Data-Driven Methods for Statistical Verification of Uncertain Nonlinear Systems

by

John Francis Quindlen

M.S., The Pennsylvania State University (2012)

B.S.,The Pennsylvania State University (2010)

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
February 2018

Data-Driven Methods for Statistical Verification of Uncertain Nonlinear Systems

by

John Francis Quindlen

Submitted to the Department of Aeronautics and Astronautics
on December 13, 2017, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Due to the increasing complexity of autonomous, adaptive, and nonlinear systems, engineers commonly rely upon statistical techniques to verify that the closed-loop system satisfies specified performance requirements at all possible operating conditions. However, these techniques require a large number of simulations or experiments to exhaustively search the set of possible parametric uncertainties for conditions that lead to failure. This work focuses on resource-constrained applications, such as preliminary control system design or experimental testing, which cannot rely upon exhaustive search to analyze the robustness of the closed-loop system to those requirements.

This thesis develops novel statistical verification frameworks that combine data-driven statistical learning techniques and control system verification. First, two frameworks are introduced for verification of deterministic systems with binary and non-binary evaluations of each trajectory's robustness. These frameworks implement machine learning models to learn and predict the satisfaction of the requirements over the entire set of possible parameters from a small set of simulations or experiments. In order to maximize prediction accuracy, closed-loop verification techniques are developed to iteratively select parameter settings for subsequent tests according to their expected improvement of the predictions. Second, extensions of the deterministic verification frameworks redevelop these procedures for stochastic systems and these new stochastic frameworks achieve similar improvements. Lastly, the thesis details a method for transferring information between simulators or from simulators to experiments. Moreover, this method is introduced as part of a new failure-adverse closed-loop verification framework, which is shown to successfully minimize the number of failures during experimental verification without undue conservativeness. Ultimately, these data-driven verification frameworks provide principled approaches for efficient verification of nonlinear systems at all stages in the control system development cycle.

Thesis Supervisor: Jonathan P. How
Title: R. C. Maclaurin Professor of Aeronautics and Astronautics

Acknowledgments

The past few years have been an interesting and enlightening phase of my life. My time at MIT has introduced me to countless new concepts and ideas that will serve me well in the future and has cultivated my professional development. It's been quite a journey and I'd like to acknowledge the following people for their help and support along the way.

First, I would like to thank my advisor, Professor Jonathan How for his guidance and wisdom. Throughout the years, I was always amazed by his ability to remember the smallest of details from our previous conversations and his breadth of knowledge about all things "control." His inputs would often highlight previously unknown limitations of existing work and help me solidify and buttress the contributions of my research. Through his guidance, I was able to become a significantly better researcher and ask the right questions about a problem.

I would also like to thank my thesis committee, Professors Girish Chowdhary, Ufuk Topcu, and Russ Tedrake, for their invaluable input and support. I've worked with Girish since my first days in ACL and have greatly benefited from his continued advice over the past years. Unbeknown to me at the time, my earliest conversations with Ufuk were some of the most critical as they ultimately led me down the direction taken in this thesis. He helped me understand the strengths and weaknesses of the various analytical and statistical verification techniques and their implementation. I would also like to thank Russ for his help in narrowing down my thesis topic and connecting it to the wider class of applications.

I am very grateful to my thesis readers, Dr. Nghia Trong and Dr. Stefan Bieniawski, for their help in the last stages of this thesis. Nghia has not been at MIT long, but already I've picked his brain about all things active learning and statistical inference. Likewise, Stefan was a fantastic supervisor during my internship at Boeing and that experience also directly motivated this thesis research. Myself and this work have continued to benefit from his insights on industry's challenges and procedures with respect to flight control system design and verification.

Without my labmates and friends, I would never have made it to this point. I would like to thank my labmates past and present - Brett Lopez, Steven Chen, Justin Miller, Mark Cutler, Shayegan Omidshafiei, Mike Everett, Kris Frey, and everyone else - for their collaboration on psets, advice on research, funny banter, and helping me stay grounded. They made our office space as enjoyable as possible, especially during all those late nights and weekends. I'm glad to have my friends Tony Tao, Matt Keffalas, Jess Hebert, and Andrew Owens for their support here at MIT. I appreciate all those Friday nights where we would unwind and watch movies or play games after a long week. The same goes for my old roommates Bobby Klein and David Clifton. I also need to thank my friends from Penn State who are up in the greater Boston area: Alan Campbell, Andrew Weinert, Charlie Cimet, and Pei Wang. I would have never guessed we would still be living in the same area after our time in undergrad. I also can't forget my best friends scattered across the country: Joe Wood, Matt Sowa, and Henry Ball. All these friends got me through the up and downs of grad school (and there were many downs).

Finally, I would like to thank my family for all their unending support. My parents have always encouraged me to pursue my passions, whatever they were, and worked tirelessly for my sister and I. There is no way I would be in my current position without them and I am eternally grateful.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Control systems are employed to ensure an open-loop system adequately satisfies a set of performance objectives. For instance, possible open-loop plants under consideration may range from localized subsystems like a computer hard-drive to physical vehicles such as an aircraft or automobile, or even higher-level system-of-systems like a team of interacting robotic agents. All of these applications will typically require some form of control system and more complex systems-of-systems will include multiple layers of control.

As the demand for higher performance, efficiency, and autonomy grows, advanced control techniques will be increasingly relied upon to meet these demands. Complex methods such as model reference adaptive control (MRAC) [1], reinforcement learning [2], and a variety of other robust nonlinear control architectures [3] will replace traditional linear control techniques which simply cannot meet the demands. These advanced control approaches have already experimentally demonstrated the ability to control damaged aircraft [4–6], aerobatic helicopters [7–9], and drifting racecars [10], and have been proposed as key enablers for brand new applications such as hypersonic [11] and lightweight flexible aircraft [12,13].

The largest obstacle impeding wider acceptance and implementation of advanced control techniques is the lack of trust in the robustness of the closed-loop system. The complexity and nonlinearity of the control architectures which allow them to perform so well also complicates predictive analysis of the closed-loop system's trajectory. For instance, reinforcement learning will eventually converge towards a locally-optimal controller, but there is no guarantee an intermediate controller will not cause catastrophic failure let alone satisfy the performance requirements. The difficulty in analyzing the robustness is further compounded by the fact the systems of interest are usually expected to operate over a wide range of possible conditions with little-to-no human oversight. Given the various conditions and nonlinear evolution of the states, it is extremely difficult to guarantee all the possible trajectories will meet the objectives. Until the approach is shown to either work robustly or gracefully degrade, any new control architecture is of rather limited utility even if experimental results point to large potential improvements. In fact, the Office of the US Air Force Chief Scientist [14] recently stated that "establishing certifiable trust in autonomous systems is the single greatest technical barrier that must be overcome to obtain the capability advantages that are achievable by the increasing use of autonomous systems" and "this level of autonomy will be enabled only when automated methods for verification and validation of complex, adaptive, highly nonlinear systems are developed." This thesis develops data-driven, black-box methods for statistical verification of nonlinear systems without the need for human supervision or restrictive modeling assumptions. The chapters will demonstrate these new verification techniques on multiple examples with adaptive, nonlinear, or otherwise complex control systems.

1.1 Motivation

The fundamental goal of control system verification is to identify at which operating conditions the closed-loop system will satisfy a certain set of performance requirements and at which conditions it will not. These requirements may cover a wide range of possible criteria such as simple concepts like stability and boundedness of the states to more complex functions of state and time. These well-defined requirements are provided by relevant certification experts or authorities, such as military [15] and civilian [16] aviation agencies. Regardless of the exact specifics, the closed-loop system must successfully meet all the given requirements in order for the control system to be considered "satisfactory" for final implementation.

There are multiple approaches that can be taken towards verification, but they typically fall into two general categories: deductive analytical techniques and statistical methods. Analytical verification approaches encompass proof-based certificates or numerically-exact solutions which provably guarantee the closed-loop system will satisfy the requirements under specific modeling assumptions. In contrast, statistical techniques relax the modeling assumptions and apply to a wider class of systems, but replace the provable guarantees with less absolute probabilistic bounds. Despite their implementation differences, both these verification approaches must contend with the same issues and considerations.

First, closed-loop systems are expected to successfully meet a wide range of different performance criteria. As discussed earlier, these requirements may include everything from stability to nonlinear functions of space and time, but the closed-loop systems may also have to simultaneously address multiple requirements with potentially competing objectives. For instance, high performance aircraft such as a F-16 fighter jet will have to satisfy high speed and maneuverability requirements to complete the assigned combat missions, but also meet slow speed and docile handling requirements for landing [17]. Given a set of competing requirements, the closed-loop system will likely not be able to satisfy all the requirements by a large margin. Ultimately, it is not immediately obvious whether the closed-loop system will meet those requirements and verification will be a nontrivial endeavor.

Second, verification requires a sufficiently-realistic representation of the actual system to capture the change in performance at various conditions. For analytical verification problems, a sufficiently-realistic representation would require the full set of equations of motion of the true system to be known or approximated. For statistical verification, this representation is typically a simulation model of the closed-loop system dynamics, but could also include a physical prototype for experiment-based testing. In the statistical verification case, simulation models will have to be of high-enough fidelity to meet the certification authority's acceptable level of realism. While certain applications may allow simple equations of motion for verification purposes,

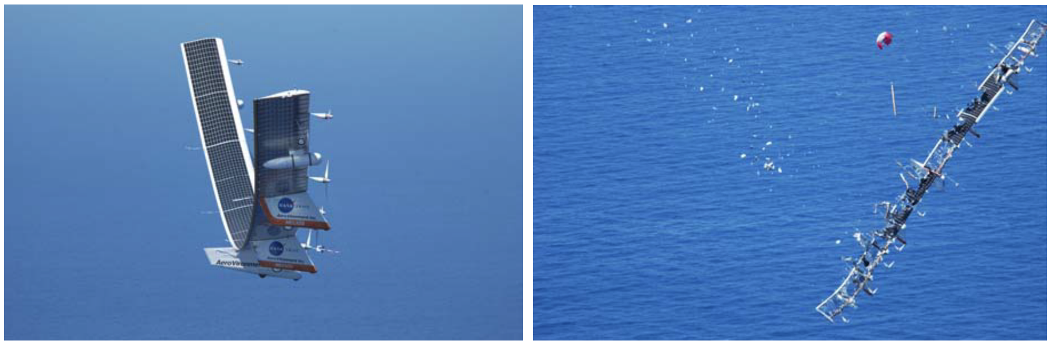
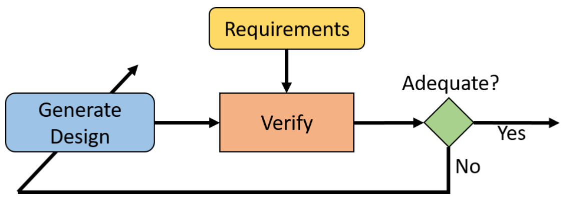

(a) Helios before breakup (b) Helios after exceeding failure limits

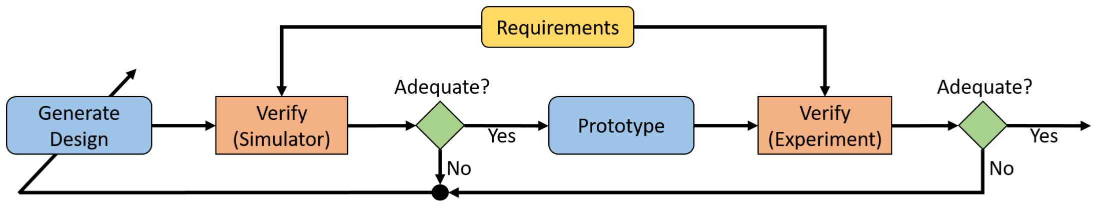
Figure 1-1: NASA Helios aircraft which broke apart due to failure modes of the flight control system inadvertently missed by linear verification methods. Image source [23]

the minimum level of realism for most applications will generally require complex simulators with models of full nonlinear dynamics and relevant saturation, logic, and switching modes. It may even require surrogate models with the most realistic depiction of a physical system as possible, such as FAA-approved flight-training simulators that are judged realistic enough to *replace* real-world flight hours for airline pilot training and currency [18].

Likewise, the verification model may be more complex than the model used to construct the control policy under consideration. For instance, many control systems are designed using a reduced-order model of the full system dynamics [19–21]. While this eases design and optimization of the controller, verification should be performed on the full-order model rather than the simplified representation. For instance, one of the root causes leading to the crash of NASA's Helios Unmanned Aerial Vehicle (UAV) was the lack of control system verification on a nonlinear model with interactions between aircraft subsystems and the effects of different meteorological conditions [22, 23]. The reliance upon linear methods "did not provide the proper level of complexity to understand the technology interactions on the vehicle's stability and control characteristics" [22] and missed failure modes for the flight control system that ultimately led the lightweight, flexible aircraft to break apart (Figure 1-1). The end result highlights the need for the verification model to adequately capture the dynamics that adversely affect the performance of the actual system.



(a) Simple control design process



(b) Simulation and experimental verification within a more complex design process

Figure 1-2: Verification as a step within a broader control system design process.

Lastly, verification is usually one step in a much larger control system design and optimization process. For instance, consider the generic iterative control design process shown in Figure 1-2(a). Verification is used to identify whether the candidate controller produces an acceptable level of robustness to uncertainties. If the system fails to meet the minimum level of robustness or some other objective function, the process is repeated until a suitable controller is produced. Although the specific implementation details differ, various control design works [24–26] employ verification methods within some form of similar iterative process. Likewise, verification may even occur at multiple stages within a control policy design cycle, as shown in Figure 1-2(b). For example, a low-cost UAV design procedure used simulation and hardware-in-the-loop testing to prune out poorly-performing control system designs earlier in the process before experimental flight testing [27]. This multi-stage verification approach could be extended even further to include multiple simulation models of increasing fidelity in conjunction with hardware-based testing. A similar approach has already been used for multi-fidelity reinforcement learning [28] and would easily transfer to verification applications. All these various examples simply serve to highlight that verification tends to be performed multiple times over the course of a control system design and optimization process.

1.2 Problem Statement

The goal of this work is to develop data-driven methods for statistical verification of nonlinear closed-loop systems. While analytical verification techniques provide provable guarantees, their restrictive modeling assumptions and conservativeness limit their utility and availability in complex nonlinear systems. For example, the large state and parameter spaces associated with industrial applications challenge analytical methods [29, 30]. In many of these applications, simulation-based statistical methods are significantly easier and faster to perform than it is to compute an analytical solution, if that is even feasible [30, 31]. Some analytical methods are able to scale to arbitrarily complex systems; however, they typically require very restrictive or conservative assumptions and abstractions to achieve that result. While they do produce a solution, the resulting guarantees may not reflect the full response of the actual system. Therefore, statistical verification is often a more suitable approach towards verification of arbitrary closed-loop systems with adaptive, nonlinear, or complex control systems.

1.2.1 Challenges

The implementation considerations discussed in Section 1.1 illustrate a number of challenges associated with verification of complex nonlinear systems. These challenges present major obstacles or hindrances to existing statistical verification procedures and motivate the approach taken in this thesis.

The primary challenge is the large overhead cost required for statistical verification using full nonlinear simulation models or, to an even greater extent, real-world experimentation. One of the leading reasons for this is the large range of possible operating conditions faced by the system. In aerospace applications, the operating conditions include various uncertainties such as weight and center of gravity location. Even when the number of uncertain variables is small, these terms span a continuous spectrum of potential values rather than a small, discrete set of possible conditions. Similarly, the aforementioned need to employ sufficiently-realistic models also contributes to the overhead cost in model-based verification. In order to adequately capture the interaction of the uncertainties with the system dynamics, verification requires a high-fidelity simulation model and a small numerical integration timestep for the simulations. These simulation models require significant computational resources and may take several minutes on a suitable computer to generate a single simulation test [32]. These issues will only be further exacerbated when verification is part of an iterative process (Figure 1-2).

Additionally, one secondary challenge faced during verification is stochasticity and the randomness it introduces into the system dynamics. Stochasticity is present in many physical systems, commonly as process and/or measurement noise in the underlying dynamics. While the system's trajectories will still vary according to the operating conditions, the stochastic noise terms will also affect the evolution of the states. Due to the randomness introduced by stochasticity, no two simulations or experiments will ever be the same, even if they are performed at the same parameter setting. In fact, when multiple trajectories are performed at the same operating condition, some trajectories may satisfy the designed requirement(s) while others may not, as shown later in Figure 5-1. Ultimately, the fact multiple trajectories at the same operating condition may produce different results means a stochastic system will no longer always satisfy or fail to satisfy the designated requirement at a particular operating condition, but will instead have a *probability* of satisfying the requirement.

Lastly, the incorporation of prior verification work in the later stages of a multi-stage verification analysis (Figure 1-2(b)) is another secondary challenge. In those problems, it is desirable from a cost perspective to transfer the results from the preceding stages into later stages and speed up the process; the question is how to do this in a correct manner. The differences in the dynamics between lower- and higher-fidelity simulation models will cause inaccuracies in the certification output from the lower-fidelity model. The same type of inaccuracy is experienced between simulation models and experimental results. These inaccuracies are usually slight but their existence does complicate forward transfer of verification output from an earlier stage of the process into later stages.

1.3 Literature Review

There exists a wide range of verification techniques that have been used to address some form of control system verification or testing. This section will overview these various techniques and discuss their merits or limitations with respect to the verification challenges identified in Section 1.2.1.

1.3.1 Analytical Verification Methods

Analytical verification encompasses a variety of disparate approaches with either closed-form or numerically-exact solutions for theoretically-proven robustness. For discrete-time dynamical systems, methods originally developed for rapidly-exploring random trees (RRTs) can be used to analyze the closed-loop trajectories from simple linear or linearized nonlinear models. In particular, chance-constrained, closed-loop RRT (CC-CL-RRT) methods [33–35] are able to efficiently propagate the effects of stochastic noise in closed-loop trajectories by placing Gaussian distributions around a nominal trajectory to model the stochasticity in the system's states. Although this has been demonstrated on relevant aerospace applications [36], the method's applicability is limited due to it's reliance upon linear or linearized models. More importantly, there is no readily-apparent extension to incorporate parametric uncertainties in non-state variables and the verification analysis would have to be repeated for each realization of the parameterized, linear model.

A second type of approach focuses on finite abstractions of the nonlinear closed-loop dynamics. These approaches either assume the closed-loop dynamics are presented as a finite transition system or approximate the dynamics in such a way. For example, Markov chain analysis methods [37,38] assumes the set of all possible states is a n-dimensional hypercube (or similar shape) and models stochasticity in the dynamics as transition probabilities between those states. Similar common verification tools such as SpaceEx [39] approximate the state space with a finite lattice and compute the approximate reachability of a hybrid system. Although more recent work [40] has applied parametric transition systems to discretize both the state and parameter spaces in adaptive control problems, the utility of finite transition systems is limited by the discretization of the dynamics. As the complexity of the system grows, the level of discretization required becomes intractable and further exacerbated in high-dimensional systems.

Satisfiability modulo theorem (SMT) solvers [41, 42] verify the δ-satisfiability of logic requirements in overapproximations of the reachable state space. These techniques construct proofs to provably guarantee whether a set of requirements will be satisfied given nonlinear differential equations representing the system's closed-loop dynamics. The approach has also been extended to include stochastic systems [43]. Even though these techniques have been shown to handle polynomial and logrithmic nonlinear systems with simple discrete mode transitions, they fail to scale well to higher-dimensional and -complexity systems and suffer from the same issues with uncertain parameters as the previous approaches.

The most relevant approach for verification of complex, uncertain nonlinear systems is bounding function-based analysis. As the name suggests, these techniques rely upon analytical functions such as Lyapunov, barrier, or storage functions to bound the reachability of the system's trajectory. For instance, the Lyapunov function common to MRAC systems defines a convex, quadric hypersurface that bounds the reachable set of states from a given set of initial conditions and parametric uncertainties [1, 44]. More advanced methods like barrier certificates [45–47] and LQR trees [48] can be constructed for a variety of nonlinear systems with stochasticity and complex performance requirements. While these analytical techniques produce extremely powerful certificates that provably verify the set of reachable states, they can be difficult to construct for arbitrary nonlinear systems [49]. The existence and conservativeness of the certificate is tied to the choice in Lyapunov function, but the correct or best function representation may not be known in advance. Likewise, as the complexity or number of states and parameters increases, the minimum complexity for an appropriate Lyapunov function also increases and complicates finding a useful verifying certificate. Recent work [29] has been able apply these techniques to higher-dimensional systems, but does so by introducing additional conservatism.

Simulation-Guided Analytical Verification Methods

In order to more easily find and construct analytical certificates, simulation-guided barrier certificates [50–53] and LQR trees [54] were developed to automate the verification process. As the choice of function representation is inherently coupled to the conservativeness of the certificate, but the best choice is generally unknown, it is difficult to compute a maximizing bounding set without prior knowledge or experience. These methods use simulation traces and the gradient of candidate Lyapunov functions to determine suitable Lyapunov functions and calculate their maximizing invariant set, effectively automating the process. These techniques also serve as a bridge between analytical verification methods and statistical ones.

1.3.2 Statistical Verification Methods

On the opposite side of the spectrum, statistical verification methods are brute-force approaches that return statistical certificates or bounds based upon large sets of trajectory data. In comparison to analytical techniques, statistical approaches are much more straightforward, but usually more data intensive. In extreme cases, the system can be treated as a black-box model with zero information on the internal model structure/dynamics.

One popular, recent development is falsification-guided software tools such as Breach [55] or S-TaLiRo [56,57]. These approaches use temporal logic properties and nonlinear optimization to intelligently search for counterexamples to a given performance requirement and simulation model. Although they have been demonstrated on multiple relevant applications, falsification approaches address a similar, but different type of problem. Rather than identify the set of operating conditions for which the system will satisfy the performance criteria, falsification methods utilize optimization techniques to converge towards a *single trajectory* (counterexample) of the system that fails to meet the requirement. This underlying optimization process controls falsification-guided testing and will direct the search towards areas with lower robustness; however, falsification methods may encounter problems similar to those seen in non-convex optimization. Just as optimization may repeatedly fall into the same local optimums, falsification searches may inadvertently return the same counterexample, even if they are initialized at different starting points. While these methods are extremely useful for quickly finding a single counterexample, they are not perfectly suited to problems where multiple counterexamples exist and the goal is to identify the entire set of unsatisfactory operating conditions. For this reason, falsification-guided techniques are of limited use to the particular problem of interest in this thesis.

The most general, widely-used, and versatile approaches are Monte Carlo methods [30,31,58–69]. Part of the reason for their popularity and versatility is their simplicity: they randomly generate a large number of trajectories and reason about the system's robustness from this finite number of observations. Monte Carlo methods are practical tools for measuring the effects of stochasticity [58–60] and parametric uncertainties [25,26,61,70] in the dynamics and have been used in conjunction with various tools like S-TaLiRo [58,62,63], PRISM [64,65], and other statistical model checking approaches [30,31,66–69]. The main drawback of Monte Carlo methods is that they rely upon the law of large numbers to provide bounds or statistical estimates [59,60], meaning a large amount of data is required. Importance sampling and cross-entropy methods [60, 61, 63,67] have been developed to reduce the number of samples required for statistically-relevant results, but are still inherently random and require many samples.

Additional techniques like Dirichlet Sigma point processes [24], set-oriented models [71], and box thresholding [72] all attempt to circumvent the limitations of Monte Carlo approaches with finite, structured groupings. Rather than randomly sample from the set of all operating conditions, these approaches select some subset of conditions explicitly chosen for their perceived informativeness about the results over the full set. Design of experiments techniques [73, 74] can also be considered to roughly fall within this category or the intersection between it and Monte Carlo methods.

While these approaches offer an alternative to pure Monte Carlo methods, they can still be viewed as sample inefficient because they rely upon pre-generated grids or lattices covering the set of all possible operating conditions. These structured groupings will typically require a fine discretization or some equivalent in order to observe the changes in performance satisfaction with adequate resolution and will therefore require a nontrivial number of simulations or experiments.

The closest approach to the work in this thesis are the Gaussian process-based methods for safe learning of regions of satisfactory performance [75, 76]. These methods combine Lyapunov analysis from analytical verification techniques with Bayesian optimization to efficiently estimate the region verified by a Lyapunov function-based barrier certificate. While these methods straddle the line between analytical and statistical verification techniques, they ultimately break analytical techniques' fundamental assumption of provable guarantees of closed-loop performance with the use of Bayesian predictions. This fact shifts the methods' implementation closer to statistical techniques, but their reliance upon a given Lyapunov function for verification of the system causes them to suffer the same limitations as analytical techniques. In effect, these methods are useful for searching the set of uncertainties to estimate the limits of a barrier certificate, but are not able to provide provable guarantees like the aforementioned analytical methods nor do they posses the same versatility of most statistical methods.

One important observation is that all of the discussed approaches, both analytical and statistical methods, do not present an explicit solution to the challenge of forward transfer of predictions from prior verification stages in a multi-stage process. Although these earlier stages indirectly aid later steps by pruning out unsatisfactory candidate controllers, none of the approaches posses a mechanism for direct incorporation of previous results. In fact, many of these approaches will lose all guarantees once applied to a different model. For instance, barrier certificates are coupled to the model of the system dynamics used to construct the proof and lose any guarantees as soon as the later stage's dynamics deviate from the original equations. The results from proceeding stages can be used to inform initializations of later steps, but there is no mechanism to transfer the output. Effectively, the later verification procedure will be performed without any direct knowledge of the earlier predictions or proofs and will have to replicate those results. This presents an obvious inefficiency and can prove extra costly in experiment-based verification if the later experiment-based stage crashes or destroys a prototype at an operating condition that was already identified as dangerous by the preceding stage(s).

1.4 Summary of Contributions

This thesis proposes a unified framework to address all the aforementioned challenges with statistical verification. At a high level, this work combines control system verification with machine learning to produce novel, data-driven procedures for efficient statistical verification in resource-constrained environments. The thesis is structured around three sets of major contributions corresponding to 1) the baseline problem with verification of deterministic systems, 2) verification of stochastic systems, and 3) multi-stage verification (Figure 1-2(b)) with multiple, sequential verification steps. These contributions build upon one another to address the fundamental challenges in Section 1.2.1 that plague efficiency and tractability in statistical verification of complex, nonlinear control systems.

The first major set of contributions concentrates on verification of deterministic systems where each reinitialization of the trajectory will produce the same exact result. This work shows deterministic verification is ultimately a binary classification problem - a trajectory will either satisfy a performance requirement or it will not - and introduces new machine learning algorithms aimed at that problem. The contributions for deterministic verification are as follows:

- The development of a new machine learning framework for statistical verification called data-driven verification certificates. Unlike barrier certificate methods, this framework directly translates raw trajectory data into a predictive certificate without intermediate analytical, proof-based steps. These data-driven certificates sacrifice the conservative guarantees of analytical certificates in order to apply to a significantly wider class of systems. Chapters 3 and 4 introduce two parallel modeling techniques each tailored to exploit one of the two most common sources of feedback on a trajectory's performance.

- The introduction of validation techniques for online quantification of prediction accuracy. Section 3.3.1 extends traditional machine learning methods for model validation to data-driven verification certificates and analyzes their effectiveness in statistical verification. Most importantly, Section 4.2.2 designs a completely new approach for online computation of prediction confidence without reliance upon external validation datasets or retraining of the prediction model. This latter validation technique minimizes the amount of additional computational overhead and guarantees the predictive certificate will explicitly answer both relevant questions for each query:

  - Is the trajectory going to satisfy the requirements?
  - How confident is the model in that prediction?

  The second question is almost as important as the prediction itself, but is frequently not addressed in statistical verification and relevant machine learning techniques.

- The development of closed-loop verification algorithms to maximize the accuracy of prediction certificates while limited to fixed sample budget. A major part of this contribution is the introduction of new entropy-based selection metrics (Section 4.3) to evaluate the informativeness of prospective trajectories and select future training samples to produce the largest expected improvement in prediction confidence. Several examples demonstrate closed-loop verification's improvement in prediction accuracy over comparable analytical verification techniques, open-loop (non-iterative) verification approaches, and competing iterative procedures adapted from existing machine learning methods.

The second set of contributions extends data-driven verification to stochastic systems. Due to the introduction of stochasticity, the preceding work in deterministic certificates will no longer apply without heavy modification. This set of contributions reproduces the same high-level concepts behind data-driven certificates and closed-loop verification for the separate and distinct stochastic problem. These contributions include:

- The development of data-driven verification frameworks to model the probability of satisfaction or failure in stochastic systems. Specifically, Chapters 5 and 6 introduce different approaches to model the various distributions of performance feedback evaluations possible with stochastic dynamics. These techniques all predict the likelihood of an unsatisfactory trajectory given a small set of individual trajectories (Chapter 5) or limited groups of repeated trajectories (Chapter 6).

- The redevelopment of closed-loop verification algorithms to address the changes with stochastic systems. Sections 5.2 and 5.4 introduce new selection criteria to rank prospective trajectories based upon their expected reduction in prediction error. Several examples in Sections 5.6 and 6.3 again demonstrate the improvement in prediction accuracy over open-loop verification approaches afforded by closed-loop verification and the novel selection metrics' further improvement over relevant procedures adapted from existing machine learning methods.

The last major set of contributions in Chapter 7 builds upon the preceding chapters to address implementation issues in multi-stage verification. These contributions mainly focus on the value of prior verification work during later stages of the process and its use to further maximize accuracy in the presence of restrictive testing constraints. The contributions are:

- The demonstration of a principled approach for forward transfer of earlier verification analysis of the same control policy on a verification model from an earlier stage. In particular, Section 7.1 details the use of nonzero-mean priors taken from the output of earlier prediction models to explicitly incorporate their effects into later stages. Simultaneously, this approach avoids naïve assumptions about the accuracy of earlier, lower-fidelity models and allows for posterior prediction models to vary drastically from their priors.

- The introduction and formalization of a variant of multi-stage verification called failure-constrained statistical verification (Section 7.2). This subproblem considers the challenge of statistical verification in experimental domains where unsatisfactory trajectories lead to unacceptable costs. Failure-constrained statistical verification places limits on the maximum allowable number of failures during testing at the experimental stage.

- The development of new closed-loop verification algorithms for failure-constrained statistical verification. Section 7.3 introduces two novel algorithms, one adaptive and one static, to simultaneously minimize the number of failures while maximizing informativeness of the prediction model. Section 7.3 also introduces a new selection criteria specifically tailored to maximize the informativeness of each experiment while limiting the likelihood of failure. Results in Section 7.4 demonstrate the limitations and unacceptable costs of existing procedures when applied to failure-constrained statistical verification and the improvement offered by the new algorithms.

