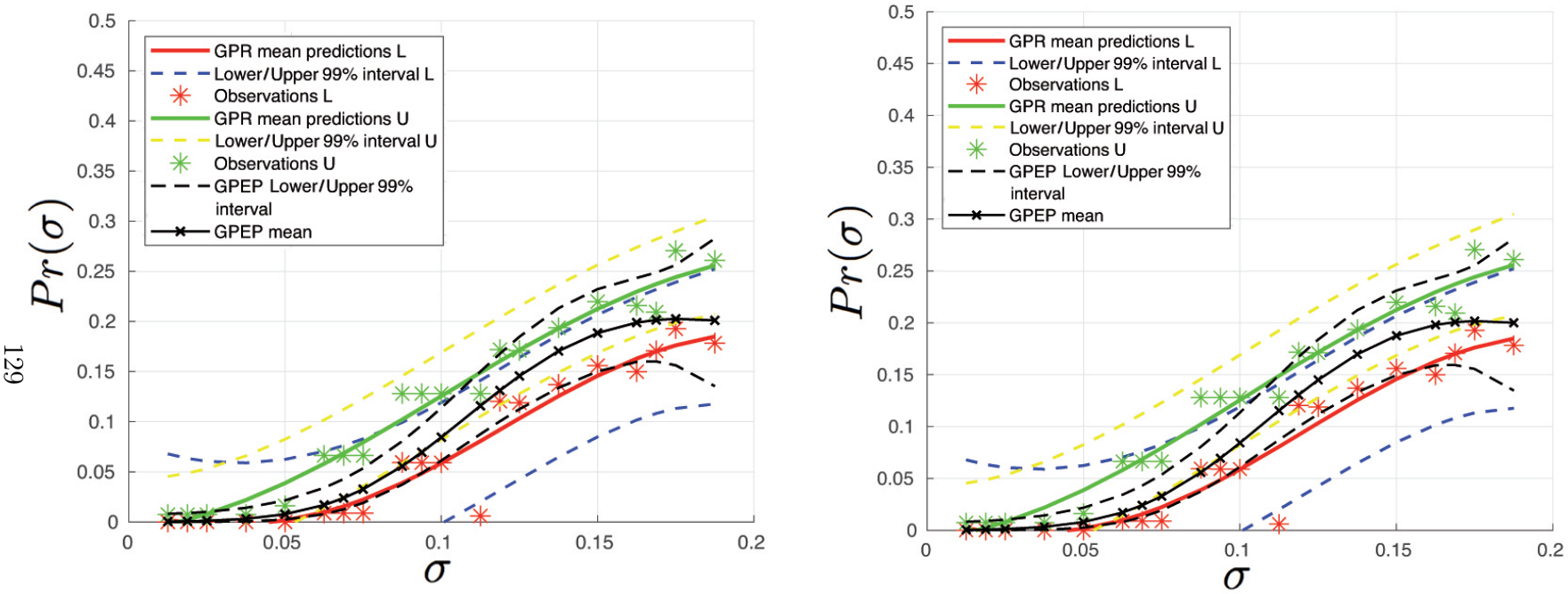
The latter facts show a great accuracy advantage of the combined method



(a) GPEP 20 samples

(b) GPEP 100 samples

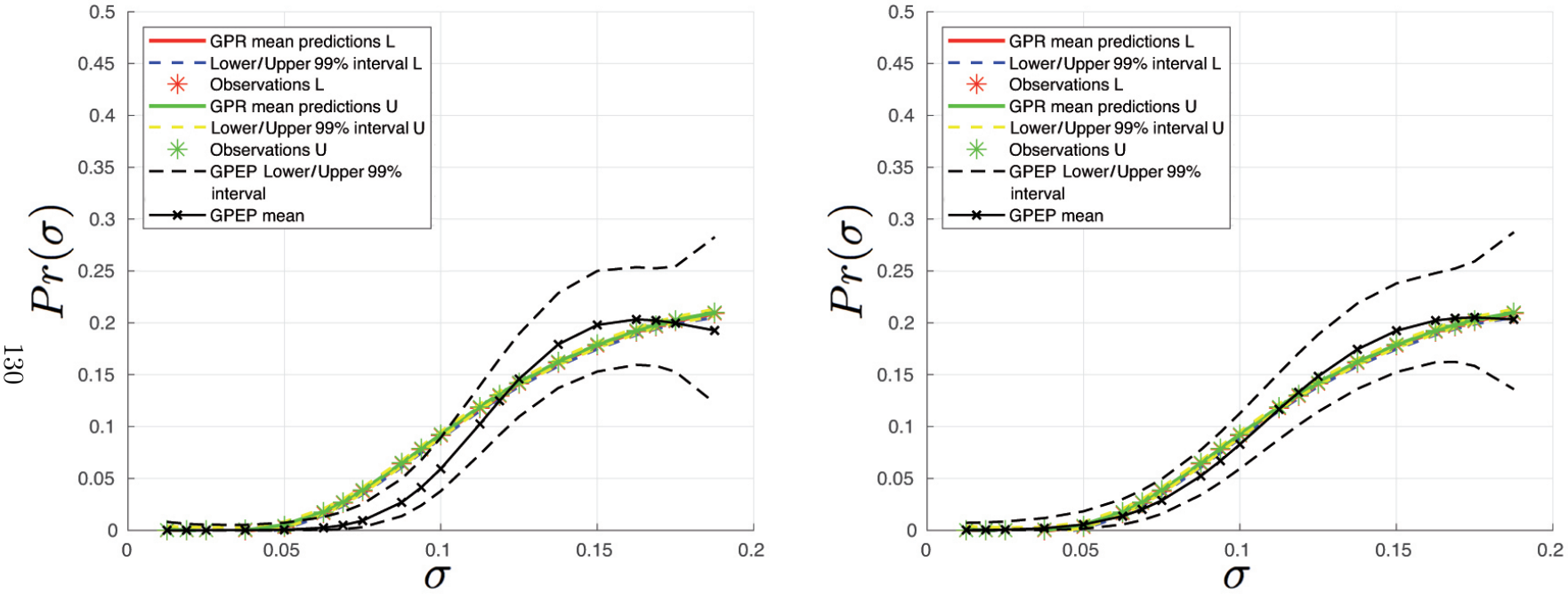
Figure 5.18: Combined (on the basis of formal approach with 0.1 precision) and GPEP approaches 0.99 confidence CIs comparison with respect to one uncertain parameter σ for 20 points.



(a) GPEP 1000 samples

(b) GPEP 3000 samples

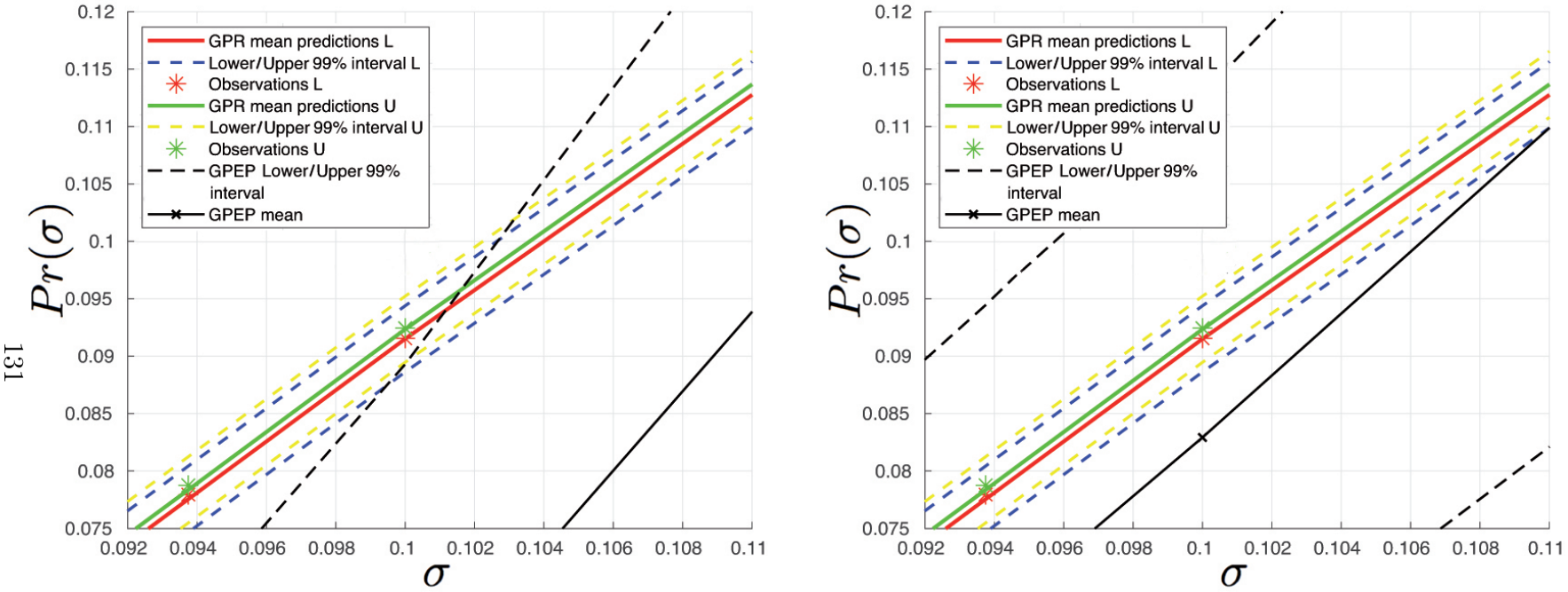
Figure 5.19: Combined (on the basis of formal approach with 0.1 precision) and GPEP approaches 0.99 confidence CIs comparison with respect to one uncertain parameter σ for 20 points.