Chapter 2

Background

This chapter introduces and details a number of tools used throughout the thesis. These sections are intended to provide basic background and motivation for the tools as well as present implementation details. Each subsequent chapter will discuss its particular application of these tools and any relevant extensions.

2.1 Support Vector Machines

Support vector machines (SVMs) are one of, if not the, most common tools for binary classification [77–82]. In binary classification problems, the SVM will classify all elements of the feature set $\Theta \subset \mathbb{R}^p$ as either an exclusive element of set Θ_- or its complement Θ_+, where $\Theta = \Theta_- \cup \Theta_+$ and $\Theta_- \cap \Theta_+ = \emptyset$. This work assumes there are only two classes (Θ_-, Θ_+), although other work has considered SVMs for multi-class classification [81, 83, 84]. Additionally, set Θ, the set of all feasible features, may be either discrete/countable or uncountable. As will be discussed later in the thesis, this work assumes the feasible set Θ is an uncountable set, but will accurately approximate it with a finely-discretized countable set.

In short, a support vector machine is a supervised learning technique that constructs an optimum maximum-margin classifier with respect to a set of training data. This training dataset consists of N input vectors $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_N\}$ with corresponding binary targets $\{y_1, y_2, \ldots, y_N\}$, where $y_i \in \{-1, +1\}$ for $i = 1, \ldots, N$. The constructed classifier will then predict the label for any arbitrary element of Θ, vector $\boldsymbol{\theta} \in \Theta$.

2.1.1 Linear Classifiers

In the simplest form of the problem, it is assumed label y_i is the output of a linear model

$$H_{\mathbb{R}}(\boldsymbol{\theta}) = \mathbf{w}^T\boldsymbol{\theta} + b, \tag{2.1}$$

with decision function

$$H(\boldsymbol{\theta}) = \operatorname{sign}(\mathbf{w}^T\boldsymbol{\theta} + b) \tag{2.2}$$

such that $y_i = H(\boldsymbol{\theta}_i)$. Elements with $\mathbf{w}^T\boldsymbol{\theta} + b > 0$ are said to belong to set Θ_+, while those with $\mathbf{w}^T\boldsymbol{\theta} + b < 0$ belong to set Θ_-. A linear optimization program can then be used to compute optimal $\mathbf{w}$ and b with respect to the training dataset. These terms define a hyperplane in the $\mathbb{R}^p$ space that is used to separate $\boldsymbol{\theta}$ datapoints. In this simplest form, the data is assumed to be linearly separable, meaning there exists two parallel hyperplanes that separate all points with $y(\boldsymbol{\theta}) = +1$ from those with $y(\boldsymbol{\theta}) = -1$. The optimal $\mathbf{w}$ and b then correspond to the unique maximum-margin hyperplane, which minimizes the distance between the decision boundary given by the hyperplane and any of the training samples [81]. As mentioned earlier, these optimal terms can be computed via the following primal linear optimization problem:

$$\begin{aligned} \underset{\mathbf{w},b}{\text{minimize}} \quad & \frac{1}{2}||\mathbf{w}||^2 \\ \text{such that} \quad & y_i(\mathbf{w}^T\boldsymbol{\theta}_i + b) \geq 1 \quad i = 1, \ldots, N. \end{aligned} \tag{2.3}$$

2.1.2 Nonlinear Classifiers

In practice, the linear model and classifier are not adequate for all types of problems and applications, particularly the ones considered in this thesis. The nonlinearities in dynamical systems often result in nonconvex regions in $\mathbb{R}^p$ which simply cannot be handled by linear models. The following subsection will discuss nonlinear classification models as well as modifications to handle the lack of linear separability in the training dataset.

In place of the linear model from (2.1), nonlinear classifiers can be constructed from nonlinear basis functions that are functions of the input vector $\boldsymbol{\theta}$. This new representation is given by

$$H_{\mathbb{R}}(\boldsymbol{\theta}) = \mathbf{w}^T \boldsymbol{\phi}(\boldsymbol{\theta}) + b, \tag{2.4}$$

where $\boldsymbol{\phi}(\boldsymbol{\theta}) \in \mathbb{R}^q$ is a vector of basis functions and typically $q > p$. The decision function $H(\boldsymbol{\theta})$ and the resulting classification rule for elements of Θ_- and Θ_+ are mostly unchanged from (2.2), with basis vector $\boldsymbol{\phi}(\boldsymbol{\theta})$ replacing $\boldsymbol{\theta}$. The new primal optimization program is also similar to (2.3):

$$\begin{aligned} \underset{\mathbf{w},b}{\text{minimize}} \quad & \frac{1}{2}||\mathbf{w}||^2 \\ \text{such that} \quad & y_i(\mathbf{w}^T \boldsymbol{\phi}(\boldsymbol{\theta}_i) + b) \geq 1 \quad i = 1, \ldots, N. \end{aligned} \tag{2.5}$$

In addition to nonlinear models, the baseline SVM classifier can be modified to accommodate datasets which are not linearly separable, as exact separation of the training data by hyperplanes is not possible in many applications [81]. Instead, a soft-margin nonlinear classifier will allow datapoints to fall on the "incorrect" side of the decision boundary, thus enabling a solution where the previous hard-margin SVM would fail. To accommodate possible inseparability in the dataset, a non-negative slack variable ξ_i is introduced for each of the N training points to relax the constraints from (2.5). While these slack variables allow for misclassifications, the objective function is also modified with a hinge-loss function in order to penalize these misclassifications and minimize their presence in the optimal solution. The resulting soft-margin primal problem is given by the following quadratic program:

$$\begin{aligned} \underset{\mathbf{w},b}{\text{minimize}} \quad & \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{N} \xi_i \\ \text{such that} \quad & y_i(\mathbf{w}^T \boldsymbol{\phi}(\boldsymbol{\theta}_i) + b) \geq 1 - \xi_i \quad i = 1, \ldots, N \\ & \xi_i \geq 0 \qquad\qquad\qquad\qquad\quad i = 1, \ldots, N \end{aligned} \tag{2.6}$$

with scalar $C > 0$. This scalar term C controls the likelihood of misclassifications and with a sufficiently-high penalty on non-zero ϵ_i, the soft-margin classifier begins to operate similar to a hard-margin SVM. Thus, the soft-margin nonlinear SVM is the most general form of classifiers and is capable of handling all the problems previously addressed by the hard-margin, linear and nonlinear SVMs. Due to the fact little-to-nothing is assumed in advance about the performance of the nonlinear systems of interest in this thesis, soft-margin nonlinear SVMs offer the most robust solution with the widest applicability.

Rather than compute the solution using the primal optimization problem in (2.6), it is generally easier to solve the problem in its dual form:

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\text{maximize}} \quad & \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \ \alpha_j \ y_i \ y_j \boldsymbol{\phi}(\boldsymbol{\theta}_i)^T \boldsymbol{\phi}(\boldsymbol{\theta}_j) \\ \text{such that} \quad & 0 \leq \alpha_i \leq C \quad i = 1, \ldots, N \\ & \sum_{i=1}^{N} \alpha_i y_i = 0, \end{aligned} \tag{2.7}$$

with weighting terms now set to

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i \ y_i \ \boldsymbol{\phi}(\boldsymbol{\theta}_i). \tag{2.8}$$

As the dimension q of the feature space associated with $\boldsymbol{\phi}(\boldsymbol{\theta})$ grows, computing the mapping $\mathbf{w}^T \boldsymbol{\phi}(\boldsymbol{\theta}) + b$ becomes increasingly computationally demanding. Instead, the "kernel trick" [79] is used to reduce this cost through a kernel function $\kappa(\boldsymbol{\theta}, \boldsymbol{\theta}') : \mathbb{R}^q \times \mathbb{R}^q \to \mathbb{R}$, defined such that

$$\kappa(\boldsymbol{\theta}, \boldsymbol{\theta}') = \boldsymbol{\phi}(\boldsymbol{\theta})^T \boldsymbol{\phi}(\boldsymbol{\theta}). \tag{2.9}$$

This use of the kernel function enables a solution to the dual problem in (2.7) to be found *without* having to work in the potentially high-dimensional feature space for $\boldsymbol{\phi}(\boldsymbol{\theta})$. Various forms of the mapping and kernel functions exist, such as polynomial, hyperbolic, and Gaussian radial basis function (RBF) [77, 79, 81]. In this work, Gaussian RBFs are used unless otherwise noted. These RBF kernels are given by

$$\kappa(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) = \exp(-||\boldsymbol{\theta}_i - \boldsymbol{\theta}_j||^2/\gamma), \tag{2.10}$$

with $\gamma > 0$. Finally, the decision function for the soft-margin, nonlinear SVM in its dual form is given by

$$H(\boldsymbol{\theta}) = \text{sign}\Big(\sum_{i=1}^{N} \alpha_i y_i \kappa(\boldsymbol{\theta}, \boldsymbol{\theta}_i) + b\Big). \tag{2.11}$$

Note that not all of the N training points are considered "active." Only a subset of the training points are selected as support vectors ($N_{sv} \leq N$); all remaining points can be considered to have $\alpha_i = 0$ and are thus not active. In practice, only the active subset of N_{sv} support vectors are used for predictions, improving computational efficiency. For the remainder of this thesis, the SVM optimization problem and its decision function will be shown and discussed in the format of the dual representation from (2.7) and (2.11).

2.2 Gaussian Process Regression Models

Gaussian process (GP) regression models, sometimes known as Kriging, are Bayesian nonparametric regression tools for modeling scalar target variables across a continuous input space [85–87]. Gaussian processes have been widely used throughout a variety of relevant applications such as adaptive control [88, 89], reinforcement learning [90, 91], and optimization [92]. In this thesis, GPs are used to model measurements of performance requirement satisfaction by the closed-loop system under consideration. The following material will provide background on the construction of GP regression models and explain certain steps which will be discussed later in the thesis.

While a Gaussian process is formally defined as the joint Gaussian distribution of a finite collection of random variables, it can more easily be thought of as a distribution over possible functions for $h(\boldsymbol{\theta})$. Here, $h(\boldsymbol{\theta})$ is a real, scalar function with input vector

$\boldsymbol{\theta} \in \mathbb{R}^p$. Additionally, the GP is completely defined by a scalar mean function $m(\boldsymbol{\theta})$ and covariance function $\kappa(\boldsymbol{\theta}, \boldsymbol{\theta}')$ such that

$$h(\boldsymbol{\theta}) = GP\big(m(\boldsymbol{\theta}), \kappa(\boldsymbol{\theta}, \boldsymbol{\theta}')\big). \tag{2.12}$$

The overall aim of the GP regression model is to infer the true, but unknown, function $h(\boldsymbol{\theta})$ from a limited number of sample locations $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_N\}$ and corresponding observations of $h(\boldsymbol{\theta})$, labeled $y(\boldsymbol{\theta}_i)$. Both the cases where $y(\boldsymbol{\theta}_i)$ are noisy and noiseless measurements of $h(\boldsymbol{\theta}_i)$ are considered and will be discussed later. The resulting GP regression model can then be used to predict function values $h(\boldsymbol{\theta})$ at unobserved input vectors $\boldsymbol{\theta}$. Rather than specifying a particular parameterized form of $h(\boldsymbol{\theta})$ and attempting to infer the correct coefficients, the GP model treats $h(\boldsymbol{\theta})$ as a random function. This allows the GP to be free of any restriction to a parameterized form, assuming one even exists, and improves its applicability.

2.2.1 Training

At its core, Gaussian process regression relies upon Bayesian inference to construct the predictive model. Model training is performed according to Bayes' rule with a prior probability distribution and likelihood model of the data used to compute a posterior probability distribution. In this problem, the posterior probability distribution defines the distribution of $h(\boldsymbol{\theta})$ values conditioned on the evidence provided by the training dataset of N sample locations and corresponding observations mentioned earlier. For simplicity, this training set is labeled as $\mathcal{L} = \{\mathcal{D}, \mathbf{y}\}$ where set $\mathcal{D} = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N\}$ contains the input locations and vector $\mathbf{y} = [y(\boldsymbol{\theta}_1), \ldots, y(\boldsymbol{\theta}_N)]^T$ is the corresponding observations. The underlying true function values of $h(\boldsymbol{\theta})$ at these training locations are labeled as vector $\mathbf{h} = [h(\boldsymbol{\theta}_1), \ldots, h(\boldsymbol{\theta}_N)]^T$. The posterior distribution of $\mathbf{h}$ is determined from the likelihood model formed by the observations in $\mathcal{L}$ and a pre-specified prior probability distribution for $\mathbf{h}$.

The prior probability distribution for $\mathbf{h}$ is assumed to be a joint, multivariate Gaussian distribution defined by mean vector $\mathbf{m} = [m(\boldsymbol{\theta}_1), \ldots, m(\boldsymbol{\theta}_N)]^T$ and $N \times N$

covariance matrix $\mathbf{K} = [\kappa(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j)]$ for $i, j = 1, \ldots, N$,

$$\mathbb{P}(\mathbf{h}|\mathcal{D}, \psi) = \mathcal{N}(\mathbf{h}|\mathbf{m}, \mathbf{K}). \tag{2.13}$$

The term ψ refers to the kernel function's hyperparameters. While there are many different possible kernel functions, this thesis always uses the default squared exponential kernel function with automatic relevance determination (SE-ARD) for GPs. The SE-ARD kernel is an RBF kernel much like the kernel in (2.9) for SVMs,

$$\kappa(\boldsymbol{\theta}, \boldsymbol{\theta}') = \sigma_f^2 \mathrm{exp}\{-0.5(\boldsymbol{\theta} - \boldsymbol{\theta}')^T \boldsymbol{\Lambda}^{-1}(\boldsymbol{\theta} - \boldsymbol{\theta}')\} \tag{2.14}$$

$$\boldsymbol{\Lambda} = \mathrm{diag}(\sigma_1^2, \sigma_2^2, \ldots, \sigma_p^2),$$

but with a weighting term $\sigma_1, \ldots, \sigma_p$ for each dimension in $\boldsymbol{\theta} \in \mathbb{R}^p$. The kernel hyperparameters ψ is the set of these terms along with the signal ratio σ_f, $\psi = \{\sigma_f, \sigma_1, \ldots, \sigma_p\}$. In comparison to isotropic squared exponential kernels, SE-ARD enables the hyperparameters associated with each element of $\boldsymbol{\theta} \in \mathbb{R}^p$ to vary independently. Combined with the hyperparameter optimization process in Section 2.2.3, this allows the GP training process to automatically discover elements with low impact on $h(\boldsymbol{\theta})$ and deemphasize them with a low σ_i or emphasize those with high sensitivity with large σ_i.

In addition to the choice of kernel function for $\mathbf{K}$ and its hyperparameters ψ, the prior mean $\mathbf{m}$ has a large impact upon the trained GP regression model. In the vast majority of applications, the prior mean $\mathbf{m}$ is set to $\mathbf{0}$ [85]. This ensures an unbiased prior probability distribution and has been shown to produce good results for countless problems [86], assuming $N \gg p$. The same zero-mean prior is used throughout the thesis, except for specific applications discussed in Chapter 7.

The likelihood model given the observations can be factorized amongst each of the N training points

$$\mathbb{P}(\mathbf{y}|\mathbf{h}, \vartheta) = \prod_{i=1}^{N} \mathbb{P}(y_i|h(\boldsymbol{\theta}_i), \vartheta), \tag{2.15}$$

where ϑ is the set of hyperparameters associated with the likelihood model. The posterior probability distribution for $\mathbf{h}$ is computed from Bayes' rule

$$\mathbb{P}(\mathbf{h}|\mathcal{L},\psi,\vartheta) \propto \mathbb{P}(\mathbf{y}|\mathbf{h},\vartheta)\ \mathbb{P}(\mathbf{h}|\mathcal{D},\psi). \tag{2.16}$$

2.2.2 Predictions

The posterior probability distribution from (2.16) is then used predict the distribution of $h(\boldsymbol{\theta})$ at unobserved points in the input space conditioned on the observed data $\mathcal{L}$. These unobserved query locations, assume N_* in total, are labeled $\mathcal{D}_*$ and $\mathbf{h}_*$ denotes their values of $h(\boldsymbol{\theta})$. According to the GP prior in (2.13), the joint probability distribution of the training $\mathbf{h}$ and the prediction $\mathbf{h}_*$ is a multivariate Gaussian distribution

$$\mathbb{P}(\mathbf{h},\mathbf{h}_*|\mathcal{D},\mathcal{D}_*,\psi) = \mathcal{N}\left(\begin{bmatrix}\mathbf{h}\\ \mathbf{h}_*\end{bmatrix}\middle|\begin{bmatrix}\mathbf{m}\\ \mathbf{m}_*\end{bmatrix},\begin{bmatrix}\mathbf{K} & \mathbf{K}_*\\ \mathbf{K}_*^T & \mathbf{K}_{**}\end{bmatrix}\right). \tag{2.17}$$

Note that the covariance matrix pictured is segmented into three components for easier viewing: the $N \times N$ covariance matrix $\mathbf{K}$ from the training data, $\mathbf{K}_{**}$ the $N_* \times N_*$ covariance matrix of the query locations, and the $N \times N_*$ cross-covariance matrix $\mathbf{K}_*$ between the two sets of $\boldsymbol{\theta}$ locations. From this, the conditional distribution of $\mathbf{h}_*$ given $\mathbf{h}$ is also a multivariate Gaussian,

$$\mathbb{P}(\mathbf{h}_*|\mathbf{h},\mathcal{D},\mathcal{D}_*,\psi) = \mathcal{N}\big(\mathbf{m}_* + \mathbf{K}_*^T\mathbf{K}^{-1}(\mathbf{h}-\mathbf{m}),\mathbf{K}_{**} - \mathbf{K}_*^T\mathbf{K}^{-1}\mathbf{K}_*\big). \tag{2.18}$$

Ultimately, the desired posterior predictive distribution of $\mathbf{h}_*$ is the conditional distribution (2.18) marginalized over the posterior distribution (2.16),

$$\mathbb{P}(\mathbf{h}_*|\mathcal{L},\mathcal{D}_*,\psi,\vartheta) = \int \mathbb{P}(\mathbf{h}_*|\mathbf{h},\mathcal{D},\mathcal{D}_*,\psi)\ \mathbb{P}(\mathbf{h}|\mathcal{L},\psi,\vartheta)\ d\mathbf{h}\ . \tag{2.19}$$

This posterior predictive probability distribution will change based upon the likelihood model of the observations. The two cases to consider are whether the observations of $h(\boldsymbol{\theta})$ are noisy or noise-free.

Noise-free Observations

The first case assumes the measurements of $h(\boldsymbol{\theta})$ are noise-free, meaning $\mathbf{y} = \mathbf{h}$. Since $\mathbf{y}$ measures $\mathbf{h}$ directly, there is no posterior uncertainty about $\mathbf{h}$ after the observations. The posterior predictive distribution of $\mathbf{h}_*$ reduces to

$$\mathbb{P}(\mathbf{h}_* | \mathcal{L}, \mathcal{D}_*, \psi, \vartheta) = \mathcal{N}\big(\mathbf{m}_* + \mathbf{K}_*^T \mathbf{K}^{-1}(\mathbf{h} - \mathbf{m}), \mathbf{K}_{**} - \mathbf{K}_*^T \mathbf{K}^{-1} \mathbf{K}_*\big), \tag{2.20}$$

which is simply the conditional distribution (2.18).

Noisy Observations

In the other case, the observations of $h(\boldsymbol{\theta})$ are assumed to be corrupted by some noise term that prevents $\mathbf{y}$ from measuring $\mathbf{h}$ directly. Many possible types of noise exist, the simplest of which is the standard uniform Gaussian noise assumption made by the vast majority of GP literature [85–92]. The following paragraphs will discuss GP predictions using observations corrupted by uniform Gaussian noise. Later chapters will discuss variations on this noise model as they apply to problems of interest.

In the standard problem, the observations $\mathbf{y}$ are corrupted by a noise term ϵ, which is assumed to follow a uniform Gaussian distribution $\epsilon \sim \mathcal{N}(0, \epsilon_n^2)$ such that $y(\boldsymbol{\theta}) = h(\boldsymbol{\theta}) + \epsilon$ and

$$\mathbb{P}(\mathbf{y} | \mathbf{h}, \epsilon_n) = \mathcal{N}(\mathbf{y} | \mathbf{h}, \epsilon_n^2). \tag{2.21}$$

Due to the fact the noise variance ϵ_n is $\boldsymbol{\theta}$-invariant and constant across the space, uniform Gaussian noise is also referred to as *homoscedastic* Gaussian noise. When noise variance ϵ_n is known in advance, likelihood model hyperparameters $\vartheta = \epsilon_n$ and the likelihood model for the observations can be written as

$$\mathbb{P}(\mathbf{y} | \mathbf{h}, \vartheta) = \mathcal{N}(\mathbf{y} | \mathbf{h}, \epsilon_n^2 \mathbf{I}). \tag{2.22}$$

The resulting posterior predictive distribution is now

$$\mathbb{P}(\mathbf{h}_*|\mathcal{L},\mathcal{D}_*,\psi,\vartheta) = \mathcal{N}\Big(\mathbf{m}_* + \mathbf{K}_*^T\left(\mathbf{K}+\epsilon_n^2\mathbf{I}\right)^{-1}(\mathbf{y}-\mathbf{m}), \mathbf{K}_{**} - \mathbf{K}_*^T\left(\mathbf{K}+\epsilon_n^2\mathbf{I}\right)^{-1}\mathbf{K}_*\Big), \quad (2.23)$$

where the predictions are based upon the evidence provided by the noisy observations $\mathbf{y}$. When ϵ_n is not known in advance, the noise variance can be approximated from the training data and the hyperparameter ϑ is set to that estimate.

2.2.3 Hyperparameter Optimization

The choice of hyperparameters ψ,ϑ greatly affects the GP regression model and its predictions. These hyperparameters define the kernel width, scaling, etc., which ultimately determine the posterior predictive distribution $\mathbb{P}(\mathbf{h}_*|\mathcal{L},\mathcal{D}_*,\psi,\vartheta)$. Surprisingly, in many recent GP-based active learning and Bayesian optimization approaches [93–96], there is little-to-no discussion about the selection of hyperparameters. Often, convergence guarantees are contingent upon the assumption that the ideal, or "true", hyperparameters that maximize the accuracy of the predictions are known in advance.

In this thesis, little about the system's performance is assumed to be known a priori, meaning the optimal hyperparameters will generally not be known in advance. Instead, the hyperparameters can only be estimated with the current available information, training dataset $\mathcal{L}$. From a Bayesian perspective, the uncertainty in the hyperparameters is modeled as the following probability distribution

$$\mathbb{P}(\psi,\vartheta|\mathcal{L}) = \frac{\mathbb{P}(\mathbf{y}|\mathcal{D},\psi,\vartheta)\mathbb{P}(\psi,\vartheta)}{\int \mathbb{P}(\mathbf{y}|\mathcal{D},\psi,\vartheta)\mathbb{P}(\psi,\vartheta)\; d\psi\; d\vartheta}\;, \quad (2.24)$$

where $\mathbb{P}(\mathbf{y}|\mathcal{D},\psi,\vartheta)$ is the marginal likelihood of $\mathbf{y}$ and $\mathbb{P}(\psi,\vartheta)$ is the joint prior distribution on the hyperparameters. Without prior knowledge about the choice of ψ,ϑ, prior $\mathbb{P}(\psi,\vartheta)$ should be set to an uniform distribution. In order to encapsulate the effects of uncertain hyperparameters, the posterior predictive distribution should be marginalized over $\mathbb{P}(\psi, \vartheta|\mathcal{L})$,

$$\mathbb{P}(\mathbf{h}_*|\mathcal{L}, \mathcal{D}_*) = \int \mathbb{P}(\mathbf{h}_*|\mathcal{L}, \mathcal{D}_*, \psi, \vartheta)\ \mathbb{P}(\psi, \vartheta|\mathcal{L})\ d\psi\ d\vartheta. \tag{2.25}$$

In practice, the analytical integral in (2.25) is computationally intractable as $\mathbb{P}(\psi, \vartheta|\mathcal{L})$ will likely be more complex than a Gaussian distribution [85]. Sampling-based approximations of (2.25) using a large number (M_{total}) of evaluations at different ψ, ϑ settings can be obtained, i.e.

$$\mathbb{P}(\mathbf{h}_*|\mathcal{L}, \mathcal{D}_*) \approx \sum_{k=1}^{M_{total}} \mathbb{P}(\mathbf{h}_*|\mathcal{L}, \mathcal{D}_*, \psi_k, \vartheta_k)\ \mathbb{P}(\psi_k, \vartheta_k|\mathcal{L}), \tag{2.26}$$

but such numerical approximations are infeasible as they require repeated inversions of the $N \times N$ matrix $\mathbf{K}$, an $\mathcal{O}(N^3)$ operation. Markov Chain Monte Carlo (MCMC) methods can also be used to compute (2.25) [85, 86], but suffer from the same limitation.

Rather than compute the full distribution over the hyperparameters and the subsequent marginalized posterior predictive distribution for $\mathbf{h}_*$, maximum likelihood estimation [85] will produce a computationally-efficient approximation of (2.25). Assuming the prior $\mathbb{P}(\psi, \vartheta)$ is uniform, the distribution $\mathbb{P}(\psi, \vartheta|\mathcal{L})$ is directly proportional to the marginal likelihood $\mathbb{P}(\mathbf{y}|\mathcal{D}, \psi, \vartheta)$ according to (2.24). Therefore, the locally-optimum hyperparameters ψ^*, ϑ^* are obtained by maximizing the model evidence $\mathbb{P}(\mathbf{y}|\mathcal{D}, \psi, \vartheta)$,

$$\psi^*, \vartheta^* = \underset{\psi, \vartheta}{\operatorname{argmax}}\ \mathbb{P}(\mathbf{y}|\mathcal{D}, \psi, \vartheta). \tag{2.27}$$

This likelihood maximization is known as *hyperparameter optimization* and can be computed using steepest ascent [85] or gradient-based [89] methods. The integral in (2.25) is then replaced with a point estimate at these maximum likelihood values,

$$\mathbb{P}(\mathbf{h}_*|\mathcal{L}, \mathcal{D}_*) \approx \mathbb{P}(\mathbf{h}_*|\mathcal{L}, \mathcal{D}_*, \psi^*, \vartheta^*). \tag{2.28}$$

2.3 Temporal Logic

In many of the verification problems considered in this thesis, the performance requirements for the system under test are trajectory-based specifications. The requirements could include traditional objectives such as stability, boundedness, rise-time, and settling time, or more complex spatial-temporal requirements such as those found in civilian [16] and military [15] aviation standards. All of the aforementioned performance requirements can be expressed in temporal logic [97], which simply provides a mathematical framework for formally defining these specifications. Various derivations of temporal logic exist, with the most common and relevant to control applications being linear temporal logic (LTL) [98–101]. This thesis will consider two particular extensions of LTL, namely metric temporal logic (MTL) [58, 63, 102, 103] and signal temporal logic (STL) [102, 104–106], which were developed to extend LTL into real-time [107]. As will be elaborated later in this section, MTL returns a binary evaluation of a requirement's satisfaction while STL returns a quantitative, real-valued measurement that indicates both the satisfaction of the requirement and the corresponding level of robustness to that requirement. Note that MTL is sometimes referred to as metric interval temporal logic (MITL) when handling interval-based formula. For clarity and simplicity, all binary temporal logic problems will be referred to as MTL.

Metric Temporal Logic

For both MTL and STL problems, a requirement is given by a temporal logic formula φ. In binary MTL problems, this formula consists of boolean atomic propositions p as well as boolean and temporal operations on those propositions. The most common temporal operators include $\Box$ and $\Diamond$, which express that a proposition/formula must "always" hold or "eventually" be true. These operators can also be functions of time intervals such as $\Box_{[t_1,t_2]}$ and $\Diamond_{[t_1,t_2]}$, which state that it must "always hold between times t_1 and t_2" or "eventually be true at some point within time interval $[t_1, t_2]$." Boolean operators include $\neg$, $\wedge$, and $\vee$ to express negation, conjunction, and disjunction. One of the more important points about temporal logic is that these boolean operators can be used to construct more complex formula from simpler ones. For example, formula $\varphi_3 = \Box_{[t_1,t_2]}\varphi_1 \wedge \Diamond_{[t_2,t_3]}\varphi_2$ states that φ_1 must hold for all times between t_1 and t_2 *and* φ_2 must occur at some point between t_2 and t_3 in order for the formula φ_3 to be satisfied.

Satisfaction of the formula is denoted by operator $\models$, where tuple $(\Phi, t) \models \varphi$ indicates trajectory Φ satisfied φ at time t. Failure to satisfy φ is denoted by $(\Phi, t) \models \neg\varphi$ or $(\Phi, t) \not\models \varphi$. For MTL problems, the boolean satisfaction of φ is signified by binary indicator function $\chi(t)$, where

$$\chi(t) = \begin{cases} +1, & \text{if } (\Phi, t) \models \varphi \\ -1, & \text{otherwise.} \end{cases} \tag{2.29}$$

The corresponding binary measurement y for verification over the entire trajectory is the minimum indicator function value, $y = \min\,\{\chi(t)\}\ \forall t \leq T_f$, where T_f is the final time of the trajectory.

Signal Temporal Logic

While MTL is useful for determining satisfaction of φ, it does not differentiate between trajectories that barely satisfy the requirement and those that have a comfortable level of robustness. Signal temporal logic addresses this limitation with the inclusion of a real-valued robustness signal $\rho^\varphi \in \mathbb{R}$ that not only signifies the boolean satisfaction of φ, but also quantifies the minimum level of robustness.

The overall syntax for STL formula φ is almost identical to MTL [106], with the exception that boolean propositions p are replaced by predicates ζ. STL predicates are boolean operators on some real-valued function of states, control inputs, and/or sensor measurements. These predicates are also used to define the robustness degree ρ^φ. For example, if the requirement specifies state $x_1(t)$ must remain above 2, then the corresponding predicate is given by $\zeta(t) = x_1(t) - 2 > 0$. With this predicate, the robustness signal is simply $\rho^\varphi(t) = x_1(t) - 2$. A short summary of robustness signals for more complex STL formula is listed below:

$$\begin{aligned}
\rho^{\varphi_1 \wedge \varphi_2}(t) &= \min\ \big(\rho^{\varphi_1}(t), \rho^{\varphi_2}(t)\big) \\
\rho^{\Box_{[t_1,t_2]}\varphi}(t) &= \min_{t' \in [t+t_1, t+t_2]} \rho^{\varphi}(t') \\
\rho^{\Diamond_{[t_1,t_2]}\varphi}(t) &= \max_{t' \in [t+t_1, t+t_2]} \rho^{\varphi}(t').
\end{aligned} \tag{2.30}$$

Note that the robustness signal $\rho^{\varphi}(t)$ is consistent with the binary indicator function $\chi(t)$

$$\chi(t) = \mathrm{sign}\big(\rho^{\varphi}(t)\big), \tag{2.31}$$

where here it is assumed $\rho^{\varphi}(t) = 0$ signifies $(\Phi, t) \nvDash \varphi$ and $\chi(t) = -1$. Similar to the MTL case, the robustness/satisfaction measurement for the entire trajectory is the minimum robustness degree, $y = \min\{\rho^{\varphi}(t)\}\ \forall t \leq T_f$, where T_f is the final time in the trajectory.

Chapter 3

Deterministic Verification with Binary Evaluations of Performance

This chapter considers the problem of binary verification of deterministic nonlinear systems. Binary verification assumes the performance of a closed-loop system is given by binary ("satisfactory"/"unsatisfactory") evaluations, which naturally segments the set of all possible operating conditions into two distinct sets: those that lead to satisfactory performance and those that do not. This chapter presents statistical data-driven verification procedures to estimate these two sets given a limited amount of simulation or experimental trajectories. The second contribution is a closed-loop verification approach which minimizes the prediction error for a fixed budget of trajectories.

3.1 Problem Description

Consider the deterministic nonlinear system

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t), \boldsymbol{\theta}) \tag{3.1}$$

subject to uncertain operating conditions $\boldsymbol{\theta} \in \mathbb{R}^p$, where $\mathbf{x}(t) \in \mathbb{R}^n$ is the state vector and $\mathbf{u}(t) \in \mathbb{R}^m$ is the control input. The open-loop dynamics in (3.1) are said to be deterministic, meaning two instantiations of (3.1) with identical $\{\mathbf{x}, \mathbf{u}, \boldsymbol{\theta}\}$ will produce the same $\dot{\mathbf{x}}$ every time. Additionally, the goal of this work is to verify the performance of closed-loop systems; therefore, the control inputs $\mathbf{u}(t)$ are assumed to be generated by deterministic control policy

$$\mathbf{u}(t) = g(\mathbf{x}(t)). \tag{3.2}$$

The resulting closed-loop deterministic system with controller (3.2) implemented is written as

$$\dot{\mathbf{x}}(t) = f_{cl}(\mathbf{x}(t), \boldsymbol{\theta}) \tag{3.3}$$

to emphasize control inputs $\mathbf{u}(t)$ can effectively be treated as hidden states within the closed-loop system dynamics.

Both the open- and closed-loop systems are functions of parametric uncertainties $\boldsymbol{\theta}$. The parametric uncertainties are treated as uncertain, time-invariant conditions that affect the state dynamics. These uncertainties $\boldsymbol{\theta}$ are assumed to fall within a known, bounded set Θ as the "known unknowns."

Assumption 3.1. *The set of all possible perturbations $\boldsymbol{\theta} \in \Theta$ is a known, compact, uncountable set $\Theta \subset \mathbb{R}^p$.*

Although the assumption limits $\boldsymbol{\theta}$ to an element of set Θ, the vast majority of physical dynamical systems will have feasible bounds on $\boldsymbol{\theta}$ and this is not a restrictive assumption. For instance, a commercial airliner at cruise will only operate between well-defined weight limits, i.e. empty weight and max take-off weight. Thus, when aircraft weight is a relevant parameter for $\boldsymbol{\theta}$, these weight limits will define bounds on the corresponding element in Θ.

The closed-loop system's trajectory is given by $\Phi(\mathbf{x}(t)|\mathbf{x}_0, \boldsymbol{\theta})$ and defines the time evolution of the states from (3.3) over a time interval $t \in [0, T_{final}]$. The trajectory itself is completely determined by two terms: a nominal initial state vector $\mathbf{x}_0$ that is constant for every instantiation of the system and parametric uncertainties $\boldsymbol{\theta}$. Any uncertainty in the initial state vector $\mathbf{x}(0)$ can be modeled as the combination of nominal $\mathbf{x}_0$ and corresponding elements of $\boldsymbol{\theta}$, ex: $\mathbf{x}(0) = \mathbf{x}_0 + \boldsymbol{\theta}$. Only the non-state terms that change across different instantiations of the system are incorporated as uncertain parameters in $\boldsymbol{\theta}$. The trajectory response can then be mapped directly to its underlying $\boldsymbol{\theta}$ values.

While the parametric uncertainties may describe uncertainty in the initial condition $\mathbf{x}(0)$, they are not restricted to simply varying initial states $\mathbf{x}(0)$. For example, in flight vehicles, $\boldsymbol{\theta}$ may include uncertain system parameters such as aircraft weight, center of gravity location, and moments inertia as well as uncertain initial states for altitude, airspeed, and vehicle orientation. In fact, these uncertain system parameters greatly affect the longitudinal and lateral response of aircraft, but are typically difficult to perfectly calculate before flight. Even within the same production variant of aircraft, small differences in manufacturing and maintenance will result in measurable changes in the aforementioned system parameters between individual flight vehicles. Regardless of the particular source, it is important to incorporate relevant uncertainties in the system parameters whenever appropriate.

Likewise, the assumption of constant parametric uncertainties $\boldsymbol{\theta}$ does not completely eliminate applications with time-varying system parameters. For instance, in conventional fuel-powered aircraft, aircraft weight will decrease as fuel is burned. In problems with short timescales relative to the length of a complete mission, the change in aircraft weight will be negligible and can approximated as a constant. For problems with longer timescales of roughly the same order as a complete flight, this approximation will no longer apply. While the weight will noticeably vary with time, the rate of fuel burn is a well-modeled function of control inputs like throttle and power settings, states, and parameters. Thus, aircraft weight can instead be treated as an internal state defined by an uncertain initial take-off weight and relevant fuel burn parameters. Many other time-varying system parameters can be treated in a similar manner.

Lastly, the parametric uncertainties may also include design parameters under consideration by the controls engineer. For instance, the engineer may examine the effect of controller gains or settings upon the satisfaction of requirements. While controller gains themselves are typically fixed before implementation on the final, "consumer-ready" product, it may be helpful to vary these terms during the verification analysis to explore their interactions with the effects of other uncertainties and their cumulative impact on requirement satisfaction. Similarly, design parameters may include other relevant features such as time delays. In that case, verification would help identify the maximum allowable time delay before the system no longer satisfies the requirement(s) and produce something similar to a time-delay margin for a nonlinear system. Ultimately, these design variables simply act as another component of $\boldsymbol{\theta}$ for the statistical verification procedure to consider during the robustness analysis.

3.1.1 Discrete Evaluations of Performance Requirement Satisfaction

In verification applications, the system's trajectory $\Phi(\mathbf{x}(t)|\mathbf{x}_0, \boldsymbol{\theta})$ is evaluated against pre-specified performance requirements. These performance requirements may include straightforward considerations such as stability and avoidance of failure states or more complex spatial and/or temporal specifications. Regardless of the level of complexity, a binary oracle determines whether these requirements are satisfied by a particular trajectory.

Assumption 3.2. *There exists an oracle which provides deterministic Boolean evaluations on whether a trajectory $\Phi(\mathbf{x}(t)|\mathbf{x}_0, \boldsymbol{\theta})$ satisfied the performance requirements under consideration. These Boolean evaluations are output as binary measurements $y \in \{+1, -1\}$ corresponding to {"satisfied", "did not satisfy"}.*

Remark 3.3. *As each trajectory is completely defined by constant $\mathbf{x}_0$ and the particular instance of $\boldsymbol{\theta}$, a binary measurement is written as $y(\boldsymbol{\theta}) \in \{+1, -1\}$ to emphasize the performance is an explicit function of $\boldsymbol{\theta}$.*

Assumption 3.2 states discrete evaluations of requirement satisfaction are provided by a general binary oracle. Typically, the requirements under consideration can be written in metric temporal logic (MTL) format discussed in Section 2.3 and the oracle would simply be based upon the binary indicator function χ from (2.29). However, the oracle output and the requirements under test can also be provided by a much larger set of sources than just MTL. For instance, the binary measurements could be output from a proprietary gray-box function or even labels produced by a human supervisor. "Oracle" is used as a catch-all term for the white or gray-box model that produces binary labels indicating whether the system satisfies particular requirements.

Systems with Multiple Requirements

Additionally, it is important to clarify that the binary oracle can be used to indicate satisfaction of a single requirement or an entire set of requirements. The preceding paragraphs and discussions referred to a set of pre-specified requirements under consideration as most real-world systems will have multiple performance requirements that have to be addressed simultaneously. In those problems, a positive measurement $y(\boldsymbol{\theta}) = +1$ indicated that the trajectory satisfied *all* the requirements, while a negative measurement $y(\boldsymbol{\theta}) = -1$ indicated the trajectory did not satisfy *all* the requirements. In the latter case, it is possible a subset of the requirements were indeed satisfied, but at least one requirement was not. If it is desirable to examine each requirement individually, the oracle could be reconfigured to output multiple binary measurements with each measurement corresponding to the satisfaction of a single requirement. However, the upcoming data-driven statistical verification techniques would have to be performed on each requirement individually.

3.1.2 Region of Satisfaction

The ultimate goal of deterministic verification is to identify which parametric uncertainties $\boldsymbol{\theta}$ will result in satisfactory performance and those which will not. The binary aspect of this problem leads to the following two definitions.

Definition 3.4. *The* region of satisfaction Θ_{sat} *contains all* $\boldsymbol{\theta} \in \Theta$ *for which the resulting trajectory satisfies the performance requirements. In other words,*

$$\Theta_{sat} := \Big\{ \boldsymbol{\theta} \in \Theta : y(\boldsymbol{\theta}) = 1 \Big\}. \tag{3.4}$$

Under weak assumption, $\Theta_{sat} \neq \emptyset$.

Definition 3.5. *The* region of failure Θ_{fail} *contains all remaining* $\boldsymbol{\theta} \in \Theta$ *for which the resulting trajectory fails to satisfy the performance requirements, i.e.*

$$\Theta_{fail} := \Big\{ \boldsymbol{\theta} \in \Theta : y(\boldsymbol{\theta}) = -1 \Big\}. \tag{3.5}$$

It is also assumed $\Theta_{fail} \neq \emptyset$. *By construction,* Θ_{fail} *is the complement of* Θ_{sat}, *so* $\Theta_{sat} \cup \Theta_{fail} = \Theta$ *and* $\Theta_{sat} \cap \Theta_{fail} = \emptyset$.

While the conditions for membership in sets Θ_{sat} and Θ_{fail} are known, the sets themselves are unknown in advance; it is not clear whether arbitrary $\boldsymbol{\theta}$ belongs to Θ_{sat} or Θ_{fail}. Therefore, the underlying aim of the verification process is to predict Θ_{sat}.

Problem 3.1. *Given the deterministic closed-loop system (3.3) and deterministic binary measurements of requirement satisfaction, compute an estimated region of satisfaction* $\widehat{\Theta}_{sat}$.

Due to the binary nature of the problem, all remaining elements not in $\widehat{\Theta}_{sat}$ are considered to be elements of estimated $\widehat{\Theta}_{fail}$, $\widehat{\Theta}_{fail} = \Theta \setminus \widehat{\Theta}_{sat}$. This same binary perspective also forms the basis for this chapter's approach.

Proposition 3.2. *The problem described in Prob. 3.1 can be viewed as a binary classification problem: predict whether queried* $\boldsymbol{\theta}$ *is an element of* $\widehat{\Theta}_{sat}$ *or* $\widehat{\Theta}_{fail}$.

3.2 Deductive Verification Methods

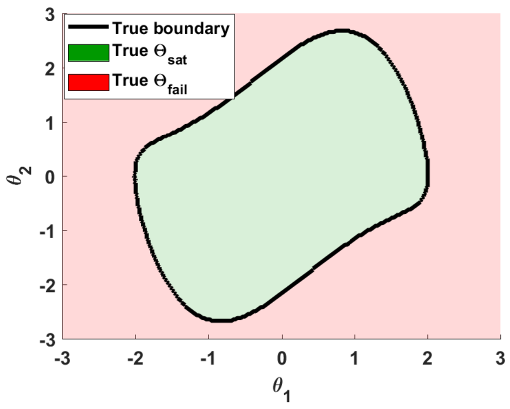
The region of satisfaction estimation objective described in Problem 3.1 can be approached from two primary directions: deductive analysis and statistical methods.

The key distinguishing factor between the two is that deductive methods produce an analytically-verified, proof-driven solution while statistical methods output weaker probabilistic bounds or approximations. While a theoretically-proven analytical result is stronger than a statistical estimate, this section will illustrate the limitations of deductive approaches which are addressed by statistical data-driven verification. In particular, it will highlight simulation-guided deductive methods that bridge the gap between two directions, and these methods will also serve as a foil to statistical methods for direct comparison of the two.

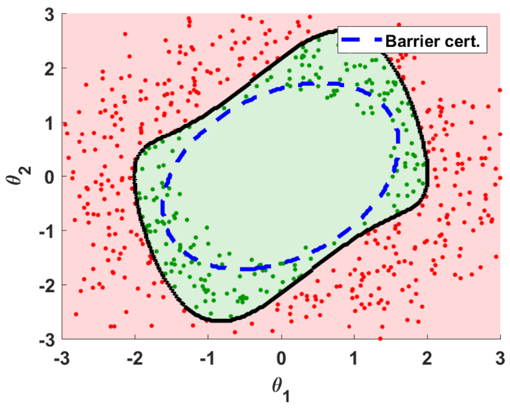
Deductive methods such as LQR-trees [48, 54] and barrier certificates [45, 46, 108] are common techniques for verification of nonlinear systems. These techniques rely upon continuously-differentiable analytical functions $V(\mathbf{x}, \boldsymbol{\theta}) : \mathbb{R}^{n+p} \to \mathbb{R}$ like Lyapunov, barrier, or storage functions to construct proofs that are used to provably guarantee certain $\boldsymbol{\theta}$ vectors will belong to Θ_{sat} under the given modeling assumptions. With respect to Problem 3.1, these elements proven to belong to Θ_{sat} form set $\widehat{\Theta}_{sat}$.

One of the primary difficulties associated with analytical function-based verification is the choice of an appropriate analytical function used to construct the proof. An appropriate Lyapunov or Lyapunov-like function must be fully known in advance for these analytical verification methods to be even applied to the problem. Simulation-guided barrier certificates [47,50–52] and LQR-trees [54] remove some of this difficulty by automatically constructing proofs with the help of simulation (or experimental) trajectories. Typically, simulations of the systems are used to discover invariant sets that bound convergent or stable trajectories according to some polynomial or logarithmic Lyapunov function of pre-specified degree. Simulations are chosen according to a data generation procedure in an attempt to discover a maximizing invariant set, and thus the largest $\widehat{\Theta}_{sat}$ [51, 109].

This process is illustrated on the well-studied [51, 53] unstable Van der Pol oscillator problem shown in Figure 3-1. In this example, the simulation-guided barrier certificate technique generates samples to maximize the size of the verified region produced by a barrier certificate with a second-order Lyapunov function. The bar-



(a) True region of satisfaction



(b) Simulation-guided barrier certificate [53]

Figure 3-1: Illustration of simulation-guided barrier certificate construction techniques [53] on an unstable Van der Pol oscillator case study. Initial conditions $\boldsymbol{\theta} = [\theta_1, \theta_2]$ for the simulations are shown as dots and their respective colors denote whether the resulting trajectory satisfied the requirement (green) or did not (red). The invariant level set described by a second-order Lyapunov function is shown as a dashed blue line. Figure 3-2 will further discuss the conservativeness of this set.

rier certificate is able to produce an estimated region of satisfaction $\widehat{\Theta}_{sat}$, but this prediction falls short of the true Θ_{sat}. This result highlights some of the limitations encountered with analytical methods.

3.2.1 Limitations

There are a number of aspects of deductive methods that limit their applicability to the verification objective in this chapter. First, deductive methods are essentially conservative by design because they seek to address a different objective. Elements that are theoretically guaranteed by the proof to meet the requirements are obviously labeled as members of set $\widehat{\Theta}_{sat}$; however, the problem lies with elements of Θ that are not verified by the proof-based certificate. As pictured in Figure 3-1(b), the quadratic barrier certificate for the Van der Pol oscillator is not able to verify all elements of Θ_{sat}, even points where simulation trajectories have shown this to be true. As nothing else can be inferred, deductive methods must classify all locations not proven safe by the barrier certificate as elements of set $\widehat{\Theta}_{fail}$, illustrated in Figure 3-2. This inherent conservativeness ensures analytical deductive methods will never misclassify

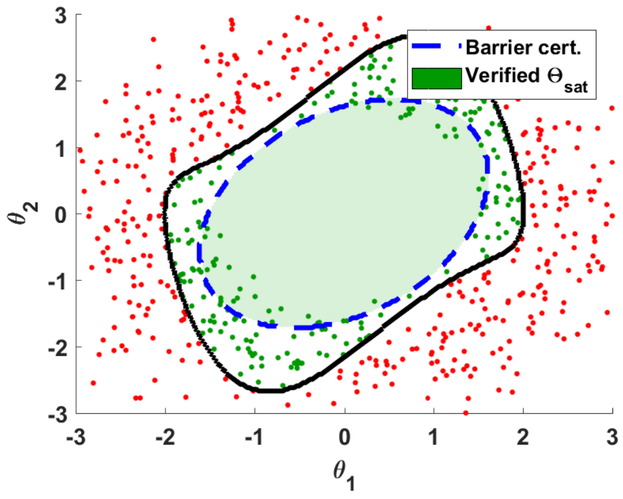


Figure 3-2: Illustration of the predicted region of satisfaction $\widehat{\Theta}_{sat}$ produced by a second-order Lyapunov function-based barrier certificate for the unstable Van der Pol oscillator previously shown in Figure 3-1. The method is conservative by design and only points verified by the barrier certificate are included in $\widehat{\Theta}_{sat}$, resulting in the noticeably conservative predictions shown as the green ellipse.

elements of Θ_{fail} within $\widehat{\Theta}_{sat}$, but this is not true for elements of Θ_{sat} within $\widehat{\Theta}_{fail}$. These barrier certificate methods addressed a slightly different objective and were not concerned with the possibility of these false negatives where $\boldsymbol{\theta} \in \Theta_{sat}$ are misclassified as elements of $\widehat{\Theta}_{fail}$.

A second limitation of analytical methods, particularly barrier certificate techniques, is that they require the availability of continuously-differentiable Lyapunov or Lyapunov-like functions. These Lyapunov functions must be specified in advance, but often the correct choice or form of the function is not known and can be difficult to determine, if one even exists. While simulation-guided methods relax this by automatically discovering function coefficients and bounding terms, they still require an approximate degree for the polynomial Lyapunov function to be set by the user [51]. If an incorrect order is chosen, there may not exist an appropriate barrier certificate for that particular set of functions, even if one exists for another form. There may even be multiple possible polynomial orders that can be used to construct barrier certificates, but the choice that maximizes the accuracy of the resulting $\widehat{\Theta}_{sat}$ is unknown.

Lastly, the computational complexity of deductive methods limits their utility. At a minimum, the corresponding analytical function $V(\mathbf{x}, \boldsymbol{\theta})$ must be a function of both the closed-loop states $\mathbf{x}$ and the parametric uncertainties $\boldsymbol{\theta}$ in order to produce a barrier certificate. This can pose a problem for high-dimensional systems, even if the dimension p of uncertainties $\boldsymbol{\theta} \in \mathbb{R}^p$ is low, due to the necessary inclusion of $\mathbf{x}$ in order to calculate suitable bounds on the trajectory response. This problem is exacerbated as the parameterized form of the function becomes more complex. In general, the function forms that maximize the accuracy of $\widehat{\Theta}_{sat}$ are higher-order polynomials or similar representations and the more complex the closed-loop dynamics, the higher the order or complexity of the function required to achieve even a suboptimal $\widehat{\Theta}_{sat}$. Therefore, it may become extremely difficult to find *any* suitable Lyapunov or Lyapunov-like function to $\widehat{\Theta}_{sat}$.

3.3 Statistical Data-Driven Verification

Instead of the previous deductive techniques, Problem 3.1 can be approached through statistical methods. Rather than utilize simulations (or experiments) to construct and refine a proof that eventually produces a verified $\widehat{\Theta}_{sat}$, statistical methods operate more directly with the data. The resulting predictions for $\widehat{\Theta}_{sat}$ and their accuracy are directly coupled to the quality of the observed data, hence the term *data-driven verification.*

3.3.1 SVM Classification Models

In order to translate a finite number of trajectories into nontrivial predictions over the entire set Θ, these trajectories are used to form a classification model. With the assumption of binary measurements $y(\boldsymbol{\theta})$, there are a number of applicable machine learning/data-mining techniques such as relevance vector machines [110], kernel logistic regression [81], decision trees [111], and random forests [81] that can be used to produce classifiers. Ultimately, this approach uses nonlinear, soft-margin support vector machines (SVMs) [77,79] for binary classification as they can efficiently handle arbitrary, nonlinearly-separable datasets without modification, an important consideration when nothing is assumed about the shape, convexity, or separability of Θ_{sat}.

The overall process of constructing and training the SVM-based classification model follows the details discussed in Section 2.1.2. The support vector machine is constructed from a training dataset $\mathcal{L}$ consisting of N trajectories initialized at locations $\mathcal{D} = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_N\}$ in Θ and the $N \times 1$ vector of their corresponding binary evaluations $\mathbf{y} = [y(\boldsymbol{\theta}_1), y(\boldsymbol{\theta}_2), \ldots, y(\boldsymbol{\theta}_N)]^T$, i.e. $\mathcal{L} = \{\mathcal{D}, \mathbf{y}\}$. Isotropic Gaussian radial basis functions (2.10) are used as kernels $\kappa(\boldsymbol{\theta}, \boldsymbol{\theta}')$ in order to project from $\mathbb{R}^p$ space into a higher-dimensional space where the dataset is linearly separable. The resulting SVM output is given in the same format as (2.11) with

$$H(\boldsymbol{\theta}) = \text{sign}\, \Big(\sum_{j=1}^{N_{sv}} \alpha_j \; y_j \; \kappa(\boldsymbol{\theta}_j, \boldsymbol{\theta}) + b \Big), \tag{3.6}$$

where $j = 1, \ldots, N_{sv}$ are the active support vectors chosen from $\mathcal{L}$ ($N_{sv} \leq N$). All elements of $\mathcal{L}$ not selected as support vectors can be viewed as having $\alpha_j = 0$ and ignored for computational efficiency.

The support vector machine classifier's output from (3.6) is used to construct the predicted regions of satisfaction and failure. The binary output is used directly as the predicted satisfaction label, $\widehat{y}(\boldsymbol{\theta}) = H(\boldsymbol{\theta})$. The estimated regions parallel Definitions 3.4 and 3.5, but with predicted label $\widehat{y}(\boldsymbol{\theta})$ in place of true $y(\boldsymbol{\theta})$.

Definition 3.6. *The* predicted region of satisfaction $\widehat{\Theta}_{sat}$ *contains all* $\boldsymbol{\theta} \in \Theta$ *for which the resulting trajectory is predicted to satisfy the performance requirements,*

$$\widehat{\Theta}_{sat} := \Big\{ \boldsymbol{\theta} \in \Theta : \widehat{y}(\boldsymbol{\theta}) = 1 \Big\}. \tag{3.7}$$

Definition 3.7. *The* predicted region of failure $\widehat{\Theta}_{fail}$ *is the complement of* $\widehat{\Theta}_{sat}$ *and contains all remaining* $\boldsymbol{\theta} \in \Theta$ *for which the resulting trajectory is predicted to fail to satisfy the performance requirements,*

$$\widehat{\Theta}_{fail} := \Big\{ \boldsymbol{\theta} \in \Theta : \widehat{y}(\boldsymbol{\theta}) = -1 \Big\}. \tag{3.8}$$

The SVM classifier contains two hyperparameters that can be tuned to adjust the prediction regions. First, the γ hyperparameter used by all isotropic RBF kernels (2.10) controls the width of the kernel function. As this γ value increases, the relevance of distant training datapoints decreases. For this work, the kernel hyperparameter is set to $\gamma = 1$ and the training positions $\mathcal{D}$ are normalized to interval $[-1, 1]$ in each of the p dimensions.

The second hyperparameter is the box constraint C used by the soft-margin SVM primal objective (2.6)/dual constraint (2.7) functions to penalize misclassifications. In the simplest implementation, a scalar $C > 0$ can be increased to more heavily penalize all misclassifications ($\widehat{y}(\boldsymbol{\theta}) \neq y(\boldsymbol{\theta})$) in the training data. As C grows towards ∞, the soft-margin SVM classifier converges towards a hard-margin classifier which assumes linear separability. However, in most applications, misclassification errors are not weighted equally: *false negatives*, where $\widehat{y}(\boldsymbol{\theta}_i) = -1$ but in reality $y(\boldsymbol{\theta}_i) = 1$, are more acceptable than *false positives*, where $\widehat{y}(\boldsymbol{\theta}_i) = 1$ but $y(\boldsymbol{\theta}_i) = -1$. This is because a false negative mistakenly classifiers a truly safe point as unsafe, but mistakenly classifying unsafe operating conditions as safe may have disastrous consequences. In such situations, the scalar box constraint can be replaced by a 2×2 matrix

$$C = \begin{bmatrix} C_{FN} & 0 \\ 0 & C_{FP} \end{bmatrix} \tag{3.9}$$

with $C_{FN} > 0$ and $C_{FP} > 0$. An increase in the ratio C_{FP}/C_{FN} will penalize false positives more heavily than false negatives to discourage the classifier from accepting such errors during the training process. It is important to note that C does not explicitly control the rate of misclassification errors; it only changes the penalties during the training process, which implicitly controls misclassification rate.

Model Validation

While the training process optimizes the classifier to suppress misclassification errors, it can not completely eliminate or control the possibility of incorrect predictions. At the most basic level, this is because a finite number of training points in $\mathcal{L}$ is used to predict response over the full uncountable set Θ. There will always be unsampled/unexplored points and regions in Θ and thus the possibility of errors will always exist. The real problem is that the binary classifier described in (3.6) does not explicitly indicate confidence in the predicted outputs $\widehat{y}(\boldsymbol{\theta})$. The following validation methods will estimate or qualify the confidence in each predicted $\widehat{y}(\boldsymbol{\theta})$ as well as estimate the cumulative rate of misclassifications over the total Θ space.

First, methods can be applied to the SVM model output to estimate the local confidence for each individual prediction $\widehat{y}(\boldsymbol{\theta})$. The simplest evaluation of prediction confidence is the non-binary model output before the sign function is applied in (3.6),

$$H_{\mathbb{R}}(\boldsymbol{\theta}) = \sum_{j=1}^{N_{sv}} \alpha_j \; y_j \; \kappa(\boldsymbol{\theta}_j, \boldsymbol{\theta}) + b. \tag{3.10}$$

This real-valued output $H_{\mathbb{R}}(\boldsymbol{\theta}) \in \mathbb{R}$ signifies the query point's distance from the prediction boundary, where $|H_{\mathbb{R}}(\boldsymbol{\theta})| \gg 0$ is further from the boundary and thus further away from points with the opposite satisfaction label. Additionally, Platt scaling [112] can also be applied to the output $H_{\mathbb{R}}(\boldsymbol{\theta})$, which uses a logistic regression model to convert $H_{\mathbb{R}}(\boldsymbol{\theta})$ into a direct measure of confidence probability.

While these methods are tools for computing some measure of confidence, they do not capture all possible errors. Mainly, the non-binary output $H_{\mathbb{R}}(\boldsymbol{\theta})$ only indicates distance from the prediction boundary, but does not indicate proximity to training points. A point $\boldsymbol{\theta}_i$ may be predicted to lie far away from the boundary, but this same point may lie far away from all the other training datapoints as well and therefore the classifier has no justification as to what the prediction should be at $\boldsymbol{\theta}_i$. $H_{\mathbb{R}}(\boldsymbol{\theta})$ does not indicate the presence of dangerous "holes" in the training data where there does not exist any training data to actually support the predictions made the SVM model. This also applies to Platt scaling since it requires this output to pass through the logistic regression model. Additionally, Platt scaling assumes linear separability, which limits the types of problems it can accurately be applied to. As a result, these two approaches provide some useful qualitative measures of local prediction confidence, but cannot provide complete quantitative estimates.

Other tools can be used to estimate the total accuracy of the SVM model predictions over the full set Θ. These approaches generally rely upon the existence of an independent validation set $\mathcal{V}$ complete with training locations and corresponding binary measurements, just like in $\mathcal{L}$. The most straightforward approach is to simply select more $\boldsymbol{\theta}$ conditions, usually through Monte Carlo sampling, and perform simulations at those conditions in order to collect new measurements and form a validation set $\mathcal{V}$. Once a classifier has been trained on the initial training dataset $\mathcal{L}$, the model is used to predict labels at all locations in $\mathcal{V}$ and these predictions are compared against the true answers, which are known for the validation set. Leave-one-out and k-fold cross-validation approaches [82,113] exploit this same idea, but instead randomly segment $\mathcal{L}$ into a set of data used to actually train the SVM model and a smaller subset that acts as $\mathcal{V}$. The cumulative prediction error on the validation set $\mathcal{V}$ is treated as the estimated prediction error for Θ. K-fold cross validation [113] performs a more complicated version of the leave-one-out approach. The process segments $\mathcal{L}$ into k equal-sized subsets. One of these subsets is used as $\mathcal{V}$ while the other $k-1$ sets collectively train the SVM. The procedure repeats k times so that each of the k sets is used as the validation set once. The validation error from each of the k validation sets is averaged together to produce a single estimate of misclassification error.

Just like before, these validation tools have their limitations. Regardless of its source, if the independent validation fails to adequately cover a region in Θ, then it will fail to accurately predict the misclassification rate in that area. Larger numbers of validation points are required to ensure better probabilistic coverage of Θ, which means additional simulations or experiments. More generally, the reliance upon an independent set of observations is a major limitation in itself. All the observed data allocated to the validation set could have been incorporated as additional data for $\mathcal{L}$, which would generally produce a more accurate classifier. Therefore, its a challenging conundrum to decide whether new trajectories should be added to the validation set $\mathcal{V}$ to improve the estimation of misclassification error rate or to the training dataset $\mathcal{L}$ to actually decrease the misclassification error [114]. All these validation methods are useful, but imperfect, tools for estimating the rate of prediction errors.