(a) GPEP 20 samples

(b) GPEP 100 samples

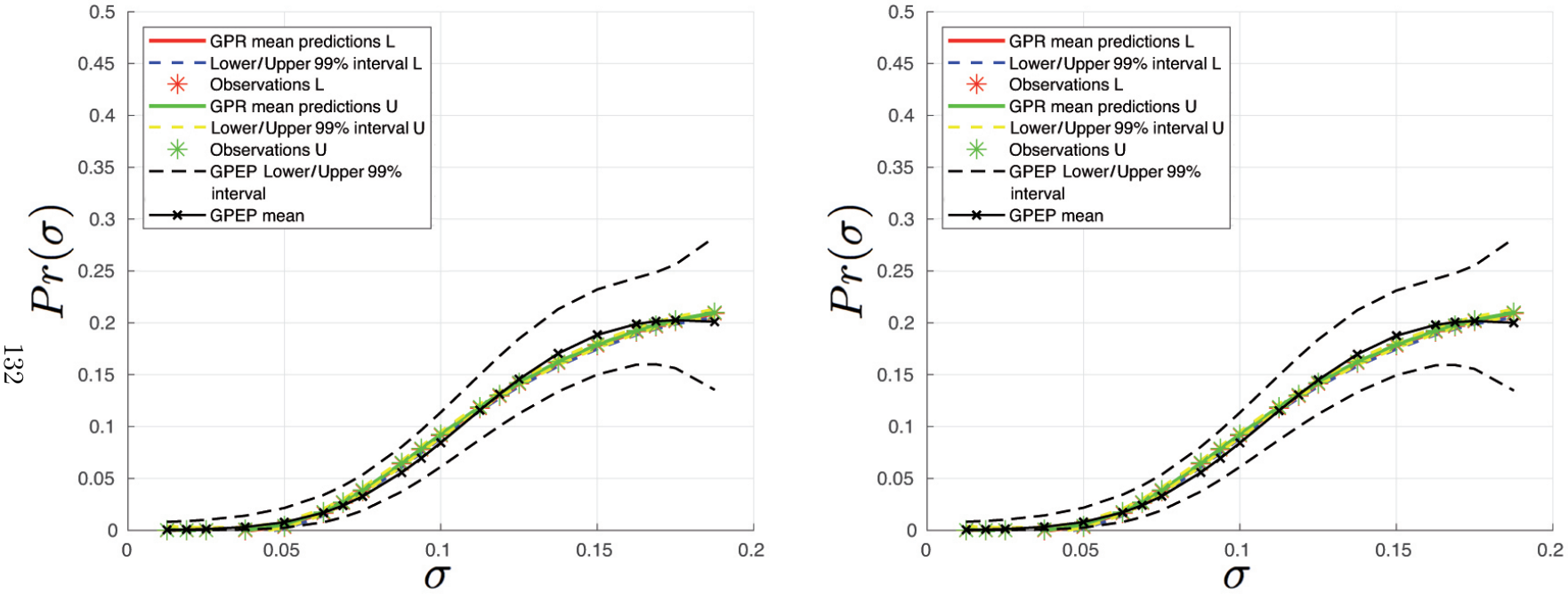
Figure 5.20: Combined (on the basis of formal approach with 0.001 precision) and GPEP approaches 0.99 confidence CIs comparison with respect to one uncertain parameter σ for 20 points.



(a) GPEP 20 samples enlargement of Figure 5.20 (a)

(b) GPEP 100 samples enlargement of Figure 5.20 (b)

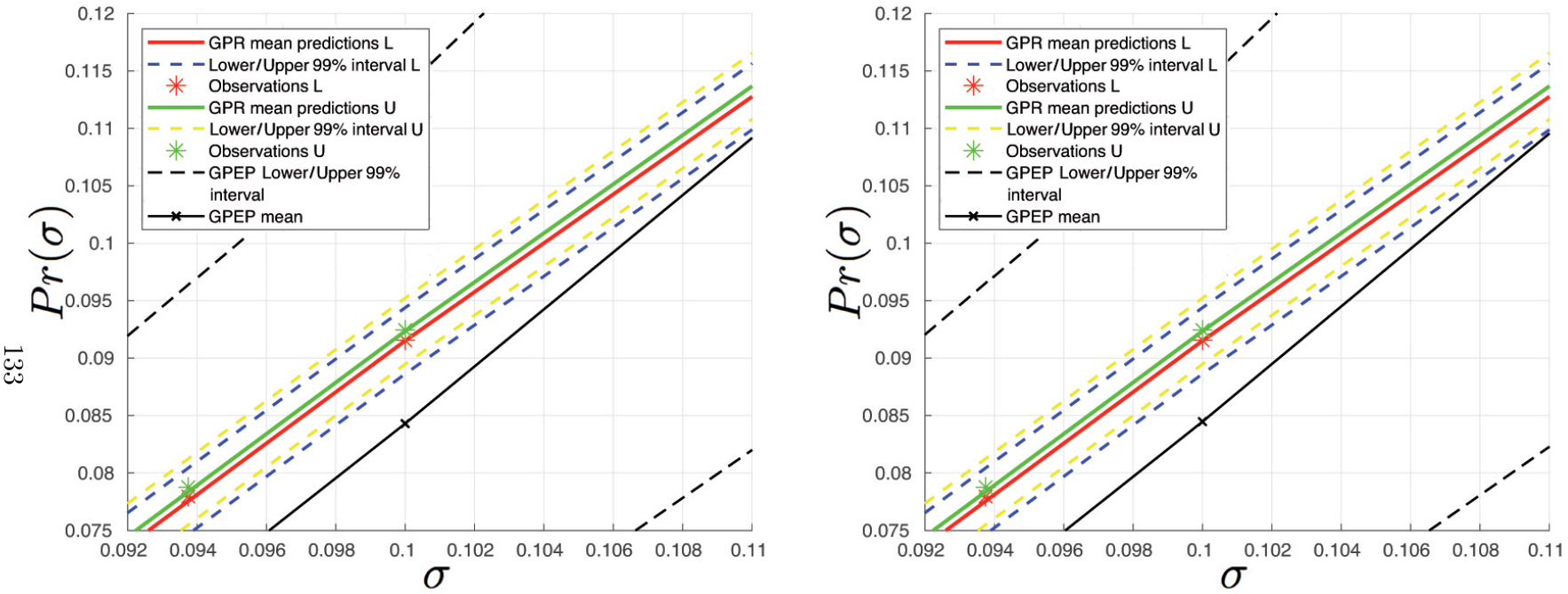
Figure 5.21: Enlargement near $\sigma = 0.1$ point for combined (on the basis of formal approach with 0.001 precision) and GPEP approaches 0.99 confidence CIs comparison with respect to uncertain parameter σ for 20 points.



(a) GPEP 1000 samples

(b) GPEP 3000 samples

Figure 5.22: Combined (on the basis of formal approach with 0.001 precision) and GPEP approaches 0.99 confidence CIs comparison with respect to one uncertain parameter σ for 20 points.



(a) GPEP 1,000 samples enlargement of Figure 5.22 (a)

(b) GPEP 3,000 samples enlargement of Figure 5.22 (b)

Figure 5.23: Enlargement near $\sigma = 0.1$ point for combined (on the basis of formal approach with 0.001 precision) and GPEP approaches 0.99 confidence CIs comparison with respect to uncertain parameter σ for 20 points.

with the increase of the formal precision. It also proves that in comparison to the classical formal approach the combined approach can provide very thin CIs over the whole parameter's domain, using fewer enclosures.

5.7.2 CPU Time Cost of the Combined Method

As it was already discussed before, the GP method has a significant advantage after the training process, because any subsequent test inputs can be calculated using a simple regression process, which is relatively fast and does not require further sampling. However, in comparison to the combined and GPEP approaches this fact does not help us to choose a winner because these two approaches both have this feature.

Computational complexity is a very important reason for the combined approach application. Formal approach computation depends on the input argument for controlling probability enclosure precision, with its complexity growing exponentially in the inverse of the precision. Also, the complexity increases exponentially with the number of system parameters. At the same time despite the fact that we need to run two GP regression processes for the combined approach, the total time, needed for the combined approach to provide GP regression is almost not noticeable in comparison with the formal part. While the GP advantage mentioned before allows us not to waste time again on parameter space exploration to estimate probability reachability function at new points, we still need to evaluate probability enclosures up to certain precision, which takes time. One of the features of the combined approach and especially its formal computational costs part is the fact that for probability function values far from the bounds (0/1 values) the formal approach spends much more time.

The GPEP approach in turn does not require such computational time as the formal approach does. The computational time of GPEP method almost fully depends on the number of samples chosen per each point. The GPEP method enables ProbReach's procedure to receive the probability value for each point and with the increase of the number of points and samples the CPU time grows dramatically.

Table 5.15 demonstrates the total CPU time difference in seconds between

Combined Approach Precision	GPEP					
	S=20	***S=50***	***S=100***	***S=200***	***S=1000***	***S=3000***
0.1	-11	-2	**19**	**58**	**184**	**552**
0.001	-65	-56	-35	**4**	**130**	**498**

Table 5.15: CPU time (in seconds) difference between GPEP (average time for 10 runs) and combined approaches (GPEP - Combined). This table presents time and samples needed for both approaches to compute CIs for 20 points with 0.99 confidence and $\boldsymbol{S}$ - number of samples per point for the Deceleration model. Results reported in **bold** show advantage for the combined approach.