3.3.2 Comparison to Simulation-Guided Barrier Certificates

Statistical data-driven verification was originally developed as a parallel, complimentary approach to simulation-guided barrier certificate techniques to identify and address conservativeness in those approaches [115]. Statistical verification can exploit the same datasets used to construct the barrier certificates in order to train SVM classifiers not restricted by the barrier certificate's assumptions. The following subsection will illustrate this fact and directly compare the two approaches applied to the same datasets.

As shown in Figure 3-1(b), simulation-guided deductive methods can be used to generate analytical barrier certificates. In that example, the approach produces quadratic Lyapunov-function based predictions of $\widehat{\Theta}_{sat}$ for an unstable Van der Pol oscillator. The sampling procedure [53] distributes $\boldsymbol{\theta}$ points in order to construct the maximum verifiable set, but even with this procedure, the verifiable results are obviously conservative. When statistical data-driven verification is applied to this same exact dataset, the SVM model will produce significantly less conservative results, pictured in Figure 3-3. The SVM model closely approximates the true boundary, with the predicted boundary (blue line) almost indistinguishable from the true boundary (black line). This is accomplished through the use of 31 support vectors (magenta circles) selected from the training dataset of 633 trajectories and binary measurements.

The accuracy of the SVM-based predictions is summarized in Table 3.1. K-fold cross-validation error using 10 folds is computed as 0.153%; however, this value is only based upon the 633 training datapoints. The true mean error is approximated using an empirical grid of 77,284 datapoints that span Θ. When evaluated on this grid, the SVM's prediction error is a close 0.48%, with a total of 203 false positives and 166 false negatives. For comparison, the simulation-guided barrier certificate avoids false positives completely, but it's conservativeness leads to a significantly higher rate of false negatives and a total misclassifcation error percentage of 11.44%. As discussed in Section 3.2, barrier certificates were designed to address a different objective, so the higher rate of false negatives is not unexpected when both false negatives and false

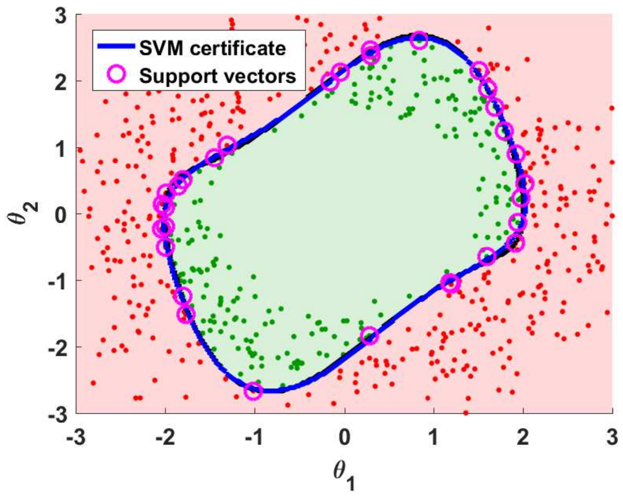


Figure 3-3: Illustration of the predicted region of satisfaction $\widehat{\Theta}_{sat}$ produced by a SVM-based statistical classifier for the unstable Van der Pol oscillator previously shown in Figure 3-1. The predicted boundary separating $\widehat{\Theta}_{sat}$ from $\widehat{\Theta}_{fail}$ is depicted as the solid blue line. Initial conditions $\boldsymbol{\theta} = [\theta_1, \theta_2]$ for the simulations are shown as dots and their respective colors denote whether the resulting trajectory satisfied the requirement (green) or did not (red). Note that this is the same dataset from Figure 3-1. The support vectors chosen from this dataset are surrounded by magenta rings.

positives are now considered. By relaxing the problem from proof-based certificates to statistical certificates, the total misclassification error is significantly reduced as a result of the decreased conservativeness with respect to false negatives at the cost of only a small increase in the rate of false positives.

Table 3.1: Comparison of SVM- and barrier certificate-based predictions generated using the same training dataset. The predictions are both evaluated on a grid of 77,284 points and the empirical mean error, false positive count, and false negative count are displayed.

Prediction Method	K-fold Error	Empirical Error	False Pos.	False Neg.
SVM classifier	0.153%	0.48%	203 of 21962	166 of 55322
Barrier cert.	N/A	11.44%	0 of 21962	8842 of 55322

3.4 Closed-Loop Statistical Verification

The statistical verification process described in the preceding section is able to produce a data-driven classifier for predictions given an arbitrary set of training data. This makes it particularly well-suited to supplement existing simulation-guided barrier certificate methods, but it can be applied to a much wider class of problems with any source of data, either simulation or experiment-based. However, Section 3.3 highlights two important, but conflicting, issues that arise with respect to this training data.

First, the coverage and expressiveness of the training data limits the quality of the data-driven predictions. As mentioned in Section 3.3.1, the predictions are only accurate in regions where the training data has adequately covered the space. The definition of "adequate" might change based upon the application and desired level of validation error, but it fundamentally requires multiple datapoints to be distributed throughout that region of Θ, with concentrations near the decision boundary. Conversely, the second issue that arises in data-driven binary verification is that only a subset of the training data is used as active support vectors for the SVM classifier; the remaining datapoints were deemed uninformative. For instance, the Van der Pol example in Section 3.3.2 ultimately only chose 31 out of 633 trajectory datapoints as support vectors. This is not much of a problem if trajectory data is easy to obtain, but quickly becomes wasteful if the source of the trajectory data is computationally-intensive simulations or experimental tests. Thus, from this viewpoint, it is better to minimize the amount of trajectory data. These two competing considerations both greatly affect the utility of data-driven verification for such resource-constrained problems.

In many applications with nonlinear systems, the latter consideration about the strain on computational resources is increasingly relevant. In simulation environments, the complexity of the simulation model necessary for verification purposes often drives computational demands. These models can range from simple two-state systems to full 6 degree-of-freedom flight simulators with realistic-enough dynamics to replace actual real-world flight hours [18]. At the latter end of the spectrum, it is impractical to run thousands upon thousands of simulation tests. Even with rather simple dynamics, verification may be part of a larger process, such as robust nonlinear control design, which restricts the number of simulations allocated to verification of each prospective controller. When applied to hardware experiments, a feasible limit on resources such as time, money, and testing objects is even more obvious. Regardless of the source and cost of the trajectory data, the computational cost of the SVM training process also increases at a rate of $\mathcal{O}(N^2)$ [116], which becomes non-negligible for large numbers of training points N. For simplicity, all these sources of concern can be imagined as imposing an upper limit on the number of training(+validation) datapoints. The objective laid out earlier in Problem 3.1 would have to be modified to reflect this constraint.

Problem 3.3. *Given the deterministic closed-loop system (3.3) and deterministic binary measurements of requirement satisfaction, compute an estimated region of satisfaction $\widehat{\Theta}_{sat}$ while subject to a limit N_{lim} on the number of allowed training datapoints.*

In an ideal scenario, the optimal training dataset would minimize prediction error while subject to a sample budget by distributing datapoints solely along the $\Theta_{sat}/\Theta_{fail}$ boundary. The prediction error would be minimized/eliminated while the entire dataset would be selected as necessary support vectors. Although this would avoid wasteful, uninformative trajectory data, it is impossible in practice because it requires Θ_{sat} and Θ_{fail} to be known in advance. Instead, this section describes an active learning-based sampling procedure that approximates this ideal dataset. It does so by iteratively selecting the best trajectory conditions to evaluate based upon the current SVM model. This allows the verification procedure to minimize approximate prediction error while subject to constraints on the training data and ultimately converge to a solution that resembles the ideal training dataset.

3.4.1 Sample-Selection Criteria

Active learning [114,116–119] describes a wide variety of machine learning techniques that iteratively select sample locations based upon a given model, obtain new measurements, and retrain that model with the new measurements. Due to the closed-loop train→select→test→retrain nature of active learning procedures, the procedure is called *closed-loop statistical verification* to emphasize the iterative feedback-based improvement of the classification model and training dataset. Central to the closed-loop process is the selection of the best prospective sample locations to most improve the predictions; however, the "best" location will change according to the objective of the active learning algorithm [117]. The objective with respect to Problem 3.3 is to minimize the prediction error of $\widehat{\Theta}_{sat}/\widehat{\Theta}_{fail}$ while limited to N_{lim} total simulation or experimental tests (trajectories).

In order to maximize the improvement in the prediction model for each additional training sample, the expected model change selection metric [118] is used to rank potential sample locations. The metric identifies the sample point that, if measured, is expected to induce the largest change between the current model and the retrained model with the new information. For support vector machines, this point that maximizes the expected model change is found using the objective function of the Lagrangian dual in (2.7). If a new sample is taken at arbitrary location $\boldsymbol{\theta}_{N_{sv}+1}$ with hypothetical measurement $y(\boldsymbol{\theta}_{N_{sv}+1})$, then the modified Lagrangian dual objective is given by

$$D(\boldsymbol{\alpha}) = \sum_{i=1}^{N_{sv}+1} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N_{sv}+1} \alpha_i \; \alpha_j \; y_i \; y_j \; \kappa(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j). \tag{3.11}$$

With the weighting term $\alpha_{N_{sv}+1}$ and bias b set to zero, the gradient of the objective with respect to $\alpha_{N_{sv}+1}$ defines the model change under consideration. This gradient is written as

$$\begin{aligned} \frac{\partial D(\boldsymbol{\alpha})}{\partial \alpha_{N_{sv}+1}} &= 1 - y(\boldsymbol{\theta}_{N_{sv}+1}) \sum_{j=1}^{N_{sv}} \alpha_j \; y_j \; \kappa(\boldsymbol{\theta}_j, \boldsymbol{\theta}_{N_{sv}+1}) && (3.12) \\ &= 1 - y(\boldsymbol{\theta}_{N_{sv}+1}) H_{\mathbb{R}}(\boldsymbol{\theta}_{N_{sv}+1}), && (3.13) \end{aligned}$$

where $H_{\mathbb{R}}(\boldsymbol{\theta}_{N_{sv}+1})$ is the non-binary output produced by the original SVM model before a measurement at $\boldsymbol{\theta}_{N_{sv}+1}$ is obtained. Note that the weighting term $\alpha_{N_{sv}+1}$ is set to zero so the $N_{sv}+1$ index is dropped from the summation function. Since the weighting term $\alpha_{N_{sv}+1}$ is non-negative, the model would only update with this sample if the gradient is positive, meaning $y(\boldsymbol{\theta}_{N_{sv}+1})H_{\mathbb{R}}(\boldsymbol{\theta}_{N_{sv}+1}) < 1$. However, the only way for $y(\boldsymbol{\theta}_{N_{sv}+1})H_{\mathbb{R}}(\boldsymbol{\theta}_{N_{sv}+1}) < 0$ is for the signs of the actual measurement $y(\boldsymbol{\theta}_{N_{sv}+1})$ and

$H_{\mathbb{R}}(\boldsymbol{\theta}_{N_{sv}+1})$ to disagree. If this measurement $y(\boldsymbol{\theta}_{N_{sv}+1})$ was indeed known somehow, then the optimal sample $\overline{\boldsymbol{\theta}}$ would maximize the model change

$$\overline{\boldsymbol{\theta}} = \text{ argmax } \Big(1 - y(\boldsymbol{\theta}) H_{\mathbb{R}}(\boldsymbol{\theta})\Big). \tag{3.14}$$

In reality, the actual measurements would not be known before their selection; therefore, it is only possible to work with the predicted measurement $\widehat{y}(\boldsymbol{\theta})$. In place of the actual model change, the sample location which maximizes *expected* model change is given by

$$\overline{\boldsymbol{\theta}} = \text{ argmax } \Big(1 - \widehat{y}(\boldsymbol{\theta}) H_{\mathbb{R}}(\boldsymbol{\theta})\Big). \tag{3.15}$$

Since the predicted measurements are binary, $\widehat{y}(\boldsymbol{\theta}) \in \{-1, 1\}$, the most informative datapoints are expected to be samples with $|H_{\mathbb{R}}(\boldsymbol{\theta}| \approx 0$,

$$\overline{\boldsymbol{\theta}} = \text{ argmin } |H_{\mathbb{R}}(\boldsymbol{\theta})|. \tag{3.16}$$

3.4.2 Sequential Sampling

The closed-loop statistical verification procedure exploits the selection criterion (3.16) to rank prospective datapoints and identify the one which is expected to most improve the prediction when a simulation or experiment is performed there. In order for this to happen, there must first be an initial classification model, constructed from some initial training dataset of N_0 points, used to start the process. This initial dataset can be generated from any open-loop process, such as traditional design of experiments (DOE) techniques like Latin hypercubes [74] or uniformly-distributed randomized data [73]. Additionally, while the selection criterion evaluates prospective sample locations, these sample locations must come from some set. Although the selection metric can evaluate any arbitrary point, it is impossible to actually analyze every point in the uncountable set Θ. Instead, the following assumption is made to ensure a feasible analysis.

Assumption 3.8. *There exists a sufficiently fine discretization of* Θ*, called* Θ_d*, which is suitable for verification. The closed-loop verification process selects its samples from* Θ_d *rather than* Θ*.*

Remark 3.9. *Because* Θ_d *is assumed to be an extremely fine approximation of* Θ*, it will never be possible to fully cover* Θ_d *with samples, i.e.* $|\Theta_d| \gg N_{lim}$*.*

In order to avoid redundant samples at the same location, the samples are chosen from the set of available sample locations $\mathcal{U}$, where $\mathcal{U} = \Theta_d \setminus \mathcal{D}$ after $\mathcal{D}$ has been updated with the new measurements.

Given this initial SVM model and $\mathcal{U}$, the closed-loop verification process can begin. The most straightforward approach is to iteratively select single measurements in a sequential manner until the sampling budget N_{lim} has been filled. The following paragraph summarizes the closed-loop verification procedure in Algorithm 1.

Step 1 lists the necessary inputs required to start the active sampling process. These inputs include the initial training dataset of N_0 points in $\mathcal{D}$ and their measurements $\mathbf{y}$, the set of available sample locations $\mathcal{U}$, and the maximum number of additional training points $T = N_{lim} - N_0$. The initial training dataset is used to train the initial SVM classification model (Step 2). The procedure then calculates the SVM output $H_{\mathbb{R}}(\boldsymbol{\theta})$ at each available $\boldsymbol{\theta} \in \mathcal{U}$ and selects the location $\overline{\boldsymbol{\theta}}$ according to the selection criterion (3.16) (Step 4). Next, the procedure performs a simulation or experiment at the selected location $\overline{\boldsymbol{\theta}}$ in order to obtain a binary measurement $y(\overline{\boldsymbol{\theta}})$ (Step 5) and incorporates this information into the training dataset $\mathcal{L}$ (Step 6). The procedure then retrains the SVM classification model to exploit the new information provided by the updated training dataset and improve the predictions (Step 7). The process in Steps 4-7 repeats until the limit of T additional simulations/experiments has been reached and the loop terminates. The final prediction model and its output $H_{\mathbb{R}}(\boldsymbol{\theta})$ produces the predicted sets $\widehat{\Theta}_{sat}$ and $\widehat{\Theta}_{fail}$ (Step 9).

Although the closed-loop verification procedure in Algorithm 1 tends to improve the resulting prediction model's accuracy over ones produced with traditional, open-loop DOE techniques, the active selection of training locations produces a higher cost

Algorithm 1 Sequential closed-loop deterministic verification framework using SVM classification models

```
1: Input: initial training set $\mathcal{L} = \{\mathcal{D}, \mathbf{y}\}$, available sample locations $\mathcal{U}$, max # of
   additional samples $T$
2: Initialize: train SVM model $H_{\mathbb{R}}(\boldsymbol{\theta})$
3: for $i = 1, 2, \ldots, T$ do
4:     Select $\overline{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}\in\mathcal{U}}{\operatorname{argmin}} |H_{\mathbb{R}}(\boldsymbol{\theta})|$
5:     Perform test at $\overline{\boldsymbol{\theta}}$, obtain measurement $y(\overline{\boldsymbol{\theta}})$
6:     Add $\{\overline{\boldsymbol{\theta}}, y(\overline{\boldsymbol{\theta}})\}$ to training set $\mathcal{L}$, remove $\overline{\boldsymbol{\theta}}$ from $\mathcal{U}$
7:     Retrain model $H_{\mathbb{R}}(\boldsymbol{\theta})$ with updated $\mathcal{L}$
8: end for
9: Return: predicted sets $\widehat{\Theta}_{sat}$, $\widehat{\Theta}_{fail}$
```

when compared to the random sampling DOE technique [73]. In addition to the $\mathcal{O}(N^2)$ cost of training the SVM prediction model at Step 2 and the $\mathcal{O}(N|\mathcal{U}|)$ cost of the predictions for the final $\widehat{\Theta}_{sat}$ and $\widehat{\Theta}_{fail}$ at Step 9, the for-loop in Steps 3-8 will increase the cumulative computational cost. As a result, the process will require $\mathcal{O}(N|\mathcal{U}|)$ operations for the selection of each point (Step 4) and $\mathcal{O}(N^2)$ operations for retraining the SVM after each measurement (Step 7). After T iterations, the active sampling procedure will ultimately require $\mathcal{O}(N|\mathcal{U}|T)+\mathcal{O}(N^2T)$ more operations than the passive, random sampling procedure. Note, this analysis does not include the cost of actually performing simulations or experiments, which will vary from example to example, but will be independent of the sample-selection method.

3.4.3 Batch Sampling

The sequential verification process described in Algorithm 1 correctly selects each point as intended; however, it neglects two potentially important computational considerations. First, the $\mathcal{O}(N^2)$ computational cost associated with retraining the active learning SVM after each iteration becomes non-negligible if the sampling budget N_{lim} is large [116]. Second, sequential sampling fails to exploit the inherent parallelism present in many, but not all, applications. Particularly in simulation environments, it is common for multiple resources such as processor cores or computers to be allocated for testing purposes. In these cases, samples can be selected in groups of

$M > 1$ points, reducing the retraining cost and allowing multiple simulations (or experiments) to be performed in parallel. Such procedures are referred to as *batch* verification processes.

Although batches of M points will reduce the total cost of retraining, batch sampling also has the potential to degrade prediction performance. The primary cause for this degradation is lack of diversity among samples in the batch. Due to the fine resolution of Θ_d, neighboring points will have similar rankings according to (3.16). If the M highest ranked points are naïvely selected for the batch, then many of the points could be located in close proximity. Redundant samples will generally induce little change in the model after the first measurement and at the very least reduce the number of samples that can be allocated towards other regions of Θ_d. Naïve implementations of batch algorithms will most likely have significantly poorer prediction performance than their corresponding sequential versions given the same total number of samples.

In order to prevent the selection of redundant points, a diversity measure is incorporated into the selection criteria. The angle between induced hyperplanes is a common heuristic diversity metric in generic active learning procedures [116,118] used to rank similarity between potential sample locations. The angle between induced hyperplanes $\phi(\boldsymbol{\theta})$ can be written in terms of kernel functions $\kappa(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j)$,

$$|\cos\left(\angle(\phi(\boldsymbol{\theta}_i), \phi(\boldsymbol{\theta}_j))\right)| = \frac{|\kappa(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j)|}{\sqrt{\kappa(\boldsymbol{\theta}_i, \boldsymbol{\theta}_i)\kappa(\boldsymbol{\theta}_j, \boldsymbol{\theta}_j)}}. \tag{3.17}$$

Maximum diversity between samples would entail maximizing this angle. Rather than select samples to maximize either expected model change *or* diversity, a weighted combination of the two will both ensure the selection of relevant datapoints and also encourage diversity among those locations to prevent redundant samples. The resulting selection metric is simply a convex combination of (3.16) and (3.17)

$$\overline{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \mathcal{U}}{\operatorname{argmin}} \left(\lambda \left|H_{\mathbb{R}}(\boldsymbol{\theta})\right| + (1-\lambda) \max_{\boldsymbol{\theta}_j \in \mathcal{S}} \left|\cos\left(\angle(\phi(\boldsymbol{\theta}), \phi(\boldsymbol{\theta}_j))\right)\right|\right), \tag{3.18}$$

where $\lambda \in [0,1]$ and $\mathcal{S}$ is the set of samples previously selected in the current batch ($|\mathcal{S}| \leq M$). All the example problems heuristically set $\lambda = 0.7$ as that setting produced good misclassification error rates in every example problem considered. Although the training and testing are performed in batches of M points, each of the $j = 1, \ldots, M$ points in the batch are selected sequentially (one-at-a-time) and added to $\mathcal{S}$. The next potential sample location in the current batch $\mathcal{S}$ is evaluated for diversity with respect to the locations already in $\mathcal{S}$ rather than the entire training dataset $\mathcal{D}$. This is because the goal is not to encourage diversity amongst all points, but only within the current batch. Ultimately, it is still desirable for samples to be distributed along the $\Theta_{sat}/\Theta_{fail}$ boundary, not spread out over all of Θ_d.

Algorithm 2 details the batch verification framework. Just as before, the procedure starts with a given initial training dataset $\mathcal{L}$ generated by an offline, open-loop process and a set of prospective sample locations $\mathcal{U}$ (Step 1). The procedure constructs the initial SVM prediction model from this training dataset (Step 2). Unlike the sequential algorithm, the batch procedure selects T batches of M samples, where it is assumed $N_0 + TM = N_{lim}$. The active sampling process is contained within Steps 3-12. In each batch, the process sequentially selects the M points according to the combined criterion in (3.18) (Steps 4-8). Once all M locations have been selected, tests are performed at each of the locations and the procedure adds this information to training dataset $\mathcal{L}$ (Steps 9-10). The procedure then retrains the SVM model to incorporate the new observations (Step 11). This process repeats until the sampling budget has been exhausted and the procedure returns the final predictions for $\widehat{\Theta}_{sat}$ and $\widehat{\Theta}_{fail}$ (Step 13).

The computational cost of the batch procedure is similar to Algorithm 1. The batch active sampling process requires additional operations when compared to the passive, random sampling approach. In particular, the selection criterion in Step 6 requires $\mathcal{O}(N|\mathcal{U}|)$ operations to rank the available sample locations according to their SVM output and $\mathcal{O}(M|\mathcal{U}|)$ operations to calculate the diversity measure for all M points in the batch. This combines with the $\mathcal{O}(N^2)$ cost of retraining the SVM in Step 11 for a total cost of $\mathcal{O}(N|\mathcal{U}|) + \mathcal{O}(M|\mathcal{U}|) + \mathcal{O}(N^2)$ operations for each

Algorithm 2 Batch closed-loop deterministic verification framework using SVM classification models

```
1: Input: initial training set $\mathcal{L} = \{\mathcal{D}, \mathbf{y}\}$, available sample locations $\mathcal{U}$, # of iterations $T$, batch size $M$
2: Initialize: train SVM model $H_{\mathbb{R}}(\boldsymbol{\theta})$
3: for $i = 1, 2, \ldots, T$ do
4:     Initialize: $\mathcal{S} = \emptyset$
5:     for $k = 1, 2, \ldots, M$ do
6:         Select $\overline{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \mathcal{U}}{\operatorname{argmin}} \Big(\lambda \big|H_{\mathbb{R}}(\boldsymbol{\theta})\big| + (1-\lambda) \max_{\boldsymbol{\theta}_j \in \mathcal{S}} \big|\cos\big(\angle(\phi(\boldsymbol{\theta}), \phi(\boldsymbol{\theta}_j))\big)\big|\Big)$
7:         Add $\overline{\boldsymbol{\theta}}$ to $\mathcal{S}$, remove $\overline{\boldsymbol{\theta}}$ from $\mathcal{U}$
8:     end for
9:     Perform tests $\forall \boldsymbol{\theta} \in \mathcal{S}$, obtain measurements $\mathbf{y}_{\mathcal{S}}$
10:    Add $\{\mathcal{S}, \mathbf{y}_{\mathcal{S}}\}$ to training set $\mathcal{L}$
11:    Retrain model $H_{\mathbb{R}}(\boldsymbol{\theta})$ with updated $\mathcal{L}$
12: end for
13: Return: predicted sets $\widehat{\Theta}_{sat}$, $\widehat{\Theta}_{fail}$
```

iteration. While the batch process does require $\mathcal{O}(M|\mathcal{U}|)$ more operations for each iteration than was required in Algorithm 1, the total cumulative cost for the same number of $N_{lim} - N_0$ points is lower. Assuming the sequential and batch procedures share the same budget N_{lim}, the batch procedure will require fewer iterations since the remaining $N_{lim} - N_0$ points are broken into batches of $M > 1$ points. However, the exact savings will vary with each particular example since the size of $\mathcal{U}$, N_{lim}, N_0, and M will change in every example. Figure 3-4 demonstrates the improvement in computational complexity produced by the batch sampling approach in Example 3.5.2. Algorithm 2 produces a lower complexity than the sequential approach in Algorithm 1 and this improvement increases with larger batch sizes M.

While batch sampling methods do offer improved computational efficiency for the same number of total training points, it is important to recognize that they do not completely supplant sequential methods, but rather supplement them whenever appropriate. In some problems, it may not be possible to perform multiple tests in parallel. When this occurs, it is generally advisable to treat the problem with a sequential procedure in order to ensure every sample is exploited to its fullest effect when selecting new trajectories.

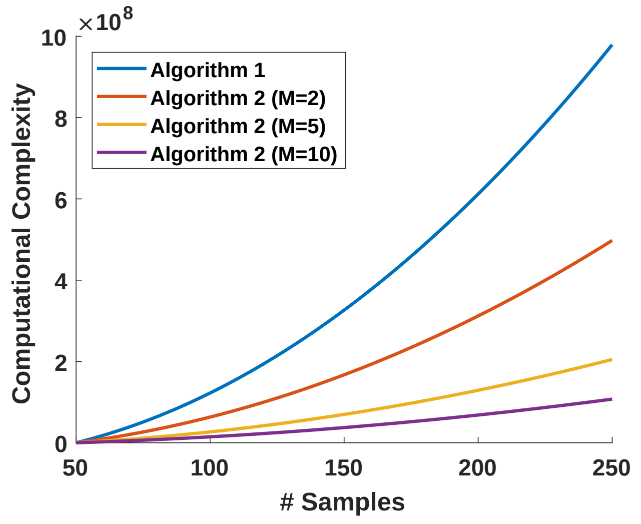


Figure 3-4: Computational complexity of the sequential (Algorithm 1) and batch (Algorithm 2) closed-loop verification procedures when applied to Example 3.5.2. By reducing the number of retraining steps, Algorithm 2 lowers the computational complexity required for active sampling. The exact reduction varies according to the example, but a larger batch size M will lower the complexity.

3.5 Simulation Results

The following section presents simulation results for three examples in nonlinear system verification. The results compare active learning-based, closed-loop statistical verification against traditional design of experiments (DOE) techniques for "open-loop" statistical verification. The section will also discuss the effectiveness of model validation methods for estimation of prediction error in these techniques. Additionally, the first two examples will contrast both open- and closed-loop statistical verification techniques with simulation-guided deductive barrier certificates discussed in Section 3.2. All of the simulation results demonstrate the improved prediction accuracy of active closed-loop statistical verification in comparison to other techniques, particularly when limited to small sampling budget.

3.5.1 Van der Pol Oscillator

The first example considers the same well-studied [51, 120] unstable Van der Pol oscillator discussed previously in the chapter in Sections 3.2 and 3.3. Due to the fact barrier certificates are known to provably verify the dynamics and are freely available, this example is also particularly useful for contrasting the statistical methods against analytical certificates.

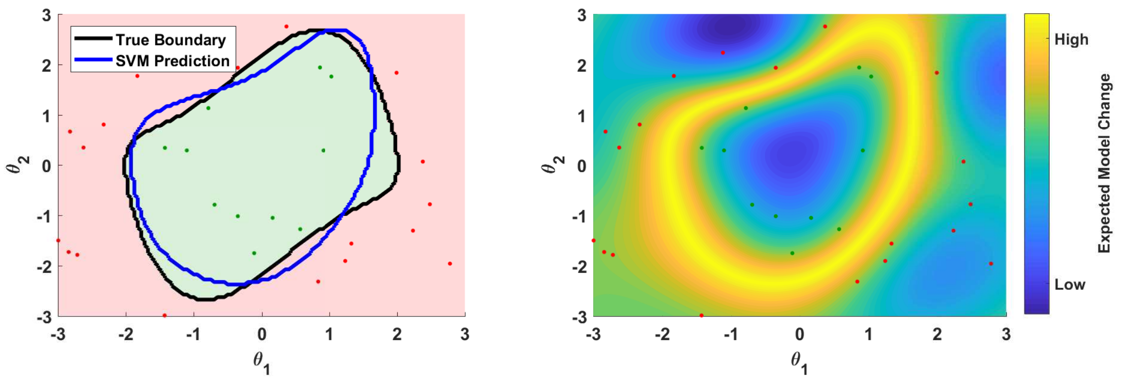
The nonlinear dynamics for the two-state system are

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -x_2 \\ x_1 + (x_2^2 - 1)x_1 \end{bmatrix}, \tag{3.19}$$

which are known to have an unstable limit cycle and an asymptotically stable equilibrium point at the origin. The perturbations $\boldsymbol{\theta}$ are uncertainties on the initial conditions, $\boldsymbol{\theta} = [x_1(0), x_2(0)]^T$. The verification goal is to determine the "region-of-attraction" (ROA): determine $\boldsymbol{\theta}$ conditions that will lead to convergence to the origin and those that will lead the trajectory to diverge. These conditions correspond to the $\Theta_{sat}/\Theta_{fail}$ sets of interest, already pictured in Figure 3-1(a). A successfully converged trajectory is indicated by $y(\boldsymbol{\theta}) = 1$ while a trajectory that diverged is indicated by $y(\boldsymbol{\theta}) = -1$.

For this example, the feasible set of initial conditions is restricted to $x_1(0) : [-3, 3]$ and $x_2(0) : [-3, 3]$, well outside the ROA boundary that separates $\Theta_{sat}/\Theta_{fail}$. This two-dimensional space is covered by a fine lattice Θ_d of 14,641 points. The closed-loop verification frameworks start with an initial training dataset of 20 trajectories randomly chosen from Θ_d. Note that since a barrier certificate is known to exist, the initial training dataset could have been provided or produced by simulation-guided deductive methods much like in Section 3.3.2. In addition to closed-loop statistical verification, open-loop data-driven verification is performed for comparison. These approaches use Latin hypercube and uniform random space-filling design of experiments (DOE) techniques [73, 74] to passively generate training datasets from Θ_d. In these two techniques, all of the trajectories' initial conditions $\boldsymbol{\theta}$ for $\mathcal{L}$ are selected at once and without any sort of feedback, hence are referred to as "open-loop".