GPEP and combined approaches (GPEP - Combined). It can be seen that for the smaller number of samples (20 and 50) the GPEP approach outperforms the combined approach. The picture changes with the increase of the number of samples so that for 200, 1,000 and 3,000 samples the combined approach shows better results. The difference standard deviations not reported in the table due to very small values ($<$0.0001).

However, with the increase of the precision of the enclosure computation of the combined approach and for models with probability values far from the bounds, the CPU time difference will increase in favour of GPEP. At the same time with the increase in the number of samples for the GPEP approach the combined approach still can show better results. For example, for the Deceleration model CPU computation time difference between the combined approach with 0.1 precision and GPEP approaches 100 samples is 19, while for GPEP approaches for 3,000 samples it is 552 (see Table 5.15).

In general, the results presented in Table 5.15 show that as in the case of CI size comparison for the Deceleration model it is hard to choose one particular favourite between GPEP and combined approaches. However, it is possible to define the main features and the differences in both approaches, which will be summarised and discussed in the next Subsection.

5.7.3 Combined Approach Issues

In this chapter I provided the results of the comparison of a novel statistical technique for computing bounded reachability probability in SnPHSs called combined approach with the GPEP method. The tests showed that the presented approach

gives statistically rigorous confidence intervals by combining the formal approach with an appropriate precision of probability enclosures and the GP regression method.

As it was expected the combined approach shows a severe reduction of the computational cost needed to construct CIs for the whole parameter's range space in comparison with a simple formal approach. In particular, the results presented in this Section showed that the combined approach can provide results similar to the GPEP method in terms of CPU computation time.

I discussed the problem of finding a number of samples which allow us to obtain the same size of CI for GPEP as for combined approach for different precision levels. I also considered the theoretical effectiveness of the use of GPEP approach with a large number of samples per input point with respect to the combined approach, which does not require an additional sampling. At the same time, it was shown that for small precision values like 0.1 (see Table 5.14) the advantage of the GPEP method also grows very quickly with the increase in the number of samples.

In this Section I also considered the problem of placing the mean lines and CIs borders with respect to the true probability enclosures. It was shown that the combined approach starting with precision 0.001 shows very promising results by providing very tight CIs and the two GPR mean functions which fit into the true probability enclosures. It can be easily concluded here that with the increase of enclosure precision we can obtain more and more rigorous results and approximate the latent probability reachability function more precisely. At the same time, GPEP approach did not show the same precision, because even for 1,000 and 3,000 samples per point the mean values of GPEP still lie outside the probability enclosures. We can conclude at this stage that the combined approach can show much smaller and more precise CIs than GPEP by involving the formal approach with an appropriate precision parameter.

Another important aspect which was discussed in this Section is the computational time costs of both the combined and GPEP approaches. Both approaches have a significant advantage after the training process because all the subsequent test inputs (with different points' values) can be calculated using a simple regression process, which is relatively fast and does not require further sampling in

comparison with the classical formal method. However, in the case of comparison between the combined and GPEP approaches we need to pay attention to the computational costs of the procedures which precede the GP computation.

For the combined approach the time for parameter space searching during the formal method use grows exponentially with the number of system parameters. At the same time, formal approach computation severely depends on the input argument for controlling probability enclosure precision. In other words, the algorithm's general computational complexity increases exponentially with the number of system parameters and the reachability depth.

Computational complexity is a very important factor for the combined approach application. The complexity increases exponentially with the number of system parameters. At the same time, the formal approach computation depends on the input argument for controlling probability enclosure precision, with its complexity growing exponentially in the inverse of the precision. We also note that for some heavy models it is quite impossible to run formal verification at all because of the extreme complexity, so that the combined approach can not be applied in general, while GPEP has no such problems. For the second part of the combined approach where we need to run two GP regression processes, the total time needed for the combined approach to provide GP regression is almost not noticeable in comparison with the formal part.

Another important problem of the combined approach is related to its formal part computational time costs. For probability function values far from the bounds (0/1 values) the formal approach spends much more time to return probability enclosures, even if we ask not to cover all the parameter's space with boxes but simply to compute the probability enclosures up to certain precision. This fact can lead to very different results in computation time for different models. If we assume that we do not have any initial information about the latent probability function then we can not predict how efficient the combined approach will be. This question requires further research and in particular an investigation of the opportunity to use simple SMC sampling in advance to create a general picture about the possibility of using the combined approach.

For the GPEP approach, we do not use any formal method's procedures. However, the GPEP method enables ProbReach's procedure to receive the probability

value for each point and with the increase of the number of points and samples the CPU time grows very quickly. In other words, the GPEP approach severely depends on the number of samples chosen per each point. Like in the combined approach case, the GPEP method time costs also depend on the model's complexity with that difference that GPEP method can always compute CIs.

Finally, both accuracy and computational costs results show the superiority of the combined and GPEP approaches with respect to all SMC methods. However, the question of the unconditional leader between the combined and GPEP approaches is still open. On the basis of the results provided in this Section we can conclude that in case of the combined approach it is possible to find a trade off between accuracy and computational costs, which is more difficult to do than in the GPEP case where we can have linear dependency of the results with the number of points and samples. This problem requires further investigation and testing on different SnPHS models.

Summing up the application of the combined approach results I would like to highlight that the novel approach's formal and GP regression combination provides very promising results in terms of calculation precision and computational complexity. On the basis of the small research into the SnPHS model it is feasible to predict an effectiveness of the combined approach application and choose an appropriate precision. The latter fact gives us hope that GP regression in combination with the formal approach as well as GPEP approach can be an effective solution not only for rare event cases but also in general.

5.8 Summary

The final Chapter 5 demonstrated the successful application of the implemented approaches to several case studies, such as cars collision scenarios and devising UVB irradiation therapy for treating psoriasis. In this Chapter I provided the comparison of the different CIs estimation techniques for extreme probability cases. The tests described in this Chapter showed that for probability values near the bounds (0 or 1) the developed modified CLT method achieves better results with respect the number of samples in comparison with other techniques and so it is strongly preferable when sampling cost is paramount.

Chapter 5 displayed the distribution of the Qint intervals for border probability values, which showed that the Qint CI always contains the true probability value within its bounds. In Qint test description I noted the fact that the classical Qint method can not outperform our modified CLT method, which leads us to the conclusion that the usage of the standard deviation formula advised in this thesis is quite an effective and simple solution. Due to this reason, the original Qint algorithm was changed by modifying the CLT method and the results were provided.

This Chapter also discusses the difference between the Bayesian CI and the CIs based on CLT and the use of MC and QMC techniques for interval calculation. As it was described in Subsection 2.3.3, the QMC advantage in error size holds for all of the tested models. In this Chapter I also highlighted a phenomenon that was noted during my research of MC and QMC methods: the actual coverage probability contains non-negligible oscillations. There exist some "unlucky" pairs between probability values and number of samples, which provide unpredictable results with respect to the number of samples.

Chapter 5 demonstrated the results obtained via GP estimation techniques based on the EP algorithm and CI estimation based on the standard Clopper-Pearson technique with MC sampling. I estimated the accuracy of the GP approach using the average CI interval size and root mean squared error (RMSE) of my estimates across all input points.

The results presented in this Chapter showed that for all the considered SnPHS models GPs are more accurate than SMC, and it is possible to conclude that GP estimation with EP is generally very accurate and more accurate than Clopper-Pearson SMC, and thus it can be used for the verification of SnPHS. In this Chapter I also considered CPU time costs of the GP and SMC techniques and compared the sample size needed by both techniques to produce results of similar accuracy. The provided results showed an outstanding performance of the GPEP algorithm, it can easily be seen that SMC requires much longer CPU time and a higher number of samples to achieve the CI results of GPEP.

Finally, in this chapter I presented some results of the novel combined approach. As a result a considerable advantage of combined approach was proven, with the increase of formal precision. It can also be concluded that it is possible

for the combined approach to grant very tight CIs over the whole parameter's domain, using fewer enclosures. This might be contrasted to the classical formal approach.

According to our expectations, the combined approach demonstrated a great reduction of computational costs required for constructing CIs for the whole parameter's range space, contrasted to a full formal approach. To be more precise, the acquired results demonstrated the possibility for the combined approach to provide results same as the GPEP method in terms of CPU computation time.

On the basis of the results provided we can conclude that in case of the combined approach it is possible to find a trade off between accuracy and computational costs, which is more difficult to do than in the GPEP case where we can have linear dependancy of the results.

Both computational cost and accuracy results prove the advantage of the combined and GPEP approaches over all SMC methods. At the same time, the question of the unconditional leader between the combined and GPEP approaches still remains unsolved.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis I have presented my work on the verification of stochastic nondeterministic parametric hybrid systems (SnPHS). This thesis offers contributions on two different levels to the verification of SnPHSs and demonstrates applicability of the devised methods and techniques to real-world case studies. From the theoretical point of view, it addresses the question of whether the reachability probability function in SnPHS can be considered for Gaussian processes (GP) approximation. My main theoretical result is the proof that the reachability probability function is a smooth function of the uncertain parameters of the model, and hence GP techniques can be used to obtain an efficient analytical approximation of the function. From the practical point of view I show that GPs are usable in practice even for models with complex dynamics.

I described the use of Monte Carlo (MC) and Quasi-Monte Carlo (QMC) statistical techniques for bounded reachability probability in SnPHSs and provided a novel approach for the central limit theorem confidence interval (CLT CI) method approximation. The new CLT CI method serves to provide statistically rigorous confidence intervals for systems without nondeterministic parameters. The presented approach also allows reducing the computational cost with respect to the number of samples in comparison to classical CLT interval construction.

In my thesis I showed how the expectation propagation (EP) algorithm can

be applied to compute confidence intervals which contain the bounded reachability probability value for systems with nondeterministic parameters. I also put an emphasis on the fact that the EP algorithm can be used for systems exhibiting smooth probability reachability functions only and consequently the GPEP techniques (a composition of GP and EP methods) can be successfully applied to SnPHSs. I demonstrated that the GP approach can be used to obtain an analytical approximation of the probability function. This allows us to predict the value of the probability function at all values of the uncertain parameters from individual model simulations at a finite (and generally rather small) number of distinct parameter values.

One of the main outcomes of this thesis is the creation of a novel, combined statistical approach to model checking SnPHSs. I combined the formal and GP regression to give joint absolute numerical and statistical guarantees. In particular, the formal approach in SnPHS with fixed nondeterministic parameters values, chosen with the help of QMC method, returns probability enclosures. These intervals are further divided into two arrays of input points (upper and lower) in order to train a GP and receive a predictive distribution over the nondeterministic parameter domain. This novel approach showed the most impressive results in terms of computational costs and accurate CIs construction.

Another practical contribution of this thesis is the creation of Confidence Interval Estimation and GPEP tools, which were developed for computing bounded reachability probability in stochastic parametric hybrid systems (SPHSs) and SnPHSs. The C++ implementation of the developed algorithms does not require any proprietary software. It is based on the ProbReach tool [73] and supports multiple SMT solvers (*i.e.*, iSAT-ODE and dReal). The parallelising OpenMP tool was used to increase Confidence Interval Estimation Tool and GPEP Tool performance on multi-core system. The developed GPEP Tool can be also easily used separately from the main ProbReach architecture.

The results conducted in this research showed a great accuracy advantage of the combined and GPEP approaches with respect to statistical model checking (SMC) methods (for both CIs types based on beta function and based on CLT).

The GP calculation techniques I considered have excellent convergence and efficiency especially when the number of samples is small. I empirically showed

a GP advantage in terms of CPU time, number of samples and CI average size with respect to standard statistical model checking.

In particular, according to the provided experiments GPEP approach spends between 2-56 times less CPU time for calculating results, it needs between 3-135 times fewer number of samples and it shows between 2-67 times smaller confidence interval average size, depending on the confidence level, in comparison to various SMC methods' results.

Very important results were obtained for the combined approach, which is based on the formal approach and GP regression method. My results showed that the combined approach can be extremely efficient and accurate and can outperform not only the formal but also the GPEP approaches.

As it was expected the combined approach presented a severe reduction of the computational cost needed to construct CIs for the whole parameter's range space in comparison with a simple formal approach. The results also displayed that in comparison to the classical formal approach the combined approach can provide very thin CIs over the whole parameter's domain, using fewer enclosures. Based on the presented results we can make a conclusion that in case of the combined approach a trade-off between accuracy and computational costs can be found.

In general, the experiments showed that the combined approach provides very promising results in terms of CI size and CPU time. This approach definitely requires further investigation with the aim to find a gold trade-off between the number of samples and CPU time spent and analysis of rare-event SnPHS models.

In this thesis, I also provided a comprehensive evaluation of CIs calculation techniques based on the MC and QMC techniques. The experiments show that my modified CLT technique is usable in practice even for complex dynamics and for probabilities close to the bounds. Based on my analysis of CIs, I suggest that my results can be used as guidelines for probability estimation techniques.

6.2 Future Work

In this Section I describe the future work directions, including possible software improvements.

One of the most important points of the future work is further investigating a combined formal-GP approach in which GP training exploits the rigorous probability enclosures computed by ProbReach's formal approach. In particular, I plan to investigate GP behaviour in rare-event cases, where the true probability function is very close to 0. In accordance with this research, I also plan to pay attention to researching rare-event EP behaviour and finding effective ways for variance minimisation.

It is well known that rare events are events that are expected to occur infrequently, in the other words they are events whose probabilities to occur are equal to about 10^{-6} or less for a probability model. In this situation, rare events often cause a failure of systems designed for high reliability. In the context of uncertainty quantification that means that a system is unable to detect a rare event, which can lead to serious problems for safe-state systems. That is why the computation of such rare-event probabilities is such an important problem. As it was stated before in my thesis analytical solutions are usually not available for complex problems. I want to address the problem of estimating rare-event probabilities by GP methods, which can also be done by involving importance sampling, and subset simulation techniques.

For testing GP techniques for the rare-event case, we could use a simple model with the small failure probability for a highly reliable dynamic system for example aircraft under uncertain turbulence excitation or building under uncertain earthquake excitation. Such models represent typical dynamic systems with both parametric uncertainty (what values of the model parameters best represent the behaviour of the system?) and nonparametric modelling uncertainty (what are the effects of the aspects of the system behaviour not captured by the dynamic model?).

Another point for the future investigation is researching the efficacy of the combined approach to the probability function values far from the bounds (0/1 values) which is very important due to the formal approach's large computational

costs to return probability enclosures for such values.

As it was discussed in this thesis, the formal approach allows us to compute probability enclosures. However, despite the given precision for nondeterministic parameter boxes, the formal approach output in form of probability enclosures vary its size in accordance with the estimated probability function value. The technique behind the algorithm returns larger probability enclosures sizes for the probability values far from the bounds. It is important to investigate this aspect of the combined approach work because it affects directly the approach's efficacy.

Computational complexity is one of the most important aspects of the combined approach application. The time for parameter space search has a strong influence on the overall use of the algorithm. This time directly depends on the formal approach partitioning procedure, because in general the combined approach does not spend much time on the regression computation by the GP algorithm.

The extension of the test model range is the next challenge for further research. In particular, I plan to include complex models whose reachability probability function can not be calculated analytically and whose evaluation takes considerable time. In accordance with the above mentioned points for the future work, I would like to add rare-event models, including complex, curved and almost flat lines and modes with probability functions which are located in the middle of the probability space. The fast oscillating function is a particular point of interest.

In future work, I also plan to do some technical improvements, one of which is 3D visualization of the chosen stochastic models and direct graphical visualization through Matlab of all produced approximation results. I also plan to make an improvement in the Gaussian process editor tool by adding new covariance features and characteristics, which are known to be governed by the mean function and covariance function. That is why for choosing distinct covariance functions, some initially obtained information on the probability reachability function may be useful.

In many practical applications, it may not be efficient to apply one covariance function with confidence to receive a good result. In light of the above, the expansion of the set of covariance functions is an important addition to the current

version of the GPEP tool. This improvement can turn GPEP into a more practical tool and address its methods to the model selection problem. As part of this work, I plan to consider the model selection problem rather broadly, to identify all aspects of future model approximation by choosing an efficient functional form for hyperparameters and the covariance function.

In fact, model selection can help to give an important understanding to the user about the properties of the data, (e.g. that a polynomial covariance function may be preferred over an exponential one) and to refine the predictions of the model. I suggest using different families of covariance functions for this purpose, including squared exponential, polynomial, neural network, constant, linear, Matern, exponential, rational quadratic, etc. It is important to note that all these families also have different hyperparameters whose values also need to be determined.

Another technical improvement aspect is implementing a more efficient parallelisation approach involving the development of a sophisticated parallelisation manager controlling the accessibility of the CPUs and the dynamic administration of the load between the threads in order to decrease idle CPUs. This could substantially improve the functioning of the developed tools as well as implementing a user-friendly interface for model representation instead of current PDRH format.

List of Tables

List of Figures

Appendix A

A.1 Algorithms

Algorithm 1 Sampling **Pr** for GP training

Inputs: H : SnPHS, $l \in \mathbb{N}$: reachability depth, $\delta > 0$: solver precision, N : number of training points, S : number of samples per training point.

Output: $X = x_1, \ldots, x_N$ training points, $Y = y_1, \ldots, y_N$ probability values.

$n \leftarrow 0$;
for $n < N$ **do**
    $v \leftarrow 0$;
    $d \leftarrow 0$;
    $x_n \leftarrow QMC\_sample(P)$; // sampling nondeterministic parameters via Quasi-Monte Carlo
    for $d < S$ **do**
        $rnd \leftarrow MC\_sample(R)$; // Monte Carlo sampling random parameters
        $d \leftarrow d + 1$;
        // formally evaluate reachability - see Section 4.2 in [72]
        switch evaluate(H, l, x_n, rnd, δ) **do**
            case unsat do $v \leftarrow v + 1$; // count 'true' unsatisfiable reachability only, anything else is satisfiable
    end for
    $y_n \leftarrow (S - v)/S$; // estimate $\mathbf{Pr}(x_n)$
    $n \leftarrow n + 1$;
end for

Algorithm 2 GP training via the Expectation-Propagation algorithm (adapted from Algorithm 3.5 in [61])

Inputs: K: covariance matrix,

X: nondet points,

N: number of points,

Y: probability values.

Output: $\grave{\omega}$, $\grave{\eta}$: natural site parameters.