Both sequential ($M = 1$) and batch ($M = 5$) versions of the closed-loop verification framework are analyzed. Figure 3-5 displays an SVM-based prediction model after 30 training samples and the corresponding ranking of prospective sample loca-

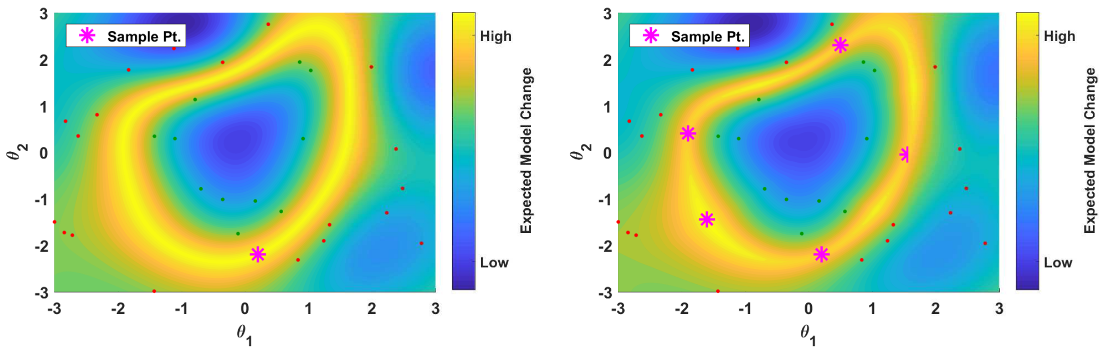


(a) SVM prediction model (b) Ranking of Θ_d according to (3.16)

Figure 3-5: [Example 3.5.1] Ranking of prospective sample locations based upon the expected model change metric. The SVM prediction model and corresponding expected model change metric are computed after 30 samples. Initial conditions $\boldsymbol{\theta} = [\theta_1, \theta_2]$ for the simulations are shown as dots and their respective colors denote whether the resulting trajectory satisfied the requirement (green) or did not (red).

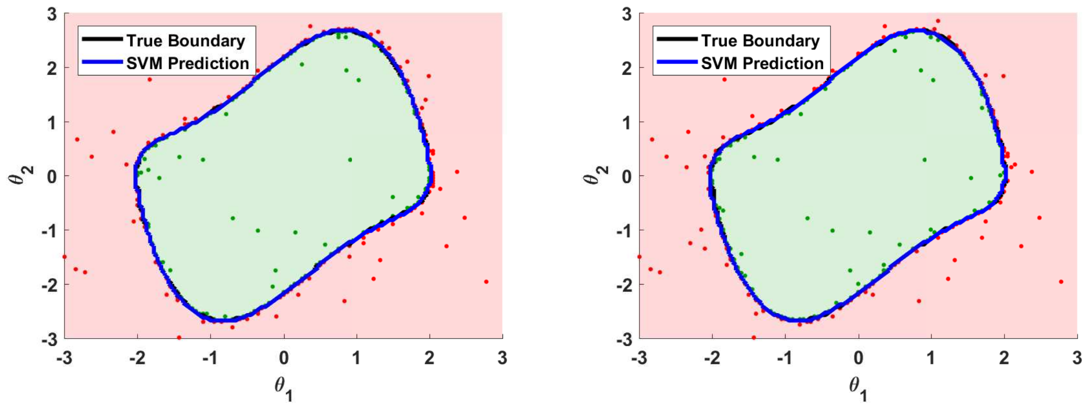
tions according to (3.16). Areas of high expected model change straddle the predicted $\widehat{\Theta}_{sat}/\widehat{\Theta}_{fail}$ boundary line. Figure 3-6 compares the selection of datapoints according to the sequential and batch closed-loop verification algorithms, where both use the same model and baseline sample ranking from Figure 3-5. The effect of the diversity metric is readily apparent in Figure 3-6(b). The sample locations are distributed around areas of high expected model changed. It is also possible to see the slight devaluing of regions surrounding the chosen samples when it is compared to Figure 3-6(a). This devaluing is a direct result of the diversity metric to discourage redundant samples. The active sampling algorithms are allowed to run for a total of 250 additional training samples. Figure 3-7 displays the final training dataset and prediction model at the conclusion of both the sequential and batch processes. Both models have converged to the correct boundary, which is almost indistinguishable from the predicted boundary, indicating that the algorithms have produced accurate $\widehat{\Theta}_{sat}$ and $\widehat{\Theta}_{fail}$.

Figure 3-8 compares the total misclassification error, the percentage of $\widehat{y}(\boldsymbol{\theta}_i) \neq y(\boldsymbol{\theta}_i)$ for all $\boldsymbol{\theta}_i \in \Theta_d$, for the various verification procedures over 100 randomly initialized runs. In both sequential and batch cases, the closed-loop statistical verification



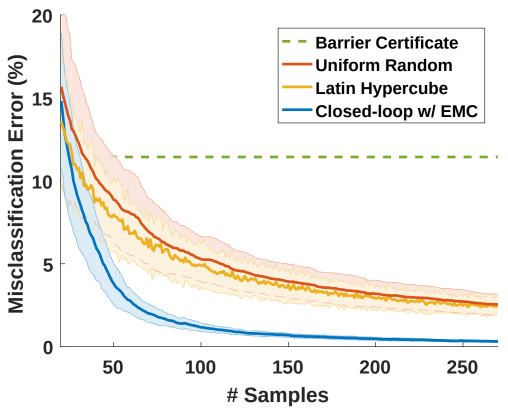
(a) Sequential (batch size $M = 1$) (b) Batch size $M = 5$

Figure 3-6: [Example 3.5.1] Comparison of sample selections for batch sizes of $M = 1$ and $M = 5$. Sample selections are shown as magenta stars.

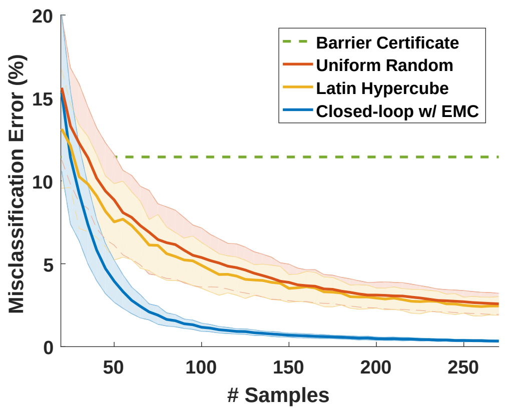


(a) Sequential (batch size $M = 1$) (b) Batch size $M = 5$

Figure 3-7: [Example 3.5.1] Comparison of prediction models after the collection of 250 training samples with Algorithms 1 and 2. Both predictions models have converged to roughly the same result, an accurate approximation of the true boundary.



(a) Sequential (batch size $M = 1$)



(b) Batch size $M = 5$

Figure 3-8: [Example 3.5.1] Comparison of the misclassification error convergence of open- and closed-loop statistical verification techniques. The lines show the mean (solid lines) and 1σ standard deviation for each statistical verification approach after 100 random initializations. For comparison, the figure also displays the misclassification error from the simulation-guided barrier certificate in Table 3.1. This error is shown as a straight line because it was taken directly from the results in [53].

procedure outperforms the two open-loop DOE-based procedures. The closed-loop verification procedure also produces the lowest standard deviation out of the three statistical verification approaches. From these results, it is clear the closed-loop verification frameworks will possess the best prediction accuracy given a limited amount of simulations.

Additionally, Figure 3-8 displays the 11.44% misclassification error produced by the 2nd order, simulation-guided barrier certificate from Section 3.2. The barrier certificate's objective is to avoid false positives, so all its misclassification errors correspond to false negatives. The figure displays the misclassification error as a straight line since this barrier certificate was taken directly from the results in [53], which used 221 simulations to generate the barrier certificate. It's also important to note that the misclassification error would change given a different Lyapunov function.

Evaluation of Validation Methods

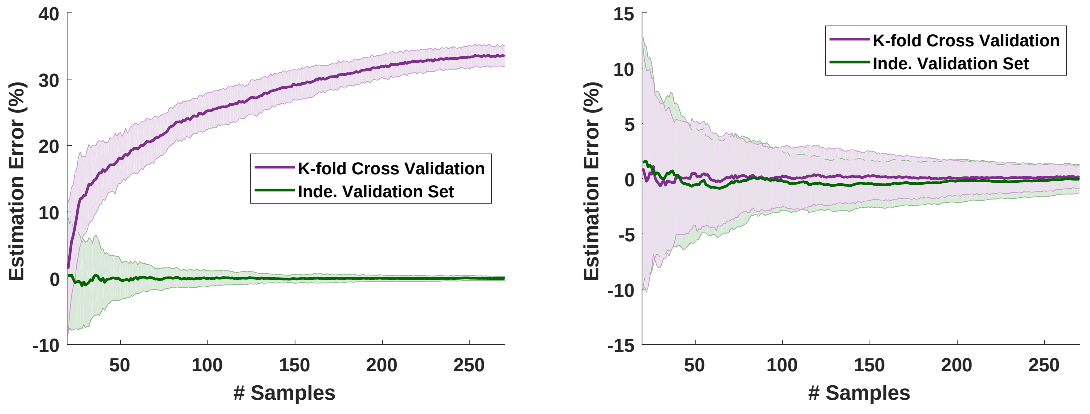
Although Figure 3-8 illustrates the convergence of misclassification errors and highlights the improvement afforded by closed-loop verification, this error value would not be known online. It requires the true labels $y(\boldsymbol{\theta})$ to already be known for every $\boldsymbol{\theta} \in \Theta_d$. Instead, Section 3.3.1 discussed two validation methods to estimate total misclassification error online: K-fold cross-validation and validation on an independent dataset. Of these, K-fold cross-validation is generally preferable as it only requires the current training set $\mathcal{L}$ while an independent validation dataset subtracts from the number of samples allocated to $\mathcal{L}$.

The estimation errors of the two validation methods applied to sequential closed-loop and open-loop verification are illustrated in Figure 3-9. Independent validation datasets are shown to accurately estimate the total misclassification error for both open- and closed-loop approaches. The decreased standard deviation of the active learning-based closed-loop procedure minimizes the estimation error when compared to the passive, random sampling DOE method. However, the estimation error for K-fold cross-validation degrades with additional samples when it is applied to the active learning procedure. This poor performance is due to the fact that active learning concentrates datapoints along the decision boundary in comparison to open-loop methods which spread the datapoints over the full space. In the active learning approach, each validation fold (subset) of the training dataset will contain a high number of points in this likely-to-be-misclassified region, thus producing a higher (and inaccurate) misclassification error than a uniformly distributed validation set. This suggests that K-fold cross-validation cannot be used to accurately estimate total misclassification error for closed-loop verification.

3.5.2 Concurrent Learning Model Reference Adaptive Controller

The second example is a model reference adaptive control (MRAC) system. In this problem, concurrent learning adaptive control (CL-MRAC) [121] is used to control an uncertain, second-order linear system

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -0.2+\theta_1 & -0.2+\theta_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \ . \tag{3.20}$$



(a) Closed-loop verification w/ EMC (b) Open-loop w/ uniform random DOE

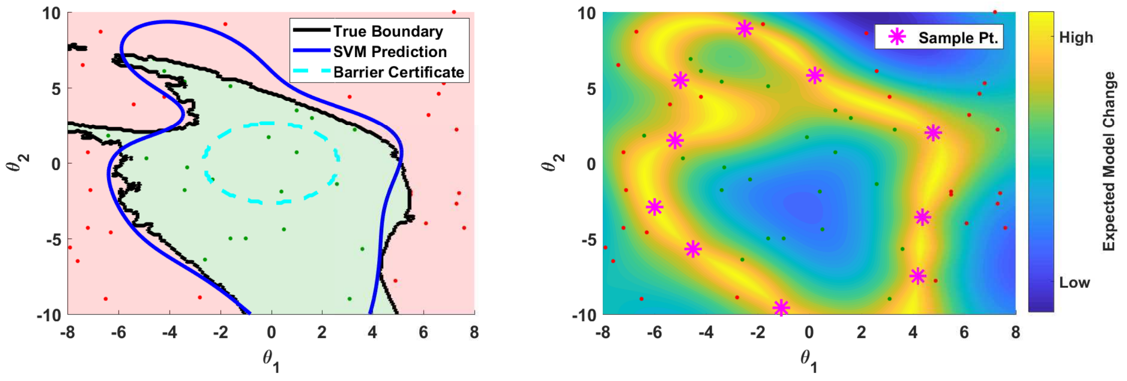
Figure 3-9: [Example 3.5.1] Comparison of estimation error using K-fold cross validation and validation on an independent validation dataset. Both figures are derived from the results of the sequential ($M = 1$) procedure. Note: results for the open-loop Latin hypercube DOE approach are not shown but are consistent with the random sampling technique.

Just as in the previous example, there are two sources of uncertainty under consideration $\boldsymbol{\theta} = [\theta_1, \theta_2]^T$; however, these terms correspond to uncertain system parameters rather than initial states $\mathbf{x}(0)$. The adaptive controller actually estimates these parameters while simultaneously steering the system to track a desired reference trajectory produced by a linear reference system. For easier viewing, the detailed discussion of the adaptive controller for $u(t)$ is found in Appendix A. Due to the adaptation, the resulting closed-loop system is highly nonlinear and difficult to analyze.

Although the CL-MRAC controller guarantees asymptotic closed-loop stability, the performance of the closed-loop system is based upon boundedness of the tracking error between the actual states and desired reference trajectory. The performance requirement is for the actual state $x_1(t)$ to remain within unit error of the reference state $x_{m_1}(t)$ at every point along the 40 second trajectory. Described in temporal logic format, this requirement states

$$\varphi_{bound} = \square_{[0,40]} \left(1 - |e_1[t]| \geq 0\right) \tag{3.21}$$

where $e_1(t) = x_{m_1}(t) - x_1(t)$ is the tracking error. The goal of the verification procedure is to determine which $\boldsymbol{\theta}$ vectors will result in trajectories that stay within this



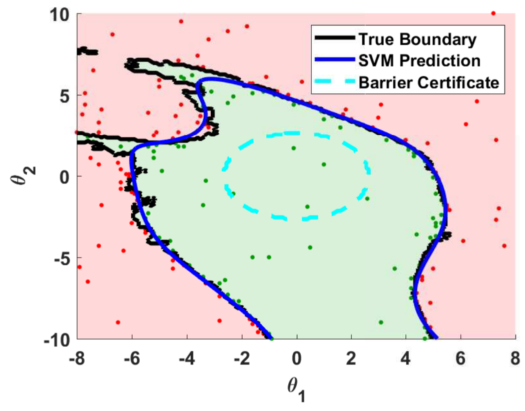
(a) SVM prediction model

(b) Ranking of Θ_d according to (3.16) and selection of sample points

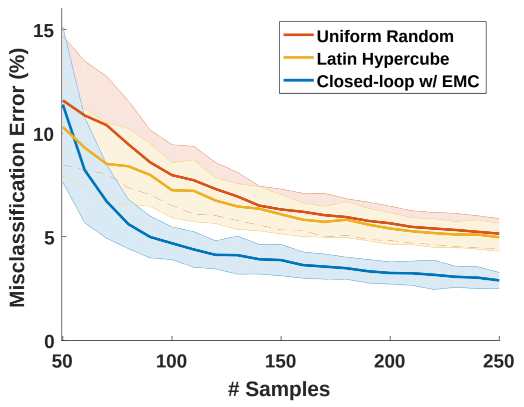
Figure 3-10: [Example 3.5.2] Ranking of prospective sample locations based upon the expected model change metric. The SVM prediction model and corresponding expected model change metric are computed after 50 samples. Parameter settings $\boldsymbol{\theta} = [\theta_1, \theta_2]$ for the simulations are shown as dots and their respective colors denote whether the resulting trajectory satisfied the requirement (green) or did not (red).

bound ($y(\boldsymbol{\theta}) = 1$) and those that will lead to failure ($y(\boldsymbol{\theta}) = -1$).

In this example, the set of all possible sample locations Θ_d is a lattice of 32,400 points covering $\theta_1 : [-8, 8]$ and $\theta_2 : [-10, 10]$. The open- and closed-loop verification procedures all start with an initial training dataset $\mathcal{L}$ of 50 randomly-initialized simulation trajectories and corresponding binary measurements. The closed-loop active sampling process operates in batches of 10 points. Figure 3-10 displays the SVM prediction model trained on an initial dataset of 50 samples and the first batch of 10 points selected according to the expected model change metric. The closed-loop procedure runs for 20 more iterations for a total of 250 training samples. Figure 3-11(a) depicts the SVM model at the conclusion of this process and it compares favorably to the barrier certificate produced by simulation-guided deductive techniques [109]. Figure 3-11(b) illustrates the convergence of the misclassification error for the statistical verification techniques. For comparison, the simulation-guided analytical barrier certificate method in [109] produced a 34% misclassification error rate, although all these errors correspond to false negatives since barrier certificates use a different objective. Just as in the Van der Pol example, closed-loop verification using active learning noticeably outperforms both the open-loop statistical verification methods



(a) Final SVM model after 250 samples



(b) Misclassification error convergence

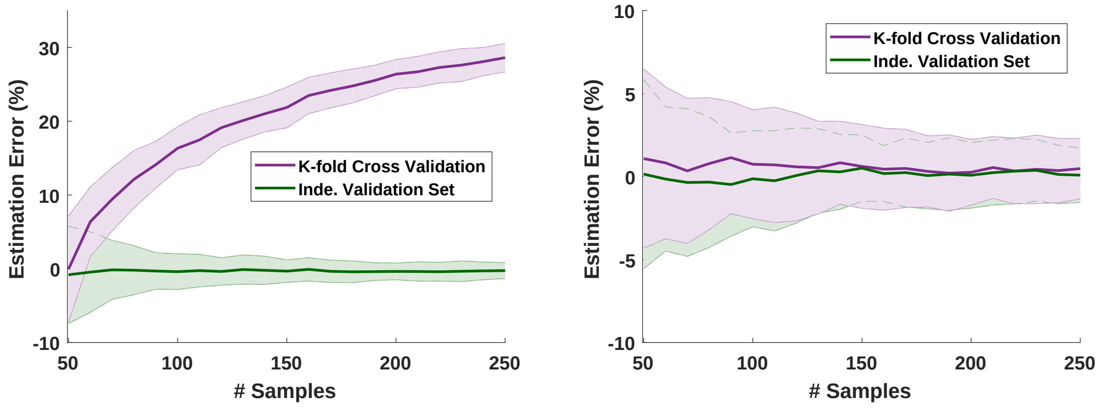
Figure 3-11: [Example 3.5.2] Misclassification error convergence of statistical verification techniques. The left plot displays the SVM prediction at the completion of $N_{lim} = 250$ training samples for the closed-loop procedure. The right plot shows the mean (solid lines) and 1σ standard deviation for each approach in 100 random initializations.

as well as current analytical barrier certificate techniques when the latter are applied to the binary classification problem.

Figure 3-12 compares the estimation accuracy of online validation using K-fold cross-validation and independent datasets. Just as was seen in the previous example in Figure 3-9, K-fold cross-validation fails to accuracy estimate the misclassification error when applied to active learning approaches. This reaffirms the previous results and further highlights issues with online estimation of prediction confidence for active learning-based approaches.

Effect of Penalizing False-Positives

The complexity of the decision boundary in this example highlights the effect of hyperparameters, namely the box constraint C, on misclassification errors. In Figure 3-11(a), even after 250 training samples, the SVM prediction model cannot completely capture the true boundary. More importantly, this figure also demonstrates a visible area of false-positive misclassification errors in the upper-left corner of the predicted boundary. These errors are typically considered worse than false negatives as they involve accidentally labeling unsatisfactory points as satisfactory. Section 3.3.1 pre-



(a) Closed-loop verification w/ EMC (b) Open-loop w/ uniform random DOE

Figure 3-12: [Example 3.5.2] Comparison of estimation error using K-fold cross validation and validation on an independent validation dataset. Note: results for the open-loop Latin hypercube DOE approach are not shown but are consistent with the other open-loop technique.

sented an extension of the constraint hyperparameter C to penalize false positives with a term C_{FP}. Figure 3-13 illustrates the effect of this penalty term as it is increased from the default value of 1 up to $C_{FP} = 6$. This term adds conservativeness to the SVM model, sacrificing total misclassification error with a larger number of false negatives for a diminishing number of false-positive errors. For the prediction model shown in Figure 3-13, the total misclassification error rate increases from 2.67% at $C_{FP} = 1$ to 9.50% at $C_{FP} = 6$, but the rate of false-positive errors decreases from 2.80% to 0. While it cannot explicitly restrict the rate of false-positive errors to a certain limit, the hyperparameters can be used to indirectly adjust the different types of misclassification errors.

3.5.3 Adaptive System with Control Saturation

The third example is a more complex CL-MRAC system with control saturation. The open-loop dynamics are the same as in (3.20) except the control input $u(t)$ is saturated between limits $-u_{max} \leq u(t) \leq u_{max}$. In order to counter the adverse effects of control saturation upon reference trajectory tracking, the baseline CL-MRAC controller is augmented with pseudo-control hedging (PCH) [122]. More information on

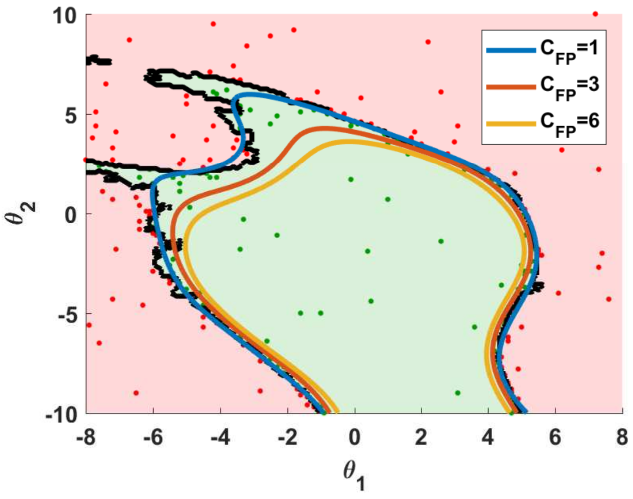


Figure 3-13: [Example 3.5.2] Effect of increasing false-positive penalties in the SVM prediction model. All three models have the same training data (shown as dots), but the C_{FP} term used to penalize false positives is increased from the default 1 to 6.

CL-MRAC with control saturation and PCH is provided in Appendix A. This new formulation increases the complexity of the closed-loop system and further complicates analytical verification of the system. Unlike the previous two examples, there is no known barrier certificate to compare to statistical verification techniques. This fact highlights the wider applicability of statistical verification procedures over existing analytical barrier certificate approaches.

In addition to changes in the control architecture, this example also has a more complex set of performance requirements for the system to satisfy. The closed-loop system is expected to satisfy three separate conditions. First, at some point within the time interval $t = [2, 3]$, the trajectory must visit the region $x_1(t) \in [0.7, 1.3]$,

$$\varphi_1 = \Diamond_{[2,3]} \; (x_1[t] - 0.7 \geq 0) \wedge \Diamond_{[2,3]} \; (1.3 - x_1[t] \geq 0). \tag{3.22}$$

Similarly, the trajectory must also reach a state within $x_1(t) \in [1.1, 1.7]$ at some point between $t = [12, 13]$, i.e.

$$\varphi_2 = \Diamond_{[12,13]} \; (x_1[t] - 1.1 \geq 0) \wedge \Diamond_{[2,3]} \; (1.7 - x_1[t] \geq 0). \tag{3.23}$$

Lastly, the system must satisfy $x_1(t) \in [-1.6, -1.2]$ at $t = 22.5$ seconds in order to satisfy the third requirement, roughly expressed as

$$\varphi_3 = \square_{[22.4,22.6]} \left(x_1[t] + 1.6 \geq 0\right) \wedge \square_{[22.4,22.6]} \left(-1.2 - x_1[t] \geq 0\right) . \tag{3.24}$$

The complete requirement is the conjunction of all three: $\varphi = \varphi_1 \wedge \varphi_2 \wedge \varphi_3$. In order for the trajectory to satisfy φ (i.e. $y(\boldsymbol{\theta} = 1)$), all three requirements must be met. If only one or two of the requirements are met, the trajectory is still labeled with the "unsatisfactory" measurement $y(\boldsymbol{\theta}) = -1$.

This example shares the same two uncertain parameters (θ_1, θ_2) as the preceding problem, but add two new sources: θ_3 captures uncertainty in the initial state $x_1(0)$ and θ_4 models variations in the control saturation limit u_{max}. The statistical procedure constructs a 4-dimensional lattice Θ_d of 2.356 million possible sample locations covering $\theta_1 : [-5, 5]$, $\theta_2 = [-5, 5]$, $\theta_3 : [-1, 1]$, and $\theta_4 : [3, 8]$. The open- and closed-loop statistical verification procedures each start with an initial training dataset of 50 randomly-selected simulation trajectories chosen from Θ_d. The closed-loop sampling procedure operates in batches of 10 points up to a sampling budget $N_{lim} = 750$ training points.

The total misclassification error produced by the statistical verification procedures in 100 randomly initialized runs is shown in Figure 3-14. In this example, the active learning-based procedure significantly outperforms the passive open-loop processes. At the conclusion of the 750 training samples, the average misclassification error of the closed-loop procedure is less than half the error produced by open-loop verification. Additionally, the standard deviation of the closed-loop procedure is lower than the open-loop approaches. This example in particular illustrates the improved performance of closed-loop statistical verification over open-loop variants and highlights the wider applicability of statistical verification to problems without suitable or appropriate analytical verification methods.

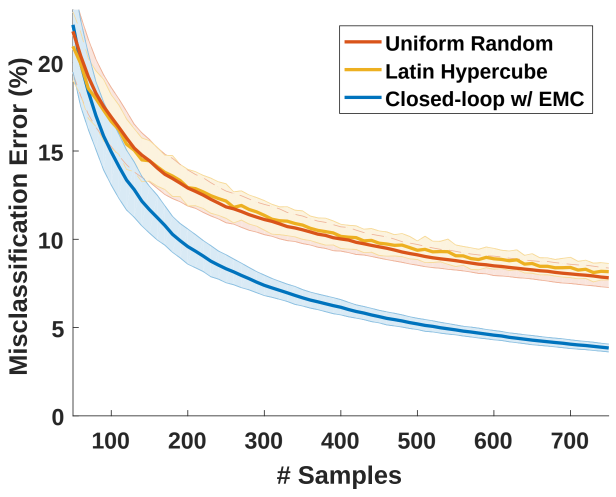


Figure 3-14: [Example 3.5.3]: Comparison of the misclassification error convergence of open- and closed-loop statistical verification techniques. The plot shows the mean (solid lines) and 1σ standard deviation for each approach in 100 random initializations.

3.6 Summary

This chapter presented the development of data-driven procedures for statistical verification of arbitrary deterministic nonlinear systems. In particular, the work in this chapter addressed the problem of binary verification, where a trajectory's satisfaction of performance requirements is measured with a binary evaluation, and the set of all possible perturbation conditions can be partitioned exactly into two distinct sets corresponding to their binary evaluation. This SVM-based data-driven approach compliments existing simulation-based analytical verification methods, but applies to a much wider class of systems that cannot be addressed with these analytical methods. More importantly, the data-driven approach enables "closed-loop" verification procedures that actively select future trajectories to best improve the statistical prediction model. Simulation results on three nonlinear systems of increasing complexity illustrate the strengths of statistical verification methods over deductive techniques and the improvement in prediction error with closed-loop verification procedures over traditional design of experiments.

Chapter 4

Deterministic Verification with Improved Evaluations of Trajectory Robustness

This chapter considers an extension of the statistical verification approaches from Chapter 3 to address the challenge with suitable online validation techniques for the learned prediction model. The binary nature of the measurements limits the types of applicable classification models and restricts the ability to adequately estimate the model's prediction accuracy online in a sample-efficient manner. The results demonstrated K-fold cross-validation's inability to effectively estimate the error rate for active learning-based closed-loop verification since the clustering of training data near the decision boundary biases the error approximations. Without K-fold cross-validation, only costly independent validation datasets can be used to estimate the prediction accuracy.

This chapter will directly address this limitation in systems capable of providing continuous measurements of satisfactory performance. As the class of systems with these continuous measurements is a subset of the class of systems considered in Chapter 3, the overall problem is mostly unchanged. This chapter presents an alternative Gaussian process regression-based formulation for binary verification. The main difference is that while the new approach still provides the same predictions, it also quantifies the confidence in those predictions without relying upon external validation methods. This new information also enables modifications to the closed-loop statistical verification procedures to further reduce the rate of prediction errors for a fixed number of trajectories.

4.1 Problem Description

The systems considered in this work are a subset of the class of systems included in Chapter 3. While this subset does exclude some systems discussed in the preceding chapter, the additional assumptions made in this work are not overly restrictive. The overall problem formulation itself remains almost entirely unchanged except for the inclusion of continuous measurements in place of solely binary evaluations.

Consider the same deterministic nonlinear system originally described in (3.1)-(3.3). The system is still subject to the same parametric uncertainties $\boldsymbol{\theta}$, which are assumed to fall within known compact set Θ just as in Assumption 3.1. The system's closed-loop trajectory also remains unchanged; trajectory $\Phi(\mathbf{x}(t)|\mathbf{x}_0, \boldsymbol{\theta})$ defines the time evolution of state vector $\mathbf{x}(t)$ over time interval $t \in [0, T_{final}]$ subject to nominal initial condition $\mathbf{x}_0$ and parameters $\boldsymbol{\theta}$.

4.1.1 Continuous Measurements of Performance Requirement Satisfaction

The system's trajectory $\Phi(\mathbf{x}(t)|\mathbf{x}_0, \boldsymbol{\theta})$ is evaluated against pre-specified performance requirements supplied by some appropriate certification agency or expert. The satisfaction of a requirement is still binary in nature, i.e. the trajectory either "satisfied" or "did not satisfy" the requirement. However, the these binary evaluations are assumed to be indicated through the sign of continuous measurements.