$\grave{\omega} \leftarrow 0$, $\grave{\eta} \leftarrow 0$, $\Sigma \leftarrow 0$, $\mu \leftarrow 0$;
repeat
  for $i < N$ **do**
    $\eta_{-i} \leftarrow \sigma_i^{-2} - \grave{\eta}_i$;
    $\omega_{-i} \leftarrow \sigma_i^{-2}\mu_i - \grave{\omega}_i$; // Compute approximate cavity parameters (Eqs. (3.55) and (3.56) [61])
    $z_i \leftarrow y_i\mu_{-i}/\sqrt{1+\sigma_{-i}^2}$;
    $\hat{z}_i \leftarrow \Phi(z_i)$; // See Eqs. (3.50) and (3.52) [61]
    $\hat{\mu}_i \leftarrow \mu_{-i} + (y_i\sigma_{-i}^2\mathcal{N}(z_i))/(\Phi(z_i)\sqrt{1+\sigma_{-i}^2})$;
    $\hat{\sigma}_i^2 \leftarrow \sigma_{-i}^2 - \frac{\sigma_{-i}^4\mathcal{N}(z_i)}{(1+\sigma_{-i}^2)\Phi(z_i)}(z_i + \frac{\mathcal{N}(z_i)}{\Phi(z_i)})$;
    \\Compute marginal moments $\hat{\mu}_i$ and $\hat{\sigma}_i^2$ see Eq. (3.58) [61].
    $\Delta\grave{\eta} \leftarrow \hat{\sigma}_i^{-2} - \eta_{-i} - \grave{\eta}_i$;
    $\grave{\eta}_i \leftarrow \grave{\eta}_i + \Delta\grave{\eta}$;
    $\grave{\omega}_i \leftarrow \hat{\sigma}_i^{-2}\hat{\mu}_i - \omega_{-i}$;
    \\Update site parameters $\grave{\eta}_i$ and $\grave{\omega}_i$ see Eq. (3.59) [61].
    $\Sigma = \Sigma - ((\Delta\grave{\eta})^{-1} + \Sigma_{ii})^{-1}C_iC_i^\top$; // see Eq. (3.70) [61], where C_i is i column of Σ.
    $\mu \leftarrow \Sigma\grave{\omega}$; // see Eq. (3.53) [61]
    $i \leftarrow i + 1$;
  end for
  $L \leftarrow cholesky(I_n + \check{C}^{\frac{1}{2}}K\check{C}^{\frac{1}{2}})$;
  \\$\check{C}$ diagonal of Σ, I_n identity matrix.
  $V \leftarrow L^\top\backslash\check{C}^{\frac{1}{2}}K$
  $\Sigma \leftarrow K - V^\top V$
  \\See Eq. (3.53) and (3.68) [61].
  $\mu \leftarrow \Sigma\grave{\omega}$
  \\Recompute approximate posterior parameters see Eq. (3.53) [61].
until convergence.

Algorithm 3 GPEP regression (adapted from Algorithm 3.6 in [61])

Inputs: k : covariance function, $\mathbf{p}^*$: nondeterministic test point, $c \in (0,1)$: confidence (coverage probability), $\grave{\omega}$, $\grave{\eta}$: natural site parameters (computed by Expectation-Propagation)

Output: $\bar{y}^*$: mean $\mathbf{Pr}(\mathbf{p}^*)$ value, Low: lower CI bound, Up : upper CI bound

1: $L \leftarrow cholesky(I_n + \check{C}^{\frac{1}{2}} K \check{C}^{\frac{1}{2}})$;
2: $\zeta \leftarrow \check{C}^{\frac{1}{2}} L^\top \backslash (L \backslash (\check{C}^{\frac{1}{2}} K \grave{\omega}))$;
3: $\bar{y}^* \leftarrow \mathbf{k}(\mathbf{p}^*)^\top (\grave{\omega} - \zeta)$; // compute mean value (Eqs. (3.60) and (3.71) [61])
4: $\mathbf{t} \leftarrow L \backslash (\check{C}^{\frac{1}{2}} \mathbf{k}(\mathbf{p}^*))$; // see Eq. (3.61) [61]
5: $\mathcal{V}(y^*) \leftarrow k(\mathbf{p}^*, \mathbf{p}^*) - \mathbf{t}^\top \mathbf{t}$;
6: $Low, Up \leftarrow \mathrm{CDF}(\bar{y}^* \mp \mathrm{CDF}^{-1}(1 - (1-c)/2)\mathcal{V}(y^*))$; // compute confidence intervals of coverage at least c (Eq. (3.72) [61])

Algorithm 4 Formal approach collecting **Pr** enclosures phase for the combined approach (adapted from Algorithm 3 in [72])

Inputs: $(H, \mathbb{P})$: SnPHS(P_{rand}, P_{non}),
$l \in \mathbb{N}$: reachability depth,
$\epsilon \in \mathbb{Q}^+$: enclosure precision,
$\kappa \in (0, \epsilon)$: size of probability enclosures,
$\rho \in \mathbb{Q}^+$: precision for nondet parameter box,
$\eta \in \mathbb{Q}^+$: multiplier for controlling precision of δ-decision procedure.

Output: L_{up}, L_{low}: lists of upper and lower values of the probability enclosure.

1: $Q \leftarrow (B_N, [a, b], \Pi_R)$; // B_N is a nondeterministic parameter box, $[a, b]$ is a probability enclosure and Π_R is a list of random parameter boxes.
2: **repeat**
3: $Q \rightarrow (B_N, [a, b], \Pi_R)$;
4: **repeat**
5: $B_R \leftarrow \Pi_R$;
6: $[c, d] \leftarrow$ **measure**$(B_R, \mathbb{P}, (\epsilon - \kappa)\frac{\mu^+(B_R)}{\mu^+(P_R)})$; // See Algorithm 5 in Section 3.5.2 of [72].
7: $\delta \leftarrow \eta \cdot \min\left(|B_R|^+\right)$;
8: **switch evaluate**$(H, l, B_R x B_N, \delta)$ **do**
9: **case sat do** $a \leftarrow a + c$;
10: **case unsat do** $b \leftarrow b - c$;
11: **case undet do** form Q_R // B_R is partitioned using procedure **bisect**, and each obtained sub-box is pushed to the queue Q_R to be analysed in the next iteration of the outer loop (see [72]).
12: **until** $(|\Pi_R| = 0)$
13: **if** $(|[a, b]| \leq \epsilon) \vee (|B_N| \leq \rho)$ **then**
14: $L \leftarrow (B_N, [a, b])$; // L is a list of pairs (nondet. parameter box, probability enclosure.
15: **else**
16: **for** $B \in Q_N$ **do**
17: $Q \leftarrow (B, [a, b], Q_R)$;
18: **end for**
19: **end if**
20: **until** $(|Q| = 0)$
21: $L_{up} = max(L)$, $L_{low} = min(L)$;

Algorithm 5 GP regression for the combined approach (adapted from Algorithm 2.1 in [61])

Inputs: k : covariance function, L_{up}: list of upper bounds of probability enclosures, L_{low}: list of lower bounds of probability enclosure, $\mathbf{p}^*$: nondeterministic test point, $c \in (0,1)$: confidence (coverage probability)

Output: CIs with coverage c.

1: $S_{up} \leftarrow cholesky(K(L_{up}) + \sigma^2 I)$; // compute Cholesky decomposition (see [61, Appendix A.4])
2: $\alpha_{up} \leftarrow S_{up}^{\top} \backslash (S_{up} \backslash (L_{up})$;
3: $\bar{y}_{up}^* \leftarrow \mathbf{k}_*^{\top} \alpha_{up}$; // compute mean value (Eq. (2.25) [61])
4: $\mathbf{t}_{up} \leftarrow S_{up} \backslash \mathbf{k}_*$;
5: $\mathcal{V}_{up}(y_{up}^*) \leftarrow k(\mathbf{p}^*, \mathbf{p}^*) - \mathbf{t}_{up}^{\top} \mathbf{t}_{up}$; // compute variance (Eq. (2.26) [61])
6: $U_b \leftarrow \mathrm{CDF}(\bar{y}_{up}^* + \mathrm{CDF}^{-1}(1 - (1-c)/2)\mathcal{V}_{up}(y_{up}^*))$; // compute CI upper bounds for L_{up} points
7: $\langle$ repeat steps 1-5 for L_{low} points $\rangle$
8: $L_a \leftarrow \mathrm{CDF}(\bar{y}_{lo}^* - \mathrm{CDF}^{-1}(1 - (1-c)/2)\mathcal{V}_{lo}(y_{lo}^*))$; // compute CI lower bounds for L_{low} points
9: **return** $[L_a, U_b]$

A.2 Tools Usage

A.2.1 Confidence Interval Estimation Tool Usage

`Confidence Interval Estimation Tool` can be executed by running the following command:

```
qmc_verifier <options>  <solver-options> <file.pdrh/file.drh>
```

The examples below demonstrate an application of the developed CI estimation approaches to the Deceleration model (see Figure 4.2). The full list of command line arguments available can be found in documentation[1].