Assumption 4.1. *There exists an oracle which provides deterministic continuous measurements of whether a trajectory $\Phi(\mathbf{x}(t)|\mathbf{x}_0, \boldsymbol{\theta})$ satisfied the performance requirement under consideration. These measurements are output as scalar variable $y \in \mathbb{R}$*

where the sign of y indicates satisfaction of the requirement. Positive measurement $y > 0$ corresponds to "satisfied" while $y \leq 0$ corresponds to "did not satisfy."

Remark 4.2. *The measurement $y = 0$ is ambiguous, but assumed to indicate failure in this work.*

As before, "oracle" is used as a generic catch-all term to capture a wide variety of possible sources for continuous measurement y. In many problems, the requirements can be written using signal temporal logic (STL) discussed in Section 2.3. In these problems, the continuous measurement is simply the STL robustness degree ρ^{φ}. While signal temporal logic will be a common source for the measurement, it is not the only one. One example later in this chapter will examine the execution of an ordered set of agents' tasks in a robust multi-agent task allocation problem. The corresponding measurement y outputs the difference between the plan's realized score and the minimum acceptable score. There are many other possible sources for y.

Regardless of the source, the continuous measurement provides an additional layer of information that can be used to rank the robustness of points with the same sign. For instance, if the requirement states the trajectory must remain above a certain threshold, measurement y indicates the difference between the lowest point in the trajectory and the threshold. Positive values indicate that the trajectory successfully remained above it, while negative values indicate just how far below the threshold the trajectory reached at some point. Two positive measurements $y_2 > y_1 > 0$ both signify satisfactory trajectories, but the trajectory with y_2 is considered "more robust" as it remains further away from the limit that delineates unsatisfactory performance. Figure 4-1 illustrates this fact in a simple example problem. The second trajectory remains farther away from the failure boundary ($x = 0.7$) and is therefore considered more robust whereas binary evaluations would rank both trajectories equally. This additional layer of information can be directly incorporated into statistical verification in order to address the limitations resulting from purely binary measurements. Just like the binary measurements in Chapter 3, continuous measurement y is an explicit function of parameters $\boldsymbol{\theta}$ and is written as $y(\boldsymbol{\theta})$ to emphasize this fact.

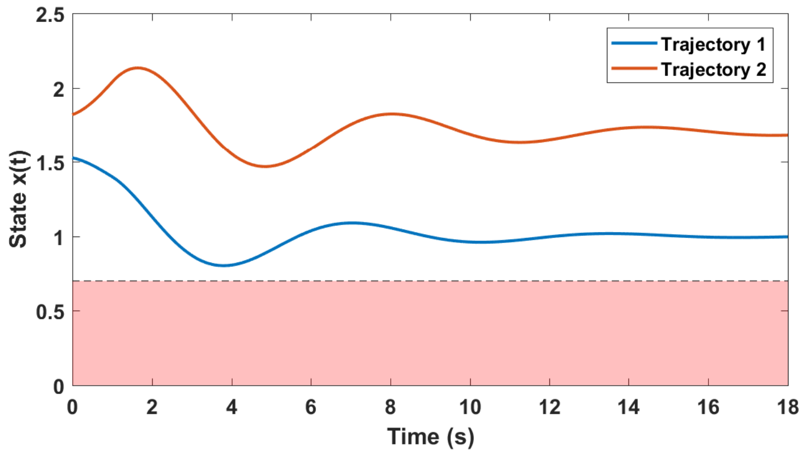


Figure 4-1: Comparison of two trajectories with the same binary evaluation but different continuous measurements. Both trajectories satisfy the requirement (stay above $x = 0.7$), but the first trajectory is less robust than the second since it strays closer to the failure limit, particularly at $t = 4$.

Systems with Multiple Requirements

While Section 3.1.1 discussed the use of binary evaluations $y(\boldsymbol{\theta}) \in \{-1, 1\}$ to signify the satisfaction of either a single requirement or an entire set of requirements, each continuous measurement $y(\boldsymbol{\theta}) \in \mathbb{R}$ can only measure the satisfaction of a single requirement. The main reason for this restriction is the robustness information provided by the non-binary measurement is tied to a particular requirement. For example, if a system under consideration has two requirements of interest, one corresponding to position (measured in feet) and another corresponding to angular position (in degrees), it is straightforward for binary evaluations to indicate whether both requirements were simultaneously satisfied since they only consider the Boolean "yes/no" result. However, a single non-binary measurement $y(\boldsymbol{\theta}) \in \mathbb{R}$ cannot simultaneously measure the robustness of the trajectory to both requirements. In order to signify the satisfaction of both requirements, two separate, parallel measurements are needed. One measurement will measure the robustness of the trajectory to the first requirement (in feet), while a second measurement will measure the trajectory's robustness to the second requirement (in degrees). At the most fundamental level, the units corresponding to each requirement are different and thus a single measurement cannot measure the robustness of the trajectory to both. This highlights an important limitation of continuous measurements.

Region of Satisfaction

Due to the addition of continuous measurements $y(\boldsymbol{\theta}) \in \mathbb{R}$, the definitions of the regions of satisfaction and failure are slightly modified.

Definition 4.3. *The* region of satisfaction Θ_{sat} *contains all* $\boldsymbol{\theta} \in \Theta$ *for which the resulting trajectory satisfies the performance requirement, i.e.*

$$\Theta_{sat} := \Big\{\boldsymbol{\theta} \in \Theta : y(\boldsymbol{\theta}) > 0\Big\}. \tag{4.1}$$

Definition 4.4. *The* region of failure Θ_{fail} *contains all remaining* $\boldsymbol{\theta} \in \Theta$ *for which the resulting trajectory fails to satisfy the performance requirement, i.e.*

$$\Theta_{fail} := \Big\{\boldsymbol{\theta} \in \Theta : y(\boldsymbol{\theta}) \leq 0\Big\}. \tag{4.2}$$

As before, both these sets are assumed to be non-empty, $\Theta_{sat} \neq \emptyset$ and $\Theta_{fail} \neq \emptyset$, and Θ_{fail} is the complement of Θ_{sat} so $\Theta_{sat} \cup \Theta_{fail} = \Theta$ and $\Theta_{sat} \cap \Theta_{fail} = \emptyset$. If the sign of the measurements is used to create a binary variable $y_{bin} = \text{sign}(y)$, the sets Θ_{sat} and Θ_{fail} specified by Definitions 4.3 and 4.4 are actually the same exact sets produced by Definitions 3.4 and 3.5 using y_{bin}. This will be illustrated later in the chapter during the example in Section 4.4.1. The ultimate goal of the verification process remains unchanged from Problem 3.1: compute an estimated region of satisfaction $\widehat{\Theta}_{sat}$.

4.2 Regression-based Binary Verification

The continuous measurements enable a drastically different approach than previously possible with only binary measurements. Rather than a binary *classification model*, a regression model can be fit to the training data to estimate $\widehat{y}(\boldsymbol{\theta}) \in \mathbb{R}$ from the continuous measurements. Although regression models replace SVM-based binary prediction models, the verification problem itself is still fundamentally binary classification. Given a queried condition $\boldsymbol{\theta}$, the goal is to predict whether $\boldsymbol{\theta}$ belongs to Θ_{sat} or Θ_{fail}. To emphasize the binary nature of the problem remains the same, the new approach is called *regression-based binary verification.*

4.2.1 Gaussian Process Regression Model

A finite collection of trajectory data (simulated or experimental) taken from Θ is used to form a regression model. There are a number of possible regression approaches that can be applied [81], but this work utilizes the common Gaussian process regression modeling technique [85], also known as Kriging. Section 2.2 provides additional information and background on Gaussian process regression models.

Informally, Gaussian process (GP) regression models define a distribution over possible $y(\boldsymbol{\theta})$ values at all $\boldsymbol{\theta} \in \Theta$ conditioned on the evidence provided by the finite collection of observed trajectories. This collection is the training dataset $\mathcal{L}$ consisting of N trajectories initialized at parameter vectors $\mathcal{D} = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_N\}$ with the corresponding $N \times 1$ vector of measurements $\mathbf{y} = [y(\boldsymbol{\theta}_1), y(\boldsymbol{\theta}_2), \ldots, y(\boldsymbol{\theta}_N)]^T$ ($\mathcal{L} = \{\mathcal{D}, \mathbf{y}\}$ as before). A posterior predictive distribution for $y(\boldsymbol{\theta})$ at unobserved locations in Θ is computed from the evidence provided by this training dataset and a pre-specified prior probability distribution.

The prior probability distribution is a joint multivariate Gaussian distribution between the training points in $\mathcal{L}$. Because nothing is assumed to be known about $y(\boldsymbol{\theta})$ beforehand, the prior distribution is initialized with zero mean to prevent inadvertently biasing the subsequent posterior distribution. The prior distribution over $\mathcal{L}$ is then given by

$$\mathbb{P}(\mathbf{y}|\mathcal{D}, \psi) = \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K}) \tag{4.3}$$

where $\mathbf{K}$ is the $N \times N$ covariance matrix with $\mathbf{K}_{ij} = \kappa(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j)$. There are a number of suitable kernel functions $\kappa(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j)$, but the squared exponential (Gaussian) kernel is by far the most popular. This approach uses the squared exponential kernel with automatic relevance determination (SE-ARD) from (2.14). The SE-ARD kernel is defined by a set of hyperparameters ψ that independently control the influence associated with each component in $\boldsymbol{\theta} \in \mathbb{R}^p$. This enables the kernel to devalue or minimize components of $\boldsymbol{\theta}$ to which $y(\boldsymbol{\theta})$ has shown low sensitivity and emphasize those to which $y(\boldsymbol{\theta})$ has shown high sensitivity.

The posterior probability distribution (2.16) over $\mathcal{L}$ is proportional to the product of the prior distribution and a likelihood model (2.15) for the measurements. Since these measurements $\mathbf{y}$ are noise-free observations of deterministic trajectories, there is no uncertainty as to the value of $y(\boldsymbol{\theta})$ at the training conditions. The likelihood model can be viewed as a collection of Dirac delta distributions or alternatively as Gaussians with widths set to 0. Unlike GP regression with noisy observations, there is no need for a second set of hyperparameters ϑ for the likelihood model.

Ultimately, the most important aspect of the GP regression model is the posterior predictive distribution used to predict $y(\boldsymbol{\theta})$ at unobserved locations in Θ. The posterior predictive distribution for measurement $y(\boldsymbol{\theta}_*)$ at arbitrary query location $\boldsymbol{\theta}_*$ is a Gaussian

$$\mathbb{P}(y(\boldsymbol{\theta}_*)|\mathcal{L}, \boldsymbol{\theta}_*, \psi) = \mathcal{N}\big(\mu(\boldsymbol{\theta}_*), \Sigma(\boldsymbol{\theta}_*)\big) \tag{4.4}$$

with posterior predictive mean $\mu(\boldsymbol{\theta}_*)$ and covariance $\Sigma(\boldsymbol{\theta}_*)$. These two terms are computed by

$$\begin{aligned} \mu(\boldsymbol{\theta}_*) &= \mathbf{K}_*^T \mathbf{K}^{-1} \mathbf{y} \\ \Sigma(\boldsymbol{\theta}_*) &= \mathbf{K}_{**} - \mathbf{K}_*^T \mathbf{K}^{-1} \mathbf{K}_* \end{aligned} \tag{4.5}$$

where scalar $\mathbf{K}_{**} = \kappa(\boldsymbol{\theta}_*, \boldsymbol{\theta}_*)$ and $\mathbf{K}_*$ is the $N \times 1$ vector of $\kappa(\boldsymbol{\theta}_*, \boldsymbol{\theta}_i)$ for $i = 1, \ldots, N$. The predicted value of the measurement at $\boldsymbol{\theta}_*$ is simply the posterior predictive mean $\mu(\boldsymbol{\theta}_*)$; in other words, $\widehat{y}(\boldsymbol{\theta}_*) = \mu(\boldsymbol{\theta}_*)$. At all the observed training locations from $\mathcal{L}$, the output of the posterior predictive mean matches the actual measurement, $\mu(\boldsymbol{\theta}) = y(\boldsymbol{\theta})\ \forall \boldsymbol{\theta} \in \mathcal{D}$. The predicted regions of satisfaction and failure are rewritten in terms of $\widehat{y}(\boldsymbol{\theta})$.

Definition 4.5. *The* predicted region of satisfaction $\widehat{\Theta}_{sat}$ *contains all* $\boldsymbol{\theta} \in \Theta$ *for* *which the resulting trajectory is predicted to satisfy the performance requirement,*

$$\widehat{\Theta}_{sat} := \Big\{\boldsymbol{\theta} \in \Theta : \widehat{y}(\boldsymbol{\theta}) > 0\Big\}. \tag{4.6}$$

Definition 4.6. *The* predicted region of failure $\widehat{\Theta}_{fail}$ *contains all remaining* $\boldsymbol{\theta} \in \Theta$ *for which the resulting trajectory is predicted to fail to satisfy the performance requirement,*

$$\widehat{\Theta}_{fail} := \Big\{\boldsymbol{\theta} \in \Theta : \widehat{y}(\boldsymbol{\theta}) \leq 0\Big\}. \tag{4.7}$$

Selection of Hyperparameters

The choice of kernel hyperparameters ψ may have a substantial effect upon the output of the GP regression model, and thus the predicted $\widehat{\Theta}_{sat}$,$\widehat{\Theta}_{fail}$. In this work, it is assumed the "ideal" or "true" hyperparameters that maximize the accuracy of the predictions $\mu(\boldsymbol{\theta})$ for all $\boldsymbol{\theta} \in \Theta$ are unknown in advance. In general, it will be difficult to correctly guess the hyperparameters without extensive prior knowledge or data. Later results in Section 4.4.1 will demonstrate the problem with incorrectly fixing the hyperparameters to suboptimal values.

If the hyperparameters are not known a priori, they will have to be estimated online using only the current available information, training dataset $\mathcal{L}$. The process is described in more depth in Section 2.2.3, but the end result is a new probability distribution over possible hyperparameter values $\mathbb{P}(\psi|\mathcal{L})$. In order to capture the full effects of the uncertain hyperparameters, the posterior predictive distribution (4.4) should be marginalized over $\mathbb{P}(\psi|\mathcal{L})$,

$$\mathbb{P}(y(\boldsymbol{\theta}_*)|\mathcal{L}, \boldsymbol{\theta}_*) = \int \mathbb{P}(y(\boldsymbol{\theta}_*)|\mathcal{L}, \boldsymbol{\theta}_*, \psi)\ \mathbb{P}(\psi|\mathcal{L})\ d\psi. \tag{4.8}$$

In practice, this integral is computationally intractable to obtain online, so it can be approximated through methods like sum-of-Gaussians (2.26), Markov Chain Monte Carlo, or maximum likelihood estimation. This work uses maximum likelihood estimation (2.27) to efficiently compute locally-optimum hyperparameters ψ^*, which is particularly important for later closed-loop verification procedures that will frequently update $\mathcal{L}$ and change the distribution $\mathbb{P}(\psi|\mathcal{L})$.

High-Dimensional and High-Volume Systems

The Gaussian process regression model discussed in this chapter is the baseline GP model. Although this representation is by far the most common, it has been shown to have difficulty with higher-dimensional systems [123], where the p in $\boldsymbol{\theta} \in \mathbb{R}^p$ is large ($p \geq 10$). More complex derivations of the baseline Gaussian process model can be employed for these high-dimensional systems or to address similar issues with high-volume data (size of Θ_d). These approaches [123,124] generally attempt to either decompose the GP into the sum of additive models or find a sparse approximation of the full GP, which becomes too computationally expensive. The focus of the work in this thesis is on the verification framework and selection metrics for closed-loop verification; therefore, these more-complex representations are not demonstrated in the results in this thesis, but the statistical verification frameworks could be easily be adapted to these approaches in high-dimensional systems.

Systems with Multiple Requirements

As discussed in Section 4.1.1, a single continuous measurement can only indicate the satisfaction of a single performance requirement. Most industrial-scale problems will require the simultaneous satisfaction of multiple requirements; therefore, the verification process in these applications will have to produce a matching number of measurements for every trajectory, with each measurement indicating the robustness of the trajectory to the corresponding requirement. In those problems, each requirement would require its own Gaussian process prediction model trained on the appropriate measurements. Although each Gaussian process would share the same training locations $\mathcal{D}$, they each model a fundamentally different regression surface. Likewise, each requirement's Gaussian process model should have its own kernel functions and hyperparameters. It is inadvisable to share hyperparameters between the Gaussian process models since the requirements may have very different sensitivities to parameter settings $\boldsymbol{\theta}$, requiring drastically different hyperparameters. Ultimately, the need for independent Gaussian processes matched to each requirement does translate into higher computational costs since the multiple GPs will have to be trained in parallel. The remainder of this chapter, as well as Chapters 5 and 7, will focus on verification problems with a single requirement and corresponding GP model. The future work section in Chapter 8 contains a more in-depth discussion of parallel GPs for multiple requirements, particularly on the extension of the closed-loop verification procedures in Section 4.3 to those problems.

4.2.2 Prediction Confidence

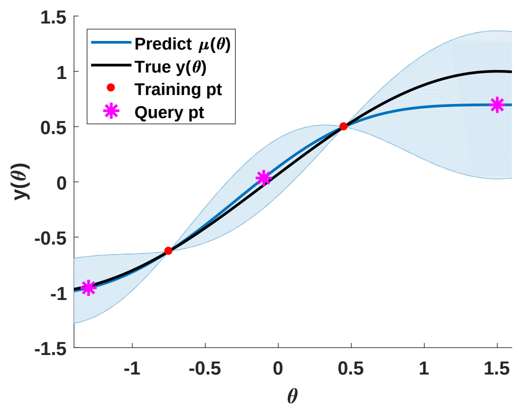
With the additional robustness information provided by continuous measurements, the GP regression model can be used to not only predict which set ($\widehat{\Theta}_{sat}$ or $\widehat{\Theta}_{fail}$) arbitrary vector $\boldsymbol{\theta}_*$ belongs to, but also explicitly quantify the confidence in that prediction. Unlike the artificial non-binary output $H_{\mathbb{R}}$ (3.10) from the SVM model in Chapter 3, the predictive mean $\mu(\boldsymbol{\theta})$ estimates actual non-binary measurements with physical meaning. More importantly, the combination of $\mu(\boldsymbol{\theta})$ and predictive covariance $\Sigma(\boldsymbol{\theta})$ defines a probability distribution around $\mu(\boldsymbol{\theta})$ where the unknown true value $y(\boldsymbol{\theta})$ will lie. This distribution is the probability density function (PDF), the probability $y(\boldsymbol{\theta})$ will fall within a particular range of values. Points with higher covariance $\Sigma(\boldsymbol{\theta})$ will have a wider distribution, signifying there is large uncertainty over how close the true $y(\boldsymbol{\theta})$ lies to the prediction $\mu(\boldsymbol{\theta})$. Meanwhile, points with lower $\Sigma(\boldsymbol{\theta})$ will have a tighter distribution and $y(\boldsymbol{\theta})$ is expected to fall much closer to $\mu(\boldsymbol{\theta})$. Effectively, the covariance indicates the confidence in the accuracy of the predictive mean.

Although the covariance $\Sigma(\boldsymbol{\theta})$ constructs confidence intervals about the accuracy of $\mu(\boldsymbol{\theta})$ with respect to unknown true $y(\boldsymbol{\theta})$, it does not explicitly measure the confidence in the binary predictions for $\widehat{\Theta}_{sat}$ and $\widehat{\Theta}_{fail}$. These binary predictions are based upon whether $y(\boldsymbol{\theta}) > 0$ rather than the exact value of $y(\boldsymbol{\theta})$ itself. Therefore, the probability distribution around $\mu(\boldsymbol{\theta})$ is only as important as it pertains to the likelihood $y(\boldsymbol{\theta}) > 0$. From a Bayesian standpoint, the confidence in the predictions for $y(\boldsymbol{\theta})$ is given by the PDF, but the confidence in the binary predictions for $\widehat{\Theta}_{sat}$ and $\widehat{\Theta}_{fail}$ is specified by the cumulative distribution function (CDF). The probability of satisfaction at each parameter vector $\boldsymbol{\theta}$ is computed through the Gaussian CDF,

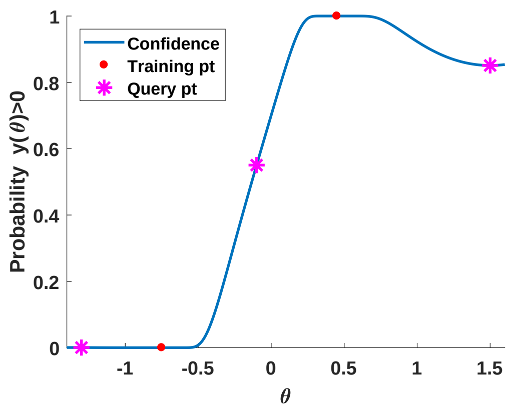
$$\mathbb{P}(y(\boldsymbol{\theta}) > 0|\mathcal{L}, \psi) = \frac{1}{2} + \frac{1}{2}\text{erf}\Big(\frac{\mu(\boldsymbol{\theta})}{\sqrt{2\Sigma(\boldsymbol{\theta})}}\Big). \tag{4.9}$$

Thus the prediction confidence $\mathbb{P}(y(\boldsymbol{\theta}) > 0)$ is a function of both the predictive mean and covariance. Perfect confidence is signified by $\mathbb{P}(y(\boldsymbol{\theta}) > 0) = 1$ or 0, meaning there is 100% probability $y(\boldsymbol{\theta}) > 0$ or 100% probability it is not. Training points $\boldsymbol{\theta} \in \mathcal{D}$ will have perfect confidence as the exact value of $y(\boldsymbol{\theta})$ is known at those points $(\mu(\boldsymbol{\theta}) = y(\boldsymbol{\theta}))$.

The intuition behind the use of the CDF to indicate binary prediction confidence rather than the PDF is illustrated in Figure 4-2. In this example, a Gaussian process regression model has been fit to a set of training points (red dots). Figure 4-2(a) pictures the GP's predictive mean $\mu(\boldsymbol{\theta})$ and the 95% confidence interval formed by the covariance $\Sigma(\boldsymbol{\theta})$. In particular, consider the output at three different query locations $\boldsymbol{\theta}_1 = -1.3$, $\boldsymbol{\theta}_2 = -0.1$, and $\boldsymbol{\theta}_3 = 1.5$ (shown as magenta stars). The covariances at the first two locations are similar, $\Sigma(\boldsymbol{\theta}_1) \approx \Sigma(\boldsymbol{\theta}_2)$, but the third location has a noticeably larger distribution with $\Sigma(\boldsymbol{\theta}_3) > \Sigma(\boldsymbol{\theta}_2)$. While covariance $\Sigma(\boldsymbol{\theta}_3)$ is larger than at the other two locations, it does not necessarily translate into a worse confidence in the binary predictions. The corresponding binary prediction confidence $\mathbb{P}(y(\boldsymbol{\theta}) > 0)$ is shown in Figure 4-2(b). Here, the two training points have perfect confidence because the true value of $y(\boldsymbol{\theta})$ is known. The three query points have unequal prediction confidence based upon the CDF (4.9). Although $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ have similar covariance levels, the fact that $\mu(\boldsymbol{\theta}_2)$ is closer to 0 means the binary prediction is highly sensitive to uncertainty over $y(\boldsymbol{\theta}_2)$ and the confidence is low ($\mathbb{P}(y(\boldsymbol{\theta}_2) > 0) = 0.5$). Likewise, even though $\Sigma(\boldsymbol{\theta}_3)$ is larger than $\Sigma(\boldsymbol{\theta}_2)$, the predictive mean's distance from 0 ($\mu(\boldsymbol{\theta}_3) = 0.7$) reduces the sensitivity to this large covariance. In an exaggerated example, it is possible to have high confidence in the binary predictions even with large covariance, provided $|\mu(\boldsymbol{\theta})| \gg 0$. The analysis of these query points highlights



(a) GP output



(b) Prediction confidence

Figure 4-2: Illustration of GP predictions and their confidence. The GP prediction mean $\mu(\boldsymbol{\theta})$ and 1σ covariance $\Sigma(\boldsymbol{\theta})$ bounds (blue) are shown against the true measurements $y(\boldsymbol{\theta})$ (black). Observed training points are shown in red. Consider the predictions and corresponding confidence of the three query points (magenta). Despite the significantly larger covariance, the right-most query point ($\boldsymbol{\theta} = 1.5$) has relatively high confidence in $\mathbb{P}(y(\boldsymbol{\theta}) > 0)$.

the fact prediction confidence is a function of both mean $\mu(\boldsymbol{\theta})$ and covariance $\Sigma(\boldsymbol{\theta})$.

It is important to note that the prediction confidence (4.9) is also a function of the training dataset $\mathcal{L}$ and the hyperparameters ψ. Changes to either one of these two will affect the resulting mean and covariance. Just as with the true posterior distribution for $y(\boldsymbol{\theta})$ (2.25), the prediction confidence should be marginalized over the predictive distribution for the hyperparameters $\mathbb{P}(\psi|\mathcal{L})$ to find the true confidence. As this is computationally intractable to solve, the prediction confidence is usually approximated as a point estimate with the maximum likelihood estimate for the hyperparameters ψ^*, just as in (4.8).

4.3 Closed-Loop Statistical Verification

Despite the introduction of the GP-based prediction model, statistical verification still suffers from many of the same limitations encountered by SVM-based classification models. For one, the local predictive accuracy of the model is tied to the quality of the training data in the surrounding region of Θ. Ideally, larger portions of the training data would be clustered in regions with high sensitivity, such as near the edge(s) separating Θ_{sat} and Θ_{fail}. Such a distribution would give the Gaussian process increased predictive accuracy in "difficult" areas, although the introduction of the prediction confidence (4.9) does enable the GP to indicate where the prediction confidence is low. Second, statistical verification is usually limited by the amount of resources (time, money, objects, etc.) allocated to the verification process. For a variety of reasons, this limit generally manifests as a cap on the number of trajectories that can be performed, and thus restricts the number of training points.

The overall verification problem is mostly unchanged from Problem 3.3: compute $\widehat{\Theta}_{sat}$ while restricted to N_{lim} number of trajectory samples. The following section presents modified versions of the sequential and batch verification procedures from Section 3.4. At the conclusion of the approaches after N_{lim} simulation or experimental tests have been performed, the GP prediction models will output the two sets $\widehat{\Theta}_{sat}$ and $\widehat{\Theta}_{fail}$, as well as predictive confidence associated with each element of the respective sets. The new predictive confidence can also be used to identify subsets of $\widehat{\Theta}_{sat}$ and $\widehat{\Theta}_{fail}$ with specific confidence levels. For example, it can identify subsets $\widehat{\Theta}_{sat}^{95\%} \subset \widehat{\Theta}_{sat}$ and $\widehat{\Theta}_{fail}^{95\%} \subset \widehat{\Theta}_{fail}$ that only contain $\boldsymbol{\theta}$ locations with at least 95% prediction confidence. Although these types of subsets are useful for estimating misclassification error, the closed-loop procedures ultimately return $\widehat{\Theta}_{sat}$ and $\widehat{\Theta}_{fail}$ because the fundamental problem is to separate Θ into exactly two sets to estimate Θ_{sat} and Θ_{fail}. Closed-loop verification will minimize the binary misclassification error of the two sets for a given sampling budget.

4.3.1 Sample-Selection Criteria

Active learning [117] again forms the basis of the closed-loop verification procedures. Active learning methods are iterative machine learning procedures that actively select the best locations for future training samples. The definition of the "best" sample varies according to the sample-selection criteria, and in turn guides the evolution of the prediction model over time as more and more samples are obtained. First, existing sample-selection metrics taken from relevant work and the approach described in Section 3.4.1 are discussed. While these will certainly work, they are not ideally suited to the regression-based binary verification problem using GPs. Subsequent paragraphs will introduce a new selection metric tailor-made for the verification problem.

Existing Approaches

One possible selection metric is a recycled version of the expected model change (EMC) criteria used for SVM-based closed-loop verification in Section 3.4. This formulation replaces the SVM model output $H_{\mathbb{R}}(\boldsymbol{\theta})$ from (3.16) with the GP posterior predictive mean $\mu(\boldsymbol{\theta})$. The resulting selection criteria ranks points according to their proximity to the $\widehat{y}(\boldsymbol{\theta}) = 0$ prediction surface boundary, i.e.

$$\overline{\boldsymbol{\theta}} = \text{ argmin } |\mu(\boldsymbol{\theta})| \tag{4.10}$$

in a sequential sampling problem. The modified EMC approach is labeled a PDF mean-focused approach since it only considers predictive mean $\mu(\boldsymbol{\theta})$ and neglects prediction covariance $\Sigma(\boldsymbol{\theta})$.

While the EMC criteria does emphasize points near the prediction surface boundary, it ignores the fact the GP output is a distribution rather than a point estimate like in a SVM. An alternative approach is to select samples according to the GP's predictive covariance $\Sigma(\boldsymbol{\theta})$. This is a common technique [92,94,117,125] used to maximize the information gain with each additional training datapoint. After theoretical analysis, the point that maximizes the information gain is simply the point with the largest covariance,

$$\overline{\boldsymbol{\theta}} = \text{ argmax } \Sigma(\boldsymbol{\theta}). \tag{4.11}$$

If the true hyperparameters ψ^* are known, then this approach is submodular as the covariance $\Sigma(\boldsymbol{\theta})$ is independent of actual measurements $y(\boldsymbol{\theta})$ and only requires the sample location $\boldsymbol{\theta}$ itself to be known.

Although the submodularity of variance-based approach (4.11) is desirable, it does not explicitly address the binary verification problem. As depicted in Figure 4-2, large covariance $\Sigma(\boldsymbol{\theta})$ does not automatically equate to large uncertainty in the binary prediction. This can also be seen in the applications addressed by the related work: arrangement of temperature sensors [125], placement of rain sensors [125], and indoor environmental monitoring [94], among others. All these applications are focused on sensing or monitoring without any binary classification needs; their goal is to distribute measurements in order to minimize uncertainty in $\widehat{y}(\boldsymbol{\theta})$, not determine a decision boundary. As discussed in Section 4.2, the binary verification problem instead focuses on the CDF of the measurements in order to measure the confidence in the binary predictions. Therefore, variance-based selection criteria are not ideally suited to this particular active learning application.