Example A.1. (Applying Confidence Interval Estimation modified CLT method to Deceleration model)

```
./qmc_verifier -t 3 --qmc-acc 0.1 --qmc-conf 0.99
--CI CLT    --verbose-result --solver dReal
../../EPPR-models/deceleration/stop-nonlinear.pdrh
```

where

- `-t 3` - specifies the reachability depth $l = 3$.
- `--qmc-acc 0.01` - specifies the half-size of the confidence interval computed by chosen CI estimation method.
- `--qmc-conf 0.99` - specifies the confidence value for chosen CI estimation method.
- `--CI CLT` - defines a CI estimation method (CLT - CLT, AC - Agresti-Coull, W - Wilson, L - Logit, ANC - Anscombe, ARC - Arcsine, Q - Qint, ALL - all listed methods, CP - exact Clopper-Pearson (see Chapter 2)).
- `--verbose-result` - provides detailed output.

[1] https://github.com/dreal/probreach/blob/master/doc/usage.md

- `--solver dReal` - specifies the full path to the solver executable. In this example it is assumed that the directory containing `dReal` is added to the path or defined as a symbolic link.
- `../../EPPR-models/deceleration/stop-nonlinear.pdrh` - specifies the full path to the file containing the PDRH model, presented in Figure 4.2.

`Confidence Interval Estimation Tool` produces the following output:

```
...
------------
ICDF sample :beta:[3.89002093333449,3.89002093333449];
UNSAT
Number of SAT: 2033
Number of UNSAT: 20103
Number of UNDET: 0
ressat: 0.09184134441633539
resunsat: 0.09184134441633539
points: 22136
samplevar = 0.08341027995489453
Interval/2 = 0.005000083948350912
------------
ICDF sample :beta:[4.034701951195709,4.034701951195709];
UNSAT
Number of SAT: 2033
Number of UNSAT: 20104
Number of UNDET: 0
ressat: 0.09183719564529973
resunsat: 0.09183719564529973
points: 22137
samplevar = 0.08340689290084505
Interval/2 = 0.004999869493754902
------------
1-test running points = 22137
```

```
CI Interval:
[8.6837326151544839e-02,9.6837065139054643e-02] | 9.9997389875098042e-03
...
```

The output represents the two last evaluations of the parameter's point and shows the sample point, goal decision, total number of SAT/UNSAT and UNDET results (see Section 3.2.1), satisfaction proportion for SAT values, total points number, current sample variance and half of the intervals size. In the last string of the output we can see final CI values [,] and the returned CI's size.

These produced CIs are presented in Chapter 5 in Table 5.3.

Example A.2. (Applying the full range of confidence interval estimation methods to Deceleration model)

```
./qmc_verifier -t 3 --qmc-acc 0.1 --qmc-conf 0.99
--CI ALL  --verbose-result --solver dReal
../../EPPR-models/deceleration/stop-nonlinear.pdrh
```

An example of the produced output is presented below:

```
...
-----------------------------------------------------------
CLT RESULTS:
10-tests average running points number for CLT=22393
[INTERVAL CLT]= [0.08668385001344965,0.0966837436565206]
-----------------------------------------------------------
AGRESTI-COUL RESULTS:
10-tests average running points number for AC=22673
[INTERVAL AC]= [0.08667091848577629,0.09667052692221982]
-----------------------------------------------------------
WILSON RESULTS:
10-tests average running points number for W=22517
[INTERVAL W]= [0.08680819267258803,0.09680816001757328]
-----------------------------------------------------------
LOGIT RESULTS:
10-tests average running points number for L=22628
```

```
[INTERVAL L]= [0.08486976079398766,0.09486953595889873]
-----------------------------------------------------------
ANSCOMBE RESULTS:
10-tests average running points number for ANS=22623
[INTERVAL ANS]= [0.08489267369498253,0.09489265488272093]
-----------------------------------------------------------
ARCSINE RESULTS:
10-tests average running points number for ARC=24365
[INTERVAL ARC]= [0.08479567364498253,0.09499567311242648]
-----------------------------------------------------------
QINT RESULTS:
10-tests average running points number for QINT=20318
[INTERVAL QINT]= [0.0885282452483752,0.09852432348023947]
-----------------------------------------------------------
```

The output represents final CIs values and the returned CI's sizes per every method. These produced CIs were presented in Chapter 5 in Table 5.3.

A.2.2 GPEP Tool Usage

`GPEP Tool` can be executed by running the following command:

```
gp <solver-options> <file.pdrh/file.drh> <options>
```

The example below demonstrates application of `GPEP Tool` to the Deceleration model presented in Figure 4.5. The full list of command line arguments available can be found in documentation[1].

Example A.3. Applying `GPEP Tool` to deceleration model (see Figure 4.5)

```
./gp -u 3 --verbose  ../../EPPR-models/deceleration/stop-nonlinear.pdrh
-n 20 --conf 0.99 --samples 100
```

where

[1] https://github.com/dreal/probreach/blob/master/doc/usage.md

- *-u 3* - *specifies the reachability depth* $l = 2$.
- *--verbose* - *provides detailed output.*
- *../../model/IFM/cars-old/stop-nonlinear.pdrh* - *specifies the full path to the file containing the PDRH model, presented in Figure 4.5.*
- *-n 20* - *specifies number of points.*
- *--conf 0.99* - *specifies the confidence of CIs for EP algorithm.*
- *--samples 100* - *specifies number of samples.*

`GPEP Tool` produces the `log.txt` file with the following output:

```
getClassProbabilities      getLowerBound          getUpperBound
------                     ------                 ------
length=20                  length=20              length=20
0.0829272                  0.0594135              0.112837
0.192409                   0.152558               0.238051
0.00550462                 0.00137792             0.0182607
0.0287358                  0.0158105              0.0494464
0.205186                   0.158444               0.259268
0.148464                   0.114409               0.188706
0.000617304                2.33169e-05            0.00843252
0.00194342                 0.000225692            0.0117152
0.174433                   0.136279               0.218758
0.203347                   0.136042               0.287339
0.0523111                  0.0341042              0.0775036
0.0135741                  0.00555663             0.0300782
0.202232                   0.161988               0.247926
0.11662                    0.0877773              0.151613
0.000185826                1.49762e-06            0.00718803
0.000339633                6.26731e-06            0.00760053
0.133063                   0.101561               0.170715
0.204449                   0.162418               0.252353
```

```
0.0201346          0.00974632          0.0386994
0.0669403          0.0460768           0.0944565
```

The output represents estimated mean reachability probability function (getClassProbabilities), CI lower bound (getLowerBound) and CI upper bound (getUpperBound) per every input point (20). The obtained results are visualised in Chapter 5 in Figure 5.10.

The tool also produces the `test.scv` file with the following output:

N%	NOND-point	CPLower	CPUpper	CPCenter
0	0.1046	0.032117	0.189152	0.09
1	0.1523	0.100817	0.309838	0.19
2	0.0574	0	0.051604	0
3	0.0763	0.003407	0.105481	0.03
4	0.1742	0.116101	0.332522	0.21
5	0.1247	0.064548	0.251285	0.14
6	0.0261	0	0.051604	0
7	0.0372	0	0.051604	0
8	0.1376	0.078684	0.275057	0.16
9	0.1876	0.116101	0.332522	0.21
10	0.0874	0.020789	0.162803	0.07
11	0.0622	5.01E-05	0.0719577	0.01
12	0.1636	0.108411	0.321226	0.2
13	0.1126	0.051009	0.226955	0.12
14	0.0111	0	0.051604	0
15	0.0183	0	0.051604	0
16	0.1187	0.057697	0.239196	0.13
17	0.1686	0.108411	0.321226	0.2
18	0.0689	0.001039	0.0894307	0.02
19	0.0931	0.026301	0.176114	0.08

The obtained results representing Clopper-Pearson CI estimation method (see Subsection 2.4.1) for comparison with the GPEP results were visualised in Chapter 5 in Figure 5.10.

References

[1] Antonov Anton A. and Ermakov Sergei M. Empirically estimating error of integration by quasi-monte carlo method. *Vestnik St. Petersburg University: Mathematics*, **47**[1]:1–8, 2015. 5, 29, 30, 98

[2] Alessandro Abate, Joost-Pieter Katoen, John Lygeros, and Maria Prandini. Approximate model checking of stochastic hybrid systems. *European Journal of Control*, **16**[6]:624 – 641, 2010. 5

[3] Alessandro Abate, Maria Prandini, John Lygeros, and Shankar Sastry. Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems. *Automatica*, **44**[11]:2724–2734, November 2008. 9

[4] Hamzah Abdel-Aziz and Xenofon Koutsoukos. Online model learning of buildings using stochastic hybrid systems based on Gaussian processes. *Journal of Control Science and Engineering*, 2017. Article ID 3035892. 6

[5] Alan Agresti and Brent A. Coull. Approximate is better than exact for interval estimation of binomial proportions. *The American Statistician*, **52**[2]:119–126, May 1998. 34

[6] Javier Leach Albert, Burkhard Monien, and Mario Rodríguez-Artalejo, editors. *Automata, Languages and Programming, 18th International Colloquium, ICALP91*, **510**. Springer, 1991. 1

[7] Rajeev Alur, Costas Courcoubetis, Thomas A. Henzinger, and Pei-Hsin Ho. Hybrid automata: An algorithmic approach to the specifica-

tion and verification of hybrid systems. In *Hybrid Systems*, **volume 736** of *Lecture Notes in Computer Science, LNCS*, pages 209–229, 1992. 1, 11, 14

[8] Frank J. Anscombe. The transformation of poisson, binomial and negative-binomial data. *Biometrika*, **35**[3/4]:246–254, December 1948. 34

[9] Frank J. Anscombe. On estimating binomial response relations. *Biometrika*, **43**[3/4]:461–464, December 1956. 34

[10] John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman. Learning bounds for domain adaptation. In *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, pages 129–136, 2007. 44

[11] Luca Bortolussi, Dimitrios Milios, and Guido Sanguinetti. Smoothed model checking for uncertain continuous-time Markov chains. *Inf. Comput.*, **247**:235–253, 2016. 6, 48, 69, 84

[12] Luca Bortolussi and Guido Sanguinetti. A statistical approach for computing reachability of non-linear and stochastic dynamical systems. In Gethin Norman and William Sanders, editors, *11th International Conference on Quantitative Evaluation of SysTems, QEST 2014*, pages 41–56. Springer, 2014. 6, 39

[13] Vasco Brattka, Peter Hertling, and Klaus Weihrauch. A tutorial on computable analysis. In S. Barry Cooper, Benedikt Löwe, and Andrea Sorbi, editors, *New Computational Paradigms*, pages 425–491. Springer New York, 2008. 16

[14] Cynthia J. Clopper and Egon S. Pearson. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, **26**[4]:404–413, December 1934. 33, 58

[15] Koby Crammer, Michael J. Kearns, and Jennifer Wortman. Learning from multiple sources. *J. Mach. Learn. Res.*, **9**:1757–1774, 2008. 44

[16] Brown Lawrence D., Cai T. Tony, and Dasgupta Anirban. Interval estimation for a binomial proportion. *Statistical Science*, **16**[2]:128–133, 2001. 3, 5, 57, 58, 101

[17] J. Estrela da Silva, Bruno Terra, Ricardo Martins, and João Borges de Sousa. Modeling and simulation of the lauv autonomous underwater vehicle. In *13th IEEE IFAC International Conference on Methods and Models in Automation and Robotics*, 2007. 9

[18] Philip Dawid and Allan Skene. Maximum likelihood estimation of observer error rates using the em algorithm. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, **28**[1]:20–28, 1979. 44

[19] Natalie Dean and Marcello Pagano. Evaluating confidence interval methods for binomial proportions in clustered surveys. *Journal of Survey Statistics and Methodology*, **3**[4]:484–503, December 2015. 34, 58

[20] Josef Dick and Friedrich Pillichshammer. Multivariate integration in weighted hilbert spaces based on walsh functions and weighted sobolev spaces. *J. Complexity*, **21**[2]:149–195, 2005. 28

[21] Wilson B. Edwin. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, **22**[158]:209–212, 1927. 33

[22] Andreas Eggers, Martin Fränzle, and Christian Herde. SAT modulo ODE: A direct SAT approach to hybrid systems. In *ATVA*, **5311** of *LNCS*, pages 171–185, 2008. 17, 72

[23] Andreas Eggers, Nacim Ramdani, Nedialko S. Nedialkov, and Martin Fränzle. Improving the SAT modulo ODE approach to hybrid systems analysis by combining different enclosure methods. *Software & Systems Modeling*, **14**[1]:121–148, 2015. 17, 50

[24] Christian Ellen, Sebastian Gerwinn, and Martin Fränzle. Statistical model checking for stochastic hybrid systems involving nondeterminism over continuous domains. *International Journal on Software Tools for Technology Transfer (STTT)*, **17**[4]:485–504, 2015. 2, 5, 6

[25] Sergei M. Ermakov. *The Monte Carlo Method and Related Questions*. Nauka: Moscow, 1975. in Russian. 29

[26] Maria Fox, Derek Long, and Daniele Magazzeni. Plan-based policies for efficient multiple battery load management. *J. Artif. Intell. Res. (JAIR)*, **44**:335–382, 2012. 9

[27] Martin Fränzle. *Analysis of Hybrid Systems: An Ounce of Realism Can Save an Infinity of States*, **volume 1683** of *Lecture Notes in Computer Science*, pages 126–140. Springer, 1999. 15

[28] Martin Fränzle, Christian Herde, Tino Teige, Stefan Ratschan, and Tobias Schubert. Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure. *JSAT*, **1**[3-4]:209–236, 2007. 15

[29] Victor Gan, Guy Albert Dumont, and Ian M. Mitchell. Benchmark problem: A pk/pd model and safety constraints for anesthesia delivery. In *ARCH@CPSWeek*, 2014. 91

[30] Sicun Gao, Jeremy Avigad, and Edmund M. Clarke. Delta-decidability over the reals. In *Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science, LICS 2012*, pages 305–314, 2012. 1, 10, 15, 16, 17, 18, 50

[31] Sicun Gao, Soonho Kong, and Edmund M. Clarke. dreal: An SMT solver for nonlinear theories over the reals. In Maria Paola Bonacina, editor, *Automated Deduction - CADE-24 - 24th International Conference on Automated Deduction, Lake Placid*, **volume 7898** of *Lecture Notes in Computer Science*, pages 208–214. Springer, 2013. 17, 72, 101

[32] Sicun Gao, Soonho Kong, and Edmund M. Clarke. Satisfiability modulo ODEs. In *Formal Methods in Computer-Aided Design, FMCAD 2013*, pages 105–112, 2013. 17

[33] Yang Gao and Martin Fränzle. *A Solving Procedure for Stochastic Satisfiability Modulo Theories with Continuous Domain*, pages 295–311. Springer International Publishing, Cham, 2015. 5

[34] G. Gevorkyan and K. Navasardyan. Uniqueness theorems for generalized haar systems. *Mathematical Notes*, **104**[1-2]:10–21, 2018. 29

[35] Paul Glasserman, Philip Heidelberger, Perwez Shahabuddin, and Tim Zajic. Multilevel splitting for estimating rare event probabilities. *Operations Research*, **47**[4]:585–600, 1999. 57

[36] Paul Glasserman and Sandeep Juneja. Uniformly efficient importance sampling for the tail distribution of sums of random variables. *Math. Oper. Res.*, **33**[1]:36–50, 2008. 57

[37] Michael Gnewuch, Anand Srivastav, and Carola Winzen. Finding optimal volume subintervals with k points and calculating the star discrepancy are np-hard problems. *J. Complexity*, **25**[2]:115–127, 2009. 3, 5, 26

[38] Alfred Haar. The theory of orthogonal function systems. *Mathematical annals (in german)*, **69**[3]:331–371, 1910. 29

[39] John H. Halton. A retrospective and prospective survey of the monte carlo method. *SIAM Review*, **12**[1]:1–63, 1970. 24

[40] Chuan-Hsiang Han and Yongzeng Lai. A smooth estimator for MC/QMC methods in finance. *Mathematics and Computers in Simulation*, **81**[3]:536–550, 2010. 23

[41] Roman Hovorka. Closed-loop insulin delivery: from bench to clinical practice. *Nature Reviews Endocrinology*, **7**[7]:385–395, 2011. 9

[42] Li Hui and David Walter. Estimating the error rates of diagnostic tests. *Biometrics*, **36**[1], 1980. 44

[43] Xiaoqing Jin, Jyotirmoy V. Deshmukh, James Kapinski, Koichi Ueda, and Ken Butts. Powertrain control verification benchmark. In *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control*, HSCC '14, pages 253–262, 2014. 9

[44] Ker-I. Ko. *Complexity Theory of Real Functions*. Birkhäuser, 1991. 16

[45] Andrei Kolmogorov. *Selected works of A.N. Kolmogorov*. Mathematics and its applications (Kluwer Academic Publishers). Soviet series ; 25-27. Kluwer Academic Publishers, Dordrecht, 1991. 35

[46] Soonho Kong, Sicun Gao, Wei Chen, and Edmund M. Clarke. dreach: δ-reachability analysis for hybrid systems. In Christel Baier and Cesare Tinelli, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 21st International Conference, TACAS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015. Proceedings*, **volume 9035** of *Lecture Notes in Computer Science*, pages 200–205. Springer, 2015. 50, 73

[47] Miklos Laczkovich. The removal of π from some undecidable problems involving elementary functions. *Proceedings of the American Mathematical Society*, **131**[7]:pp. 2235–2240, 2003. 15

[48] Xiangfang Li, Oluwaseyi Omotere, Lijun Qian, and Edward R. Dougherty. Review of stochastic hybrid systems with applications in biological systems modeling and analysis. *EURASIP Journal on Bioinformatics and Systems Biology*, **2017**[1]:8, Jun 2017. 9, 12

[49] Bing Liu, Soonho Kong, Sicun Gao, Paolo Zuliani, and Edmund M. Clarke. Parameter synthesis for cardiac cell hybrid models using delta-decisions, 2014. 9

[50] Kalpana K. Mahajan, Sangeeta Arora, and Kamaljit Kaur. Bayesian estimation for gini index and a poverty measure in case of pareto distribution using jeffreys' prior. *MASA*, **10**[1]:63–72, 2015. 32

[51] Khosrow Maleknejad and Farshid Mirzaee. Numerical solution of linear fredholm integral equations system by rationalized haar functions method. *Int. J. Comput. Math.*, **80**[11]:1397–1405, 2003. 28

[52] Oded Maler, Zohar Manna, and Amir Pnueli. From timed to hybrid systems. In *Real-Time: Theory in Practice, REX Workshop, Mook, The Netherlands, June 3-7, 1991, Proceedings*, **volume 600** of *Lecture Notes in Computer Science*, pages 447–484. Springer, 1991. 1

[53] Thomas P. Minka. Expectation propagation for approximate Bayesian inference. In *UAI*, pages 362–369, 2001. 6, 11, 48, 49, 50, 63