Binary Entropy-based Sampling Criteria

To replace the existing selection criteria, an entirely new metric is developed, specifically tailored to exploit binary prediction confidence (4.9). This metric employs binary classification entropy to quantify the uncertainty in the predictions at prospective sample locations. Note that variance-based methods [94, 125] may also use the term "entropy," but that term refers to conditional entropy with respect to mutual information, which is entirely different from binary classification entropy. In subsequent discussions in this thesis, any reference to "entropy" is linked to binary classification entropy. The binary classification entropy $E(\boldsymbol{\theta})$ is determined by the prediction confidence $\mathbb{P}(y(\boldsymbol{\theta}) > 0)$,

$$\begin{aligned} E(\boldsymbol{\theta}|\mathcal{L},\psi) = -\Big(&\mathbb{P}(y(\boldsymbol{\theta}) > 0|\mathcal{L},\psi)\log_2\mathbb{P}(y(\boldsymbol{\theta}) > 0|\mathcal{L},\psi) + \\ &\mathbb{P}(y(\boldsymbol{\theta}) \leq 0|\mathcal{L},\psi)\log_2\mathbb{P}(y(\boldsymbol{\theta}) \leq 0|\mathcal{L},\psi)\Big). \end{aligned} \tag{4.12}$$

The binary classification entropy is pictured as a function of $\mathbb{P}(y(\boldsymbol{\theta}) > 0)$ in Figure 4-3. Since the probability of failure is the opposite of the probability of success, i.e. $\mathbb{P}(y(\boldsymbol{\theta}) \leq 0) = 1 - \mathbb{P}(y(\boldsymbol{\theta}) > 0)$, the entropy $E(\boldsymbol{\theta})$ can be written as a function of just $\mathbb{P}(y(\boldsymbol{\theta}) > 0)$. The entropy is also a function of the chosen hyperparameters ψ and the current training dataset $\mathcal{L}$. The true entropy can be computed by marginalizing over the distribution of possible hyperparameters, but this is computationally intractable and is instead approximated using the MLE hyperparameters ψ^* as in (4.8).

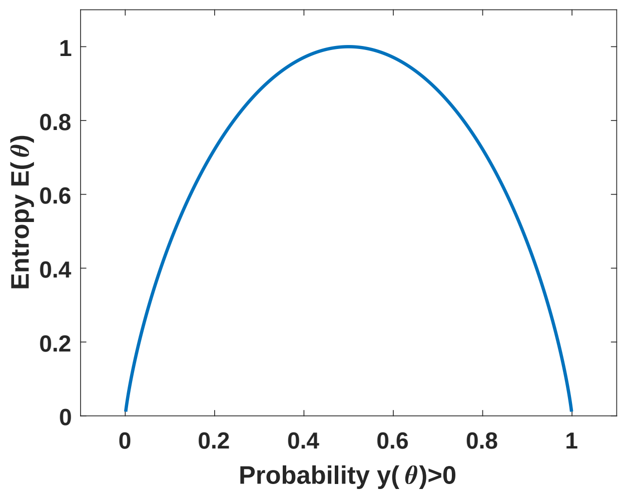


Figure 4-3: Binary classification entropy (4.12) as a function of $\mathbb{P}(y(\boldsymbol{\theta}) > 0)$. Note that $\mathbb{P}(y(\boldsymbol{\theta}) \leq 0) = 1 - \mathbb{P}(y(\boldsymbol{\theta}) > 0)$ and thus the binary classification entropy $E(\boldsymbol{\theta})$ can be written as a function of solely $\mathbb{P}(y(\boldsymbol{\theta}) > 0)$.

In addition to the local entropy at each location $\boldsymbol{\theta}$, the total cumulative entropy over an entire set is also of interest. In particular, it is useful to measure the cumulative entropy over the fine discretization of Θ, set Θ_d from Assumption 3.8. The total entropy is simply the sum of the local entropy over all locations $\boldsymbol{\theta} \in \Theta_d$,

$$E(\Theta_d|\mathcal{L}, \psi) = \sum_{i=1}^{|\Theta_d|} E(\boldsymbol{\theta}_i|\mathcal{L}, \psi). \tag{4.13}$$

The ideal selection metric would minimize the posterior cumulative entropy with the chosen sample,

$$\overline{\boldsymbol{\theta}} = \text{argmin } E\Big(\Theta_d|\mathcal{L}^+, \psi\Big), \tag{4.14}$$

where $\mathcal{L}^+$ is the training set with the additional data $\mathcal{L}^+ = \mathcal{L} \cup \{\overline{\boldsymbol{\theta}}, y(\overline{\boldsymbol{\theta}})\}$. The problem with selection metric (4.14) is the unavailability of the posterior entropy since it requires the measurement $y(\overline{\boldsymbol{\theta}})$ to be known before a simulation or experiment is actually performed there. In its place, the *expected* posterior entropy can be computed using the current expected measurement $\mathbb{E}[y(\boldsymbol{\theta})] = \widehat{y}(\boldsymbol{\theta}) = \mu(\boldsymbol{\theta})$. The minimization of the expected posterior entropy follows the same principle

$$\overline{\boldsymbol{\theta}} = \text{argmin } \widehat{E}\Big(\Theta_d|\widehat{\mathcal{L}}^+, \psi\Big) \tag{4.15}$$

with the artificial training dataset $\widehat{\mathcal{L}}^+ = \mathcal{L} \cup \{\overline{\boldsymbol{\theta}}, \widehat{y}(\overline{\boldsymbol{\theta}})\}$.

While feasible, metric (4.15) is impractical for large datasets because the GP will have to be retrained for each prospective sample location since $\widehat{\mathcal{L}}^+$ changes. Given an artificial training dataset $\widehat{\mathcal{L}}^+$, the expected GP output is

$$\mathbb{E}\big[\mathbb{P}(y(\boldsymbol{\theta}_*)|\widehat{\mathcal{L}}^+, \boldsymbol{\theta}_*, \psi)\big] = \mathcal{N}\big(\widehat{\mu}^+(\boldsymbol{\theta}_*), \Sigma^+(\boldsymbol{\theta}_*)\big), \tag{4.16}$$

which in turn affects the prediction confidence and finally the binary classification entropy. The GP training process involves the inversion of the kernel matrix $\mathbf{K}$, now incremented with additional datapoint $\{\overline{\boldsymbol{\theta}}, y(\overline{\boldsymbol{\theta}})\}$, a $\mathcal{O}(N^3)$ cost *for every prospective sample location under consideration.* This cost quickly becomes intractable for large Θ_d and $\mathcal{U}$, especially for some of the examples considered later in this chapter with grid space Θ_d consisting of up to millions of possible sample locations. Therefore, minimization of the expected posterior entropy (4.15) is possible for small grids, but impractical for many verification problems.

In place of the expected posterior entropy, it is also useful to maximize the posterior decrease in binary classification entropy with each new simulation or experimental test. The rate of decrease in the cumulative posterior entropy is given by

$$\overline{\boldsymbol{\theta}} = \text{argmax } \left(E\Big(\Theta_d|\mathcal{L}, \psi\Big) - \widehat{E}\Big(\Theta_d|\widehat{\mathcal{L}}^+, \psi\Big)\right), \tag{4.17}$$

although this too suffers from the same impracticality as (4.15) because it requires the cumulative expected posterior entropy. However, the the *local* decrease in posterior entropy does not require the repeated $\mathcal{O}(N^3)$ inversion of $\mathbf{K}$. When considering the posterior entropy at the location where a test has been performed, the result is always $E(\boldsymbol{\theta}|\mathcal{L}, \psi) = 0$. This occurs because the covariance $\Sigma(\boldsymbol{\theta}) = 0$ at all the training locations, which would then include the sample location under consideration if a simulation or experiment is indeed performed there. Given this fact, the maximization of the posterior decrease in local entropy reduces to

$$\overline{\boldsymbol{\theta}} = \operatorname{argmax} \left(E\Big(\boldsymbol{\theta}|\mathcal{L}, \psi\Big) - \widehat{E}\Big(\boldsymbol{\theta}|\widehat{\mathcal{L}}^{+}, \psi\Big)\right) \tag{4.18}$$

$$= \operatorname{argmax} \left(E\Big(\boldsymbol{\theta}|\mathcal{L}, \psi\Big) - 0\right) \tag{4.19}$$

$$= \operatorname{argmax}\ E\Big(\boldsymbol{\theta}|\mathcal{L}, \psi\Big). \tag{4.20}$$

Selection metric (4.20) finally presents a sample-selection criteria motivated by both reduction in prediction uncertainty and computational tractability.

In comparison to the existing approaches, (4.20) can be viewed almost as a nonlinear combination of (4.10) and (4.11). Points with low $|\mu(\boldsymbol{\theta})|$ and high covariance $\Sigma(\boldsymbol{\theta})$ are ranked above points with just one of those traits. The closest relevant approach is GP regression for level set optimization [93] which expands upon variance-based methods and GP optimization. The approach places confidence intervals around the mean to predict level sets and bound the information gain with each sample. Unfortunately, it makes a number of restrictive assumptions that limit its utility for a binary verification problem. First, it assumes the true hyperparameters are known and fixed in order to formulate the bound on the information gain which is central to the selection strategy. In most problems starting with zero knowledge, this assumption cannot be made. Later results will show that fixing the hyperparameters to incorrect values will lead to poor performance, even with active learning. Second, the approach's critical bound on information gain is inversely proportional to the measurement noise. The deterministic verification problem has no measurement noise, and even if it is approximated as a narrow-width Gaussian, the bound will be near infinite and of zero use. All of this discussion serves to highlight that binary entropy-based selection criteria are similar to existing selection metrics, but are uniquely tailored to address the deterministic, closed-loop statistical verification problem.

4.3.2 Sequential Sampling

The least-complex closed-loop verification procedure is the sequential approach that selects one sample at a time between retraining steps. Just as with the processes described in Section 3.4, there must first be an initial training dataset $\mathcal{L}$ of passively-selected trajectories before a GP regression model can be constructed. This dataset can be generated using any open-loop, passive procedure such as Latin hypercube [74] or randomly-distributed [73] design of experiments techniques. Once the initial GP model has been obtained, the active selection of samples can begin. These samples are chosen from the available sample set $\mathcal{U}$, which is the remainder of lattice Θ_d after observed training points have been removed: $\mathcal{U} = \Theta_d \setminus \mathcal{D}$. After each iteration, $\mathcal{U}$ is updated to remove the selected training point. The select-test-retrain process is repeated for $T = N_{lim} - N_0$ additional points, where N_0 is the size of the initial training dataset. Algorithm 3 details the complete closed-loop verification procedure, which is summarized in the following paragraph.

Step 1 lists the aforementioned inputs for the closed-loop verification procedure: the initial training dataset $\mathcal{L}$, the available sample set $\mathcal{U}$, and the number of additional samples T. Given this information, the algorithm constructs the initial GP regression model (Step 2). The procedure performs the actual active sampling process in Steps 3-8. In Step 4, the algorithm computes the GP predictive mean and covariance at all $\boldsymbol{\theta} \in \mathcal{U}$ and uses these terms to calculate the entropy at each location in order to rank the points accordingly. The process selects the highest-ranked location $\overline{\boldsymbol{\theta}}$ and performs a simulation or experimental test with those parameter settings to obtain measurement $y(\overline{\boldsymbol{\theta}})$ (Step 5). Once this information has been added to the training dataset $\mathcal{L}$ (Step 6), the retrained GP model in Step 7 incorporates the new observations. This process continues until the number of training points reaches N_{lim}. Once the active sampling process terminates, the procedure returns the final prediction model with the predicted sets $\widehat{\Theta}_{sat}$ and $\widehat{\Theta}_{fail}$ and the corresponding prediction confidence $\mathbb{P}(y(\boldsymbol{\theta}) > 0|\mathcal{L}, \psi)$ for all points in Θ_d (Step 9).

The computational complexity of the GP-based sequential procedure in Algorithm

Algorithm 3 Sequential closed-loop deterministic verification framework using GP regression models

```
1: Input: initial training set $\mathcal{L} = \{\mathcal{D}, \mathbf{y}\}$, available sample locations $\mathcal{U}$, max # of additional samples $T$
2: Initialize: train GP regression model
3: for $i = 1, 2, \ldots, T$ do
4:     Select $\overline{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \mathcal{U}}{\operatorname{argmax}}\ E\big(\boldsymbol{\theta} | \mathcal{L}, \psi\big)$
5:     Perform test at $\overline{\boldsymbol{\theta}}$, obtain measurement $y(\overline{\boldsymbol{\theta}})$
6:     Add $\{\overline{\boldsymbol{\theta}}, y(\overline{\boldsymbol{\theta}})\}$ to training set $\mathcal{L}$, remove $\overline{\boldsymbol{\theta}}$ from $\mathcal{U}$
7:     Retrain model with updated $\mathcal{L}$
8: end for
9: Return: predicted sets $\widehat{\Theta}_{sat}$, $\widehat{\Theta}_{fail}$, and confidence $\mathbb{P}(y(\boldsymbol{\theta}) > 0 | \mathcal{L}, \psi)\ \forall \boldsymbol{\theta} \in \Theta_d$
```

3 is as follows. At the start of the process, the GP regression model training in Step 2 requires the full $\mathcal{O}(N^3)$ complexity to invert the $\mathbf{K}$ matrix. This cost will rise when hyperparameter optimization is performed. Given the precomputed inverse $\mathbf{K}^{-1}$ in Step 2, the $\mathcal{O}(N^2) + \mathcal{O}(N|\mathcal{U}|)$ and $\mathcal{O}(N^2|\mathcal{U}|) + \mathcal{O}(N|\mathcal{U}|)$ costs to compute $\mu(\boldsymbol{\theta})$ and $\Sigma(\boldsymbol{\theta})$ at all $\boldsymbol{\theta} \in \mathcal{U}$ dominates the complexity required to find entropy $E(\boldsymbol{\theta}|\mathcal{L}, \psi)$. Once the process has selected the highest-ranked location $\overline{\boldsymbol{\theta}}$ and performed a test there, the retraining process in Step 7 poses the next major source of computational cost. While this retraining process requires the inversion of the $\mathbf{K}$ matrix, this time with an extra column and row corresponding to kernel evaluations with $\overline{\boldsymbol{\theta}}$, the process can employ a number of techniques to reduce the cost. Using the Woodbury identity for matrix inversion [126], the actual cost to invert the new $\mathbf{K}$ matrix reduces to $\mathcal{O}(N^2) + \mathcal{O}(N)$ operations, thus avoiding the full cubic cost in Step 2. As the process repeats Steps 4-7, the same matrix inversion technique will maintain quadratic complexity for Step 7 by reusing the previous iteration's computations for $\mathbf{K}^{-1}$. Just as in Section 3.4, this complexity analysis ignores the cost to perform an actual simulation or experiment in Step 5 since it will vary from example to example.

Although Algorithm 3 is written using the new binary entropy-based selection metric, the results in Section 4.4 will also examine the performance of closed-loop verification using expected model change (EMC) and variance-based selection criteria. In the PDF variance-based approach, Step 4 is replaced by the corresponding selection metric (4.11) and other changes are needed. The PDF mean-focused/EMC approach uses the same sampling procedure from Section 3.4, but with (4.10) replacing (3.16).

As part of the training and retraining steps, the hyperparameters ψ are also re-optimized with the new training dataset. Therefore, the different versions of the closed-loop procedure using binary entropy, EMC, and variance selection metrics will posses different hyperparameters since each procedure will construct a different $\mathcal{L}$.

4.3.3 Batch Sampling

While Algorithm 3 will select samples as intended, its viability breaks down when applied to large Θ_d. It suffers from the same two limitations as the SVM-based sequential algorithm: 1) the computational cost of retraining the GP after every single additional sample becomes non-negligible and 2) the sequential approach does not exploit parallelism inherent to many applications, particularly simulation-based verification. The selection of samples in batches of M points will help address both of these concerns.

The main challenge with batch sampling is encouraging adequate diversity among the M datapoints. If the M highest-ranked samples are naïvely selected, it will likely choose redundant points in close proximity to one another. Instead, it is generally more efficient to spread the samples out over multiple regions of relatively high ranking. The closed-loop batch framework for SVMs in Algorithm 2 avoided redundancy with the addition of a diversity measure (3.17). This diversity measure penalized possible sample locations in close proximity to locations already chosen for the current batch, thus spacing out the M points along the prediction surface. The following subsection will present multiple alternatives for batch selection with binary entropy-based selection criteria.

Approximate Entropy Reduction

One extension of Algorithm 3 is to artificially update the GP prediction model after each sample selected for the batch. Points previously chosen for the current batch are stored in set $\mathcal{S}$. These stored points are used to create an artificial training dataset $\widehat{\mathcal{L}}^+ = \mathcal{L} \cup \{\mathcal{S}, \widehat{\mathbf{y}}_{\mathcal{S}}\}$ which is then used to retrain the GP model. The resulting approximate GP model outputs predictive mean $\widehat{\mu}(\boldsymbol{\theta})$ and covariance $\widehat{\Sigma}(\boldsymbol{\theta})$ that update the approximate prediction confidence

$$\widehat{\mathbb{P}}(y(\boldsymbol{\theta}) > 0 | \widehat{\mathcal{L}}^+, \psi) = \frac{1}{2} + \frac{1}{2}\mathrm{erf}\Big(\frac{\widehat{\mu}(\boldsymbol{\theta})}{\sqrt{2\widehat{\Sigma}(\boldsymbol{\theta})}}\Big) \tag{4.21}$$

and subsequent binary entropy $\widehat{E}(\boldsymbol{\theta}|\widehat{\mathcal{L}}^+, \psi)$. For each point in the batch after the first, the posterior reduction in entropy is not completely accurate, but rather an approximation since $\widehat{\mathcal{L}}^+$ is not the actual (unknown) training dataset $\mathcal{L}^+$. Algorithm 4 details the complete batch procedure.

As before, the first two steps list the necessary inputs and compute the initial GP regression model. Unlike the sequential approach in Algorithm 3, the batch procedure breaks the remaining $N_{lim} - N_0$ allowable simulations or experiments into T batches with M measurements in each batch, assuming $N_0 + TM = N_{lim}$. Given this initial model, Steps 3-15 contain the batch active sampling process. At the start of each batch, the procedure computes the predictive mean and covariance to find the entropy $E(\boldsymbol{\theta}|\mathcal{L}, \psi)$ for all $\boldsymbol{\theta} \in \mathcal{U}$ (Step 4). The procedure initializes the approximate entropy to this value and begins the batch selection process. The process selects the location with the highest approximate entropy (Step 6) and stores that location in set $\mathcal{S}$ (Step 7). At this point, no simulations or experiments have been performed yet, so the procedure constructs the artificial training dataset $\widehat{\mathcal{L}}^+$ with the predictive mean $\widehat{y}(\boldsymbol{\theta}) = \mu(\boldsymbol{\theta})$ taking the place of an actual measurement (Step 8). The process uses the artificial training dataset to compute an approximate GP model (Step 9) in order to estimate the effects of a measurement at location $\overline{\boldsymbol{\theta}}$ upon the entropy (Step 10) when selecting subsequent locations. Steps 6-10 repeat until the batch set $\mathcal{S}$ has been filled with M locations. Once the batch selection process has chosen all M locations, the algorithm performs tests at those $\boldsymbol{\theta}$ settings and obtains the actual measurements $\mathbf{y}_{\mathcal{S}}$ (Step 12). Steps 13 and 14 add this information to the actual training dataset $\mathcal{L}$ and update the GP regression model. The algorithm will reinitialize the batch process and

Algorithm 4 Batch closed-loop deterministic verification framework using approximate entropy reduction

```
1:  Input: initial training set $\mathcal{L} = \{\mathcal{D}, \mathbf{y}\}$, available sample locations $\mathcal{U}$, # of iterations $T$, batch size $M$
2:  Initialize: train GP regression model
3:  for $i = 1, 2, \ldots, T$ do
4:      Initialize: $\mathcal{S} = \emptyset$, approximate entropy $\widehat{E}(\boldsymbol{\theta}|\widehat{\mathcal{L}}^{+}, \psi) = E(\boldsymbol{\theta}|\mathcal{L}, \psi)$
5:      for $k = 1, 2, \ldots, M$ do
6:          Select $\overline{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \mathcal{U}}{\text{argmax}}\ \widehat{E}\big(\boldsymbol{\theta}|\widehat{\mathcal{L}}^{+}, \psi\big)$
7:          Add $\overline{\boldsymbol{\theta}}$ to set $\mathcal{S}$, remove $\overline{\boldsymbol{\theta}}$ from $\mathcal{U}$
8:          Construct artificial training set $\widehat{\mathcal{L}}^{+} = \mathcal{L} \cup \{\mathcal{S}, \widehat{\mathbf{y}}_{\mathcal{S}}\}$
9:          Train approximate GP model with artificial $\widehat{\mathcal{L}}^{+}$
10:         Recompute approximate entropy $\widehat{E}\big(\boldsymbol{\theta}|\widehat{\mathcal{L}}^{+}, \psi\big)$
11:     end for
12:     Perform tests $\forall \boldsymbol{\theta} \in \mathcal{S}$, obtain measurements $\mathbf{y}_{\mathcal{S}}$
13:     Add $\{\mathcal{S}, \mathbf{y}_{\mathcal{S}}\}$ to training set $\mathcal{L}$
14:     Retrain model with updated $\mathcal{L}$
15: end for
16: Return: predicted sets $\widehat{\Theta}_{sat}$, $\widehat{\Theta}_{fail}$, and confidence $\mathbb{P}(y(\boldsymbol{\theta}) > 0|\mathcal{L}, \psi)\ \forall \boldsymbol{\theta} \in \Theta_d$
```

continue until it has completed T batches. After the iterative process is complete, the procedure returns the predicted sets $\widehat{\Theta}_{sat}$ and $\widehat{\Theta}_{fail}$ and the corresponding prediction confidence $\mathbb{P}(y(\boldsymbol{\theta}) > 0|\mathcal{L}, \psi)$ for all points in Θ_d (Step 16).

There are two main issues with the procedure listed in Algorithm 4. First, the approximate entropy $\widehat{E}\big(\boldsymbol{\theta}|\widehat{\mathcal{L}}^{+}, \psi\big)$ diverges with increasing M. Since the true measurements $\mathbf{y}_{\mathcal{S}}$ are unknown during the selection of the batch, the approximate GP is forced to rely upon artificial measurements $\widehat{\mathbf{y}}_{\mathcal{S}}$ in order to approximate the effects of those sample locations upon the ranking of future potential sample locations. As the number of samples in $\mathcal{S}$ grows, cumulative error between what would have been the true mean $\mu(\boldsymbol{\theta})$ with the actual measurements and the approximate mean $\widehat{\mu}(\boldsymbol{\theta})$ with artificial measurements grows. This discrepancy will then cause the error between the approximate confidence and entropy and their true values to grow as well. Interestingly enough, the approximate covariance $\widehat{\Sigma}(\boldsymbol{\theta})$ is actually the true covariance since covariance only requires the location $\boldsymbol{\theta}$ and is independent of the actual measurement $y(\boldsymbol{\theta})$. This same fact is the root of the submodularity of the PDF variance-based methods.

The second issue with the approximate entropy procedure is the high computational load. Since Step 9 retrains the approximate GP after every new point in $\mathcal{S}$, the computational complexity matches the complexity required for the sequential procedure in Algorithm 3. This lack of an improvement over Algorithm 3 completely contradicts one of the two reasons for batch sampling in the first place. The combination of these two issues limits the practicality of Algorithm 4.

Importance-Weighted Random Sampling

Rather than rely upon expensive approximations of future effects in order to encourage diversity in batch set $\mathcal{S}$, importance-weighted random sampling can be used to efficiently select samples. The crux of this approach is to form a probability distribution from the current binary entropy

$$\mathbb{P}_E(\boldsymbol{\theta}) = \frac{1}{Z_E} E(\boldsymbol{\theta}|\mathcal{L}, \psi), \tag{4.22}$$

where $Z_E = \sum_{i=1}^{|\Theta_d|} E(\boldsymbol{\theta}_i|\mathcal{L}, \psi)$, and randomly select samples from this distribution. This approach is essentially importance sampling Monte Carlo estimation [59,127,128] using binary classification entropy. Regions with high entropy will have a larger probability of selection than areas with low entropy. Although this does not completely eliminate the possibility of redundant points in $\mathcal{S}$, the randomized sampling will generally lead samples to be distributed across all regions of high probability (entropy). Assuming Θ_d and $\mathcal{U}$ are very large, the entropy will be high in many regions near the prediction boundary and it is not likely that all samples are clumped into a single nearby region.

Algorithm 5 depicts the importance-weighting batch procedure. This procedure begins almost exactly the same as Algorithm 4; however, the importance-weighting approach only utilizes the current entropy and does not require an approximate GP model to be constructed. In Step 5, the process converts the entropy into probability distribution $\mathbb{P}_E(\boldsymbol{\theta})$. Next, M sample locations are randomly chosen without replace-

Algorithm 5 Batch closed-loop deterministic verification framework using importance-weighted random sampling

```
1: Input: initial training set $\mathcal{L} = \{\mathcal{D}, \mathbf{y}\}$, available sample locations $\mathcal{U}$, # of iterations $T$, batch size $M$
2: Initialize: train GP regression model
3: for $i = 1, 2, \ldots, T$ do
4:     Initialize: $\mathcal{S} = \emptyset$
5:     Transform $E(\boldsymbol{\theta}|\mathcal{L}, \psi)$ into probability distribution $P_E(\boldsymbol{\theta})$
6:     Generate $M$ random samples from $P_E(\boldsymbol{\theta})$, add to $\mathcal{S}$
7:     Perform tests $\forall \boldsymbol{\theta} \in \mathcal{S}$, obtain measurements $\mathbf{y}_{\mathcal{S}}$
8:     Add $\{\mathcal{S}, \mathbf{y}_{\mathcal{S}}\}$ to training set $\mathcal{L}$
9:     Retrain model with updated $\mathcal{L}$
10: end for
11: Return: predicted sets $\widehat{\Theta}_{sat}$, $\widehat{\Theta}_{fail}$, and confidence $\mathbb{P}(y(\boldsymbol{\theta}) > 0|\mathcal{L}, \psi)$ $\forall \boldsymbol{\theta} \in \Theta_d$
```

ment from $\mathcal{U}$ according to this probability distribution (Step 6). Unlike the previous algorithm, all M points are randomly generated in one step, which yields computational savings compared to the previous sequential process within each batch of M points. The procedure then performs simulations or experiments at the selected $\boldsymbol{\theta}$ settings (Step 7) and adds the resulting measurements to the training dataset (Step 8). Finally, the procedure retrains the GP regression model with the M additional datapoints (Step 9). The iterative process repeats for T batches until the sample budget has been exhausted. Once it reaches this termination point, the algorithm outputs predicted sets $\widehat{\Theta}_{sat}$ and $\widehat{\Theta}_{fail}$ and confidence $\mathbb{P}(y(\boldsymbol{\theta}) > 0|\mathcal{L}, \psi)$ (Step 11).

In terms of computational complexity, Algorithm 5 offers noticeable improvement over Algorithms 3 and 4. The importance-weighted procedure requires the same costs to train the GP regression model in Step 2 and compute the entropy $E(\boldsymbol{\theta}|\mathcal{L}, \psi)$ for Step 5. Unlike Algorithm 4, the new procedure transforms the entropy into $\mathbb{P}_E(\boldsymbol{\theta})$, which requires at least two more $\mathcal{O}(|\mathcal{U}|)$ operations. Given this probability distribution, importance-weighted random sampling without replacement (Step 6) will require $\mathcal{O}(M \log|\mathcal{U}|) + \mathcal{O}(M)$ operations in the best-case scenario [129]. The same Woodbury matrix inversion identity [126] will help lower the complexity for Step 9 by reusing the previous iteration's computations for $\mathbf{K}^{-1}$, allowing the inversion to only require on the order of $\mathcal{O}(M^3) + \mathcal{O}(N^2 M) + \mathcal{O}(M^2 N) + \mathcal{O}(N^2)$ operations for the batch update. Figure 4-4 compares the computational complexity of Algorithm 5 against Algorithms 3 and 4 in Example 4.4.1 and demonstrates the importance-weighted procedure's improvements over those approaches.

Determinantal Point Process-based Random Sampling

The main drawback with importance-weighted random sampling is ensuring set $\mathcal{S}$ does not contain any redundant points. Particularly, when the batch size M is low, it is desirable to spread samples out across regions with similar levels of high probability/entropy. Importance weighting will generally distribute the points across regions of high probability, but it is likely some of the points will be in close proximity and will be redundant. In order to address this redundancy issue while still maintaining the computational feasibility of importance-weighting, the procedure in Algorithm 5 can be augmented with random matrix theory methods.

The main tool to encourage diversity is determinantal point processes (DPP). A more detailed discussion of DPPs is found in the seminal work [130, 131], but an overview of the approach is given in Appendix D. In short, DPPs take a large number (M_T) of $\boldsymbol{\theta}$ locations randomly drawn according to $\mathbb{P}_E(\boldsymbol{\theta})$ and construct a matrix that measures correlation between the samples and penalizes similarities among the datapoints. The correlation matrix's eigenvalues and eigenvectors can be used to generate a second random set of datapoints distributed in regions of high probability, but with increased spatial dispersion within those regions. For active learning purposes, the set of chosen samples $\mathcal{S}$ is this second set with the modified dispersion. In particular, this work uses a special form of determinantal point processes called a k-DPP that is optimized to generate small subsets of M points given a larger initial set of $M_T \geq 1000$ points from $\mathbb{P}_E(\boldsymbol{\theta})$.