[54] Harald Niederreiter. *Random number generation and Quasi-Monte Carlo methods*, **volume 63** of *CBMS-NSF regional conference series in applied mathematics*. SIAM, 1992. 25, 26

[55] Erich Novak and Henryk Wozniakowski. Relaxed verification for continuous problems. *Journal of Complexity*, **8**[2]:124 – 152, 1992. 15

[56] Manfred Opper and Ole Winther. Gaussian processes for classification: Mean-field algorithms. *Neural Computation*, **12**[11]:2655–2684, 2000. 6, 11, 48, 49

[57] Giray Oñkten and Warren Eastman. Randomized quasi-monte carlo methods in pricing securities. *Journal of Economic Dynamics and Control*, **28**[12]:2399–2426, 2004. 27

[58] Bob Phillips. Rule of three. *Archives of Disease in Childhood*, **90**[6]:642, 2005. 57

[59] Vivek Pradhan and Tathagata Banerjee. Confidence interval of the difference of two independent binomial proportions using weighted profile likelihood. *Communications in Statistics - Simulation and Computation*, **37**[4]:645–659, 2008. 3, 57

[60] Federico Ramponi, Debasish Chatterjee, Sean Summers, and John Lygeros. On the connections between PCTL and dynamic programming. In *HSCC*, pages 253–262. ACM, 2010. 5

[61] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. MIT Press, 2006. 36, 39, 40, 41, 43, 44, 48, 50, 63, 68, 69, 156, 157, 159

[62] Stefan Ratschan. Safety verification of non-linear hybrid systems is quasi-decidable. *Formal Methods in System Design*, **44**[1]:71–90, 2014. 15

[63] Vikas C. Raykar, Shipeng Yu, Linda H. Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. Learning from crowds. *J. Mach. Learn. Res.*, **11**:1297–1322, 2010. 45, 47, 151

[64] Daniel Richardson. Some undecidable problems involving elementary functions of a real variable. *The Journal of Symbolic Logic*, **33**[4]:pp. 514–520, 1968. 15

[65] Filipe Rodrigues, Francisco C. Pereira, and Bernardete Ribeiro. Learning from multiple annotators: Distinguishing good from random labelers. *Pattern Recognition Letters*, **34**[12]:1428 – 1436, 2013. 73

[66] Filipe Rodrigues, Francisco C. Pereira, and Bernardete Ribeiro. Gaussian process classification and active learning with multiple annotators. In *ICML*, pages 433–441, 2014. 6, 47, 151

[67] Laith R. Sahawneh, James Mackie, Jonathan Spencer, Randal W. Beard, and Karl F. Warnick. Airborne radar-based collision detection and risk estimation for small unmanned aircraft systems. *J. Aerospace Inf. Sys.*, **12**[12]:756–766, 2015. 22

[68] Shun Sakuraba and Akio Kitao. Multiple markov transition matrix method: Obtaining the stationary probability distribution from multiple simulations. *Journal of Computational Chemistry*, **30**[12]:1850–1858, 2009. 22

[69] Juljusz Schauder. Eine eigenschaft des haarschen orthogonalsystems. *Mathematische Zeitschrift*, **28**[1]:317–320, 1928. 29

[70] Mohamed Osama Ahmed Sharan Vaswani. Learning from multiple annotators: A survey. *avaliable at: https://core.ac.uk/download/pdf/43577079.pdf*. 48

[71] Victor S. Sheng, Foster J. Provost, and Panagiotis G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*, pages 614–622, 2008. 44

[72] Fedor Shmarov. *Probabilistic bounded reachability for stochastic hybrid systems*. PhD thesis, Newcastle University, UK, 2018. http://hdl.handle.net/10443/4046. 19, 20, 23, 32, 55, 65, 69, 73, 155, 158

[73] Fedor Shmarov and Paolo Zuliani. ProbReach: Verified probabilistic δ-reachability for stochastic hybrid systems. In *HSCC*, pages 134–139. ACM, 2015. 10, 18, 23, 54, 72, 84, 101, 142

[74] Fedor Shmarov and Paolo Zuliani. Probabilistic hybrid systems verification via SMT and Monte Carlo techniques. In *HVC*, **volume 10028** of *LNCS*, pages 152–168, 2016. 2, 5, 6, 9, 10, 12, 13, 14, 15, 16, 18, 21, 56

[75] Fedor Shmarov and Paolo Zuliani. SMT-based reasoning for uncertain hybrid domains. In *AAAI-16 Workhop on Planning for Hybrid Systems*, pages 624–630, 2016. 18

[76] Someshwar Singh and James H. Taylor. Uncertainty estimation in wind power forecasts using monte carlo simulations. In *2018 IEEE Canadian Conference on Electrical & Computer Engineering, CCECE 2018, Quebec, QC, Canada, May 13-16, 2018*, pages 1–6, 2018. 23

[77] Steven S. Skiena. *Sorting and Searching*, pages 103–144. Springer London, 2008. 20

[78] Ilya Sobol. *Multidimensional quadrature formulas and Haar functions*, **4** of *10*. Moscow, Science press, 3 edition, 7 1969. 2, 29

[79] Ilya M. Sobol'. On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Comput. Math. and Math. Phys.*, **7**[4]:86 – 112, 1967. 5, 25, 29

[80] Sören Sonnenburg, Gunnar Rätsch, Sebastian Henschel, Christian Widmer, Jonas Behr, Alexander Zien, Fabio De Bona, Alexander Binder, Christian Gehl, and Vojtech Franc. The SHOGUN machine learning toolbox. *Journal of Machine Learning Research*, **11**:1799–1802, 2010. 6

[81] Sybert H. Stroeve, Henk A.P. Blom, and G.J. (Bert) Bakker. Systemic accident risk assessment in air traffic by monte carlo simulation. *Safety Science*, **47**[2]:238–249, 2009. 22

[82] Alfred Tarski. A decision method for elementary algebra and geometry. *Manuscript, RAND Corp.*, 1948. 15

[83] Tino Teige and Martin Fränzle. *Stochastic Satisfiability Modulo Theories for Non-linear Arithmetic*, pages 248–262. Springer Berlin Heidelberg, 2008. 5

[84] Ilya Tkachev and Alessandro Abate. Formula-free finite abstractions for linear temporal verification of stochastic hybrid systems. In *HSCC*, pages 283–292. ACM, 2013. 5

[85] Joseph Traub, Greg Wasilkowski, and Henryk Woźniakowski. *Information-based Complexity*. Academic Press, 1988. 2

[86] Bruno Tuffin. Randomization of quasi-monte carlo methods for error estimation: Survey and normal approximation. *Monte Carlo Meth. and Appl.*, **10**[3-4]:617–628, 2004. 26, 27

[87] María Dolores Ugarte, T. Goicoa, and Ana F. Militino. Empirical bayes and fully bayes procedures to detect high-risk areas in disease mapping. *Computational Statistics & Data Analysis*, **53**[8]:2938–2949, 2009. 57

[88] Paul S. Wang. The undecidability of the existence of zeros of real elementary functions. *J. ACM*, **21**[4]:586–589, October 1974. 15

[89] Peng Wang, David A. Barajas-Solano, Emil M. Constantinescu, Shrirang Abhyankar, Debojyoti Ghosh, Barry F. Smith, Zhenyu Huang, and Alexandre M. Tartakovsky. Probabilistic density function method for stochastic odes of power systems with uncertain power input. *SIAM/ASA Journal on Uncertainty Quantification*, **3**:24, 2015. 9

[90] Qinsi Wang, Paolo Zuliani, Soonho Kong, Sicun Gao, and Edmund M. Clarke. SReach: A bounded model checker for stochastic hybrid systems. In *Computational Methods in Systems Biology - 13th International Conference, CMSB 2015*, **volume 9308** of *LNCS*, pages 15–27, 2015. 12

[91] Norbert Wiener. *Extrapolation, interpolation, and smoothing of stationary time series, with engineering applications*. Technology Press of the Massachusetts Institute of Technology, Cambridge, 1949. 35

[92] Yan Yan, Rómer Rosales, Glenn Fung, Mark W. Schmidt, Gerardo Hermosillo Valadez, Luca Bogoni, Linda Moy, and Jennifer G. Dy. Modeling annotator expertise: Learning when everybody knows a bit of something. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, pages 932–939, 2010. 46, 151

[93] Håkan L. S. Younes, Marta Z. Kwiatkowska, Gethin Norman, and David Parker. Numerical vs. statistical probabilistic model checking. *STTT*, **8**[3]:216–228, 2006. 4, 54

[94] Håkan L. S. Younes and Reid G. Simmons. Statistical probabilistic model checking with a focus on time-bounded properties. *Inf. Comput.*, **204**[9]:1368–1409, 2006. 4, 54

[95] Hong Zhang, Wenhong Hou, Laurence Henrot, Sylvianne Schnebert, Marc Dumas, Catherine Heusèle, and Jin Yang. Modelling epidermis homoeostasis and psoriasis pathogenesis. *Journal of The Royal Society Interface*, **12**[103], 2015. 93

[96] Vladimir Antonovich Zorich. *Mathematical Analysis II*. Springer, 2nd edition, 2016. 60

[97] Paolo Zuliani, André Platzer, and Edmund M. Clarke. Bayesian statistical model checking with application to Stateflow/Simulink verification. *Formal Methods in System Design*, **43**[2]:338–367, 2013. 32