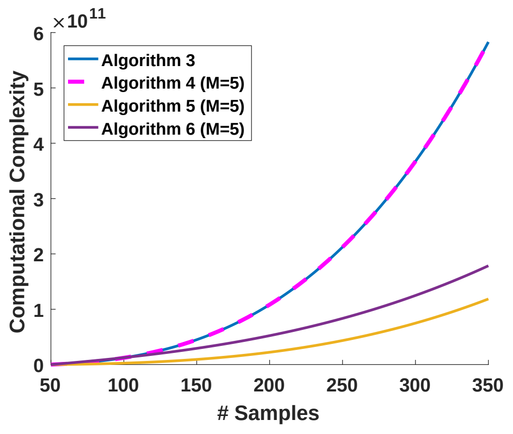
The k-DPP process is inserted into the batch sampling framework in Algorithm 6. The only changes from Algorithm 5 are in Steps 6 and 7 where the k-DPP is constructed and the M samples are randomly generated according to the k-DPP. The main difference is that the k-DPP does require some additional computational overhead in comparison to the baseline importance weighting approach. In Step 6, the

Algorithm 6 Batch closed-loop deterministic verification framework using determinantal point processes

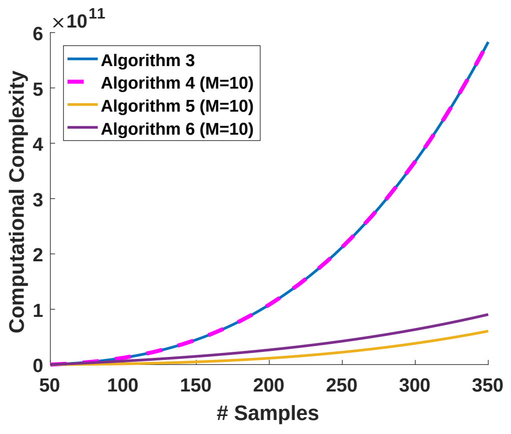
```
1: Input: initial training set $\mathcal{L} = \{\mathcal{D}, \mathbf{y}\}$, available sample locations $\mathcal{U}$, # of iterations $T$, batch size $M$
2: Initialize: train GP regression model
3: for $i = 1, 2, \ldots, T$ do
4:     Initialize: $\mathcal{S} = \emptyset$
5:     Transform $E(\boldsymbol{\theta}|\mathcal{L}, \sigma)$ into probability distribution $P_E(\boldsymbol{\theta})$
6:     Generate $M_T$ random samples from $P_E(\boldsymbol{\theta})$, construct k-DPP
7:     Generate $M$ random samples according to k-DPP, add to $\mathcal{S}$
8:     Perform tests $\forall \boldsymbol{\theta} \in \mathcal{S}$, obtain measurements $\mathbf{y}_{\mathcal{S}}$
9:     Add $\{\mathcal{S}, \mathbf{y}_{\mathcal{S}}\}$ to training set $\mathcal{L}$
10:    Retrain model with updated $\mathcal{L}$
11: end for
12: Return: predicted sets $\widehat{\Theta}_{sat}$, $\widehat{\Theta}_{fail}$, and confidence $\mathbb{P}(y(\boldsymbol{\theta}) > 0|\mathcal{L}, \psi)$ $\forall \boldsymbol{\theta} \in \Theta_d$
```

k-DPP steps requires many more samples ($M_T \gg M$) from $\mathbb{P}_E(\boldsymbol{\theta})$ to first construct the k-DPP, as well as the additional operations with the correlation matrix's eigenvalues and eigenvectors. In total, the cost of constructing and sampling from the k-DPP requires an additional $\mathcal{O}(M_T^3) + \mathcal{O}(M_T M^3)$ operations per batch. Although this cost is higher than Algorithm 5, it is still faster than the original batch procedure in Algorithm 4 for small M and $M_T \ll |\mathcal{U}|$. Figure 4-4 illustrates both the slight increase in complexity over Algorithm 5 and the large improvement over Algorithms 3 and 4. Just as with the baseline importance-weighted approach, this improvement will increase as batch size M increases due to the decreased number of retraining steps necessary for the same number of measurements.

A comparison of all three batch sampling strategies is shown in Figure 4-5. Each algorithm begins with the same initial GP model and training dataset and selects a batch of $M = 10$ samples. Although Figure 4-5(b) does distribute points across the full space, it fails to adequately disperse some of those samples. In the upper-left corner, there are 4 samples in close proximity that likely contain redundant information. By comparison, the other two figures disperse all 10 points across high entropy regions with little overlap. This illustrates that k-DPP sampling is a viable alternative to Algorithm 4 with lower computational overhead and greater diversity than baseline importance-weighted random sampling.



(a) Sequential vs. batch (M = 5) procedures

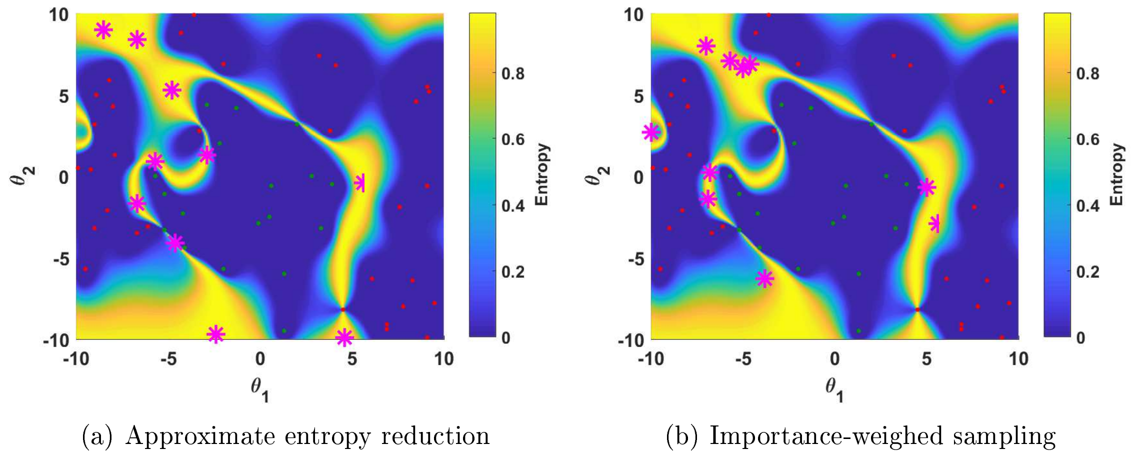


(b) Sequential vs. batch (M = 10) procedures

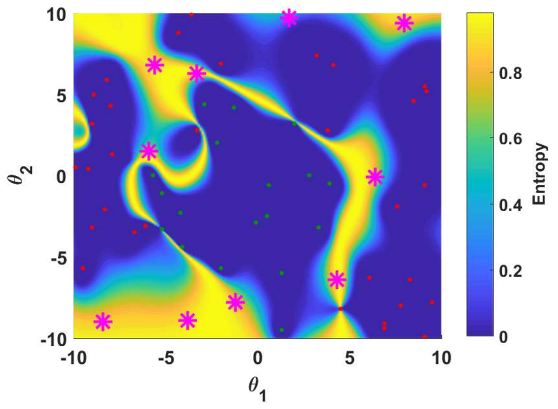
Figure 4-4: Computational complexity of the sequential (Algorithm 3) and batch (Algorithms 4-6) closed-loop verification procedures when applied to Example 4.4.1. Note that Algorithm 4 offers no improvement in complexity over Algorithm 3, while the efficiency of the other two improves at M increases. Since the k-DPP requires additional operations, Algorithm 6 possesses a slightly higher complexity than the baseline importance-weighted approach in Algorithm 5.

4.4 Simulation Results

The sequential and batch closed-loop verification algorithms are demonstrated on numerous examples representative of the wide class of systems of interest. The first example is the same 2D CL-MRAC verification problem from Section 3.5.2. This example compares regression-based binary verification against SVM-based classification methods from Chapter 3. The second example examines verification in a robust multi-agent task allocation problem. This type of system is beyond the scope of the previously-discussed analytical verification techniques. This fact highlights the new statistical verification frameworks' wider applicability when compared to traditional analytical approaches. The later two examples demonstrate closed-loop statistical verification on more complex, higher-dimensional systems. The results in these two examples illustrate Algorithm 6's clear improvement in prediction error over existing open- and closed-loop methods.



(a) Approximate entropy reduction (b) Importance-weighed sampling



(c) k-DPP sampling

Figure 4-5: Selection of samples according to approximate entropy reduction (Algorithm 4), baseline importance-weighted random sampling (Algorithm 5), and k-DPP sampling (Algorithm 6). The latter strategy distributes samples in regions of high entropy, but with less redundancy than Algorithm 5 and a lower computational cost than Algorithm 4.

4.4.1 Concurrent Learning Model Reference Adaptive Controller

The first example is the same concurrent learning model reference adaptive control (CL-MRAC) system previously examined in Section 3.5.2. The system is corrupted by two sources of uncertainty, parameters $\boldsymbol{\theta} = [\theta_1, \theta_2]^T$, which are estimated online with the CL-MRAC adaptive law. While the open-loop state dynamics listed in (3.20) are linear, the adaptive control scheme adds a large amount of complexity and causes the closed-loop dynamics to become nonlinear. This complexity prevents analytical verification techniques from producing useful estimates for $\widehat{\Theta}_{sat}$ and highlights the necessity of statistical verification methods, displayed in Figure 3-11(a).

The performance requirement is also unchanged. The actual state $x_1(t)$ must remain within 1 position unit of the reference state $x_{m_1}(t)$ at every point along the 40 second trajectory,

$$\varphi_{bound} = \Box_{[0,40]} \left(1 - |e_1[t]| \geq 0\right) \tag{4.23}$$

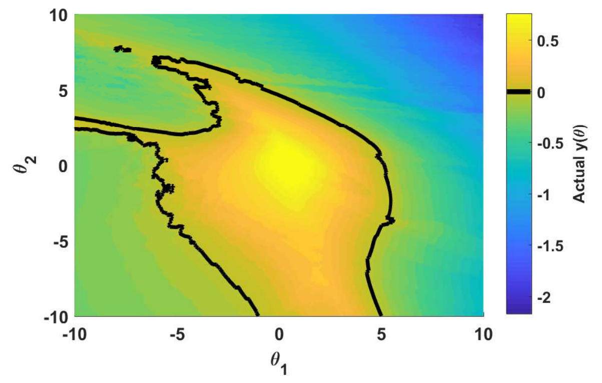
where $e_1(t) = x_{m_1}(t) - x_1(t)$ is the tracking error. Unlike the previous chapter, continuously-valued performance evaluations are assumed to be available in place of binary measurements. The signal temporal logic robustness degree $\rho^{\varphi_{bound}}[t](\boldsymbol{\theta})$ indicates whether the trajectory satisfies φ_{bound} at time t. The complete trajectory's robustness degree is the scalar term

$$\rho^{\varphi}(\boldsymbol{\theta}) = \min_{t' \in [0,40]} \rho^{\varphi_{bound}}[t'](\boldsymbol{\theta}). \tag{4.24}$$

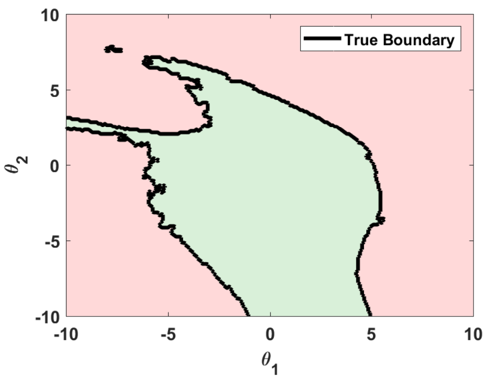
for the minimum level of robustness along the entire trajectory. This robustness degree is passed to the Gaussian process regression model as measurement $y(\boldsymbol{\theta}) = \rho^{\varphi}(\boldsymbol{\theta})$. Parameter vectors $\boldsymbol{\theta}$ with $y(\boldsymbol{\theta}) > 0$ indicate the resulting trajectory successfully met requirement φ_{bound}, while parameters with $y(\boldsymbol{\theta}) \leq 0$ resulted in unsatisfactory performance. The robustness degree also quantifies the level of robustness (or lack thereof) demonstrated by the trajectory. In this example, if $y(\boldsymbol{\theta}) \leq 0$ then the value of $y(\boldsymbol{\theta})$ measures how far $x_1(t)$ deviated from $x_{m_1}(t)$ at the worst point in the trajectory. When $y(\boldsymbol{\theta}) > 0$, the value indicates just how close the trajectory was to failure at its least-robust point.

Figure 4-6(a) displays the true robustness degree measurement $y(\boldsymbol{\theta})$. The black line indicates the separation boundary between Θ_{sat} and Θ_{fail} at $y(\boldsymbol{\theta}) = 0$. These two sets are shown in Figure 4-6(b). This binary shape is the same exact plot previously seen in Figure 3-11(a). By definition, the sets $\Theta_{sat}, \Theta_{fail}$ produced by both binary MTL and non-binary STL are equivalent.

The sampling grid Θ_d is a lattice of 40,401 points that cover the space between $-10 \leq \theta_1 \leq 10$ and $-10 \leq \theta_2 \leq 10$. From this grid, a training set $\mathcal{L}$ of 50 randomly



(a) Surface of $y(\boldsymbol{\theta})$

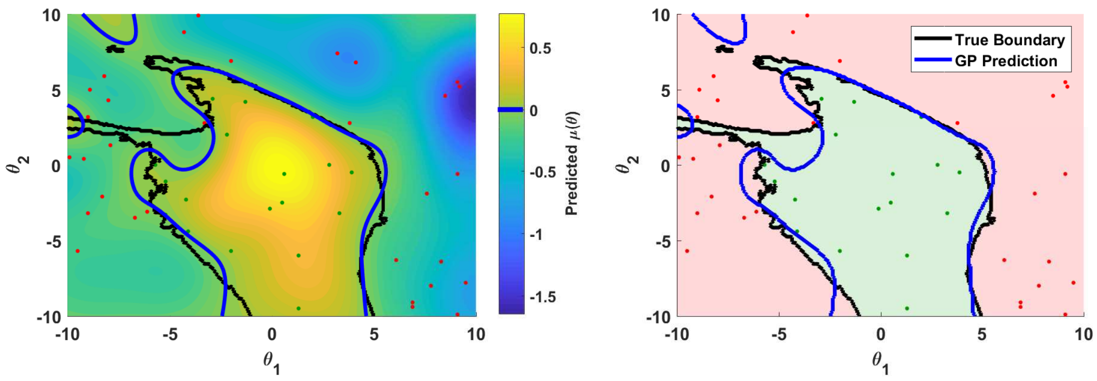


(b) True $\Theta_{sat}/\Theta_{fail}$

Figure 4-6: [Example 4.4.1] Plot of the actual $y(\boldsymbol{\theta})$ surface from the STL robustness degree over Θ. The right-hand plot displays the Θ_{sat} (green) and Θ_{fail} (red) sets resulting from whether $y(\boldsymbol{\theta}) > 0$ or $y(\boldsymbol{\theta}) \leq 0$.

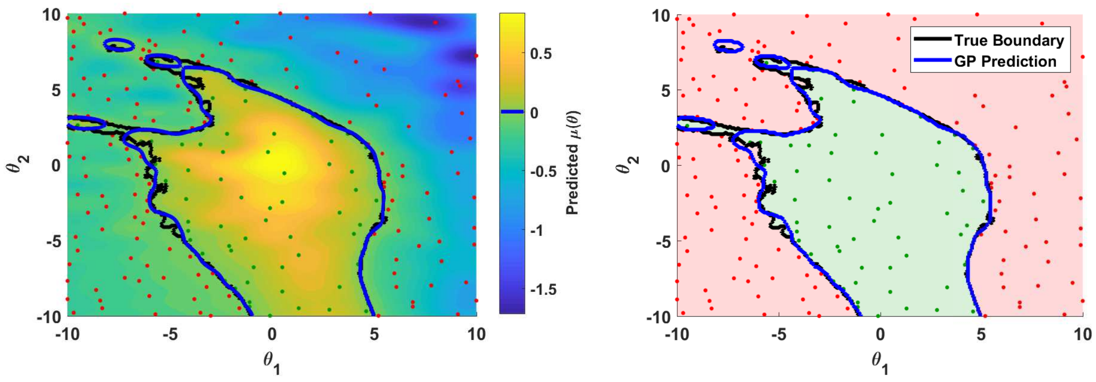
chosen trajectories and their measurements is used to train an initial GP regression model. This model is the starting condition for the closed-loop verification procedures. Figure 4-7 shows an example of the GP regression model trained on this small initial training set of 50 points. Given the sparsity of the training dataset, the GP regression model will only have limited information on which to base its predictions. As a result, the predicted boundary separating the two sets (blue line) is a rough estimate of the true boundary (black line) and obviously needs more data to correctly converge towards the true shape pictured in Figure 4-6. The closed-loop sampling approach will speed up this convergence by distributing samples into informative regions of Θ_d.

Two versions of Algorithm 6 are examined, one with batch size $M = 5$ and one with $M = 10$. Figure 4-5(c) already displayed the selection of the first batch of $M = 10$ points using the the same model from Figure 4-7 as the starting point. These entropy-based algorithms are compared against closed-loop verification methods using the PDF variance-based selection metric (4.11) and the modified expected model change (EMC) metric applied to Gaussian processes (4.10). The closed-loop approaches are also evaluated against open-loop verification methods. Due to the similar performance between Latin hypercube and uniform-random DOE approaches in Section 3.5, only the random sampling DOE procedure will be used in subsequent comparisons. All of these approaches will operate until a sampling budget of $N_{lim} = 350$ trajectories



(a) Surface of $\widehat{y}(\boldsymbol{\theta})$ (b) Predicted $\widehat{\Theta}_{sat}/\widehat{\Theta}_{fail}$

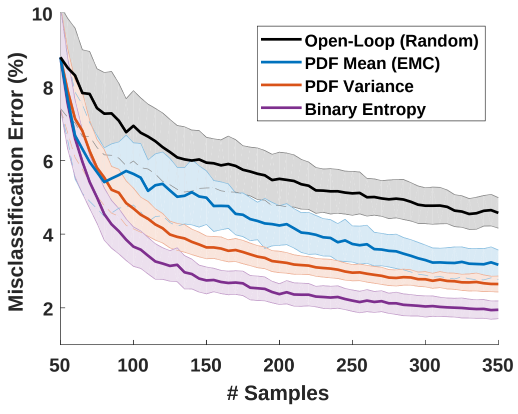
Figure 4-7: [Example 4.4.1] Initial prediction model after an initial training dataset of 50 randomly-selected parameter settings for the trajectories. Parameter settings $\boldsymbol{\theta} = [\theta_1, \theta_2]$ for the simulations are shown as dots and their respective colors denote whether the resulting trajectory satisfied the requirement (green) or did not (red). The figure illustrates the prediction boundary (blue line) that separates $\widehat{\Theta}_{sat}$ and $\widehat{\Theta}_{fail}$ against the true boundary (black line) separating Θ_{sat} and Θ_{fail} (shown as green/red regions in the right-hand plot). As a result of the small training dataset, the GP model has limited information on which to base its predictions, leading to the areas with noticeable misclassification errors.



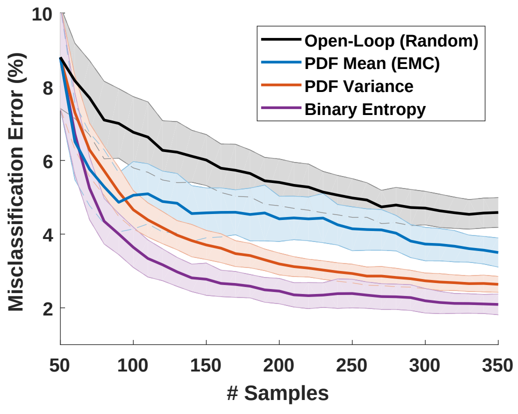
(a) Surface of $\widehat{y}(\boldsymbol{\theta})$ (b) Predicted $\widehat{\Theta}_{sat}/\widehat{\Theta}_{fail}$

Figure 4-8: [Example 4.4.1] Final prediction model after 350 samples. The resulting prediction boundary separating $\widehat{\Theta}_{sat}$ and $\widehat{\Theta}_{fail}$ is a much better approximation of the true boundary than in Figure 4-7, although it is still an imperfect representation.

has been reached. Figure 4-8 pictures the same problem from Figure 4-7 after the completion of the closed-loop verification process with all 350 samples. The final estimate is a much more accurate representation of the true shape, although it still misclassifies a few areas along the boundary.



(a) Batch size $M = 5$



(b) Batch size $M = 10$

Figure 4-9: [Example 4.4.1] Misclassification error convergence of Algorithm 6 in comparison to the other approaches over the same 100 random initializations. For ease of viewing, the standard deviation intervals around the mean (solid lines) are given by the 0.5σ bound rather than the traditional 1σ bound.

Figure 4-9 illustrates the performance of the open- and closed-loop procedures starting from the same 100 random initializations. In both the $M = 5$ and $M = 10$ versions of Algorithm 6, the entropy-based selection metric outperforms closed-loop verification using the other metrics as well as the generic open-loop approach. At the conclusion of the 350 samples, the entropy-based closed-loop algorithms demonstrate a 20%($M = 10$) and 26% ($M = 5$) improvement in average prediction error over the PDF variance-based approach. Both of these approaches have similar standard deviation levels and are better in both mean and standard deviation than the EMC-based and open-loop procedures.

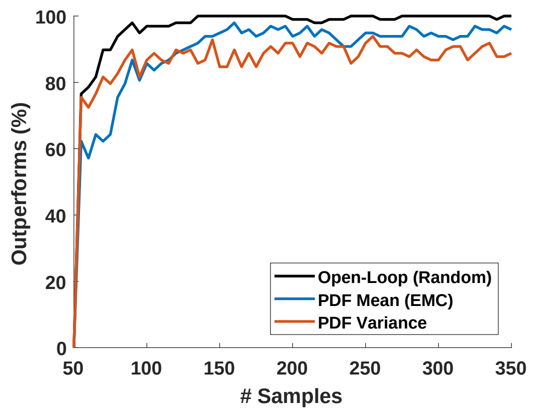
The results in Figure 4-9 are also directly compared to the results for Algorithm 2 from Figure 3-11. After 250 samples (what was used in Figure 3-11), the new entropy-based algorithm using continuous measurements has a 17% improvement in average misclassification error over the EMC approach in Algorithm 2 using only binary measurements. Interestingly enough, the variance-based approach has roughly the same average error while the modified EMC algorithm actually does 43% *worse* than the original version (Algorithm 2) using only binary measurements. This quick comparison confirms that regression-based closed-loop verification using binary classification entropy is preferable to the classification-based verification procedures in Chapter 3. Since the misclassification error rate for Algorithm 6 beats Algorithm 2, the regression-based closed-loop framework also outperforms the analytical barrier certificates.

While the results in Figure 4-9 demonstrated Algorithm 6's improvement in average misclassification error over the other three procedures, the results did not directly indicate whether Algorithm 6 is consistently better, only that it has a lower average error. However, since the approaches all start with the same 100 random initializations, the approaches can be directly compared against one another for each of the 100 runs. Figure 4-10 displays the percentage of these 100 runs where Algorithm 6 has either a lower or matching level of misclassification error to the indicated approaches. By the completion of the closed-loop verification process, Algorithm 6 outperforms or matches the PDF variance-based method in 89% of the runs for both batch sizes. Likewise, Algorithm 6 outperforms the modified EMC and open-loop techniques in almost 95% and 100% of the test cases. These results couple with Figure 4-9 to highlight the improved rate of misclassification errors produced by the binary entropy-based sampling algorithms. Given a limited sampling budget, Algorithm 6 will consistently produce the most accurate prediction model.

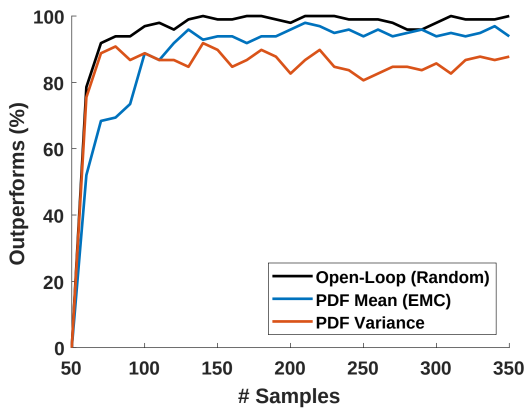
Confidence Levels in the Predictions

The new ability to explicitly measure prediction confidence online without an external validation set can be used for a variety of purposes. The primary use is to provide a local confidence level associated with a queried location $\boldsymbol{\theta} \in \Theta$; however, it is also possible to examine the confidence levels over the entire lattice Θ_d. Since the true misclassification error rate from Figure 4-9 is unknown during actual execution of the procedures, the prediction confidence is an important tool to estimate the total rate of misclassification errors.

First, prediction confidence is employed to identify subsets of $\widehat{\Theta}_{sat}$ and $\widehat{\Theta}_{fail}$ with certain levels of confidence. For example, consider the initial GP model from Figure 4-7. All the binary predictions have a certain level of confidence which can be used to



(a) Batch size $M = 5$



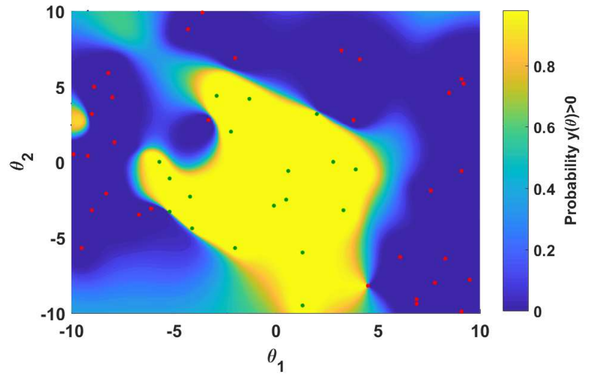
(b) Batch size $M = 10$

Figure 4-10: [Example 4.4.1] Ratio of runs where Algorithm 6 directly outperforms or matches the misclassification error rate of the indicated approaches. All strategies start with the same initial training set and thus each approach can be directly compared to the others with the same initialization.

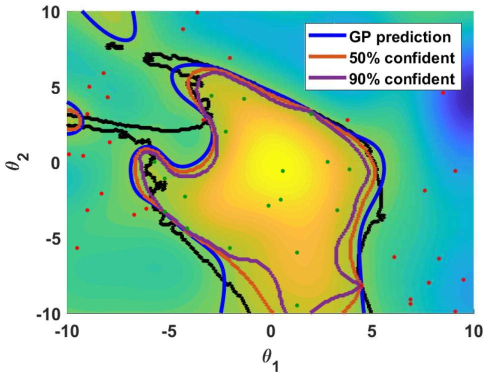
further segment Θ_d and identify regions where misclassifications are likely to occur. Figure 4-11 illustrates that the prediction confidence for $\mathbb{P}(y(\boldsymbol{\theta}) > 0)$ segments $\widehat{\Theta}_{sat}$ into tighter subsets with a minimum confidence level. At the 90% confidence level, the new subset $\widehat{\Theta}_{sat}^{90\%}$ is more conservative than the initial prediction, but removes many (but not all) of the misclassifications. The prediction confidence for $\mathbb{P}(y(\boldsymbol{\theta}) \leq 0)$ segments $\widehat{\Theta}_{fail}$ in a similar fashion with $\widehat{\Theta}_{fail}^{90\%}$. The remaining points not in $\widehat{\Theta}_{sat}^{90\%}$ or $\widehat{\Theta}_{fail}^{90\%}$ have a high likelihood of misclassification.

The $\widehat{\Theta}_{sat}^{50\%}$, $\widehat{\Theta}_{sat}^{90\%}$, etc. subsets are analogous to the $C_{FP} = \{1, 3, 6\}$ predicted sets in Figure 3-13 that penalized false-positive errors with constraint C_{FP}. Unlike the previous results, the confidence levels for each subset $\widehat{\Theta}_{sat}^{50\%}$, $\widehat{\Theta}_{sat}^{90\%}$, etc. now have an explicit meaning whereas the C_{FP} term was merely a penalization on the training process that had no quantifiable measure of impact. More importantly, these subsets do not require retraining of the model and are produced simultaneously. In order to produce the different sets in Figure 3-13, the SVM had to be retrained for each value of C_{FP}. For GP-based verification, the GP only has to be trained once and the prediction confidence will segment $\widehat{\Theta}_{sat}$ and $\widehat{\Theta}_{fail}$ without retraining.

Additionally, the prediction confidence is independent of the chosen sampling met-

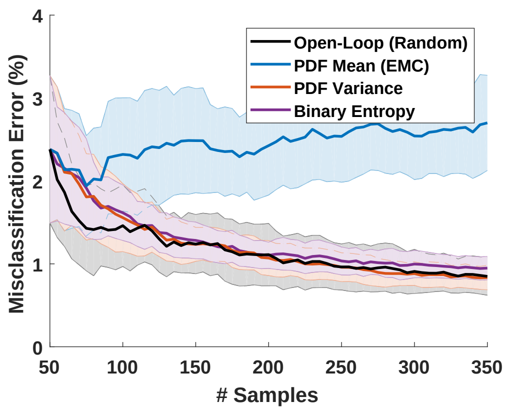


(a) Prediction confidence $\mathbb{P}(y(\boldsymbol{\theta}) > 0)$

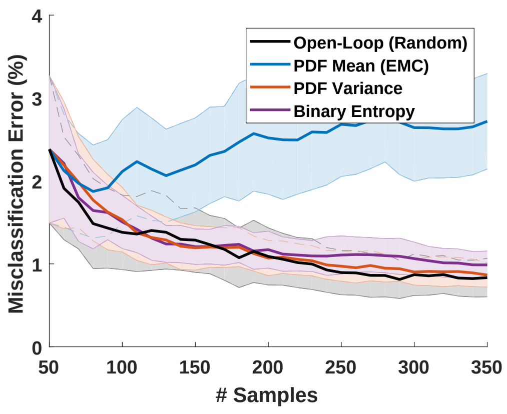


(b) Confidence levels for $\widehat{\Theta}_{sat}$

Figure 4-11: [Example 4.4.1] Illustration of confidence levels in the predictions of $\widehat{\Theta}_{sat}$. Note: the corresponding levels in $\widehat{\Theta}_{fail}$ are not shown.



(a) Batch size $M = 5$



(b) Batch size $M = 10$

Figure 4-12: [Example 4.4.1] Rate of misclassification error in the 95% prediction confidence level. This percentage is out of the number of points classified with 95% prediction confidence rather than the total set Θ_d. For ease of viewing, the standard deviation intervals also correspond to 0.5σ bounds.

ric or whether open- or closed-loop statistical verification is employed. Figure 4-12 shows the misclassification error rate corresponding to the 95% confidence intervals of $\widehat{\Theta}_{sat}$ and $\widehat{\Theta}_{fail}$ for the various approaches. Regardless of the chosen procedure, the confidence levels have a much lower ratio of misclassification errors compared to the full $\widehat{\Theta}_{sat}, \widehat{\Theta}_{fail}$ even during the initial stages of the process. The EMC metric is a slight outlier, but the overall rate (<5%) still makes sense within the 95% confidence level.

Importance of Hyperparameter Optimization

As was discussed in Section 4.2, the true hyperparameters are not assumed to be known in advance and maximum likelihood estimation optimizes the hyperparameters online. Especially during the initial steps of the closed-loop process, the distribution of points within $\mathcal{L}$ will change drastically and the hyperparameters will likely vary accordingly. While it may seem advantageous to simply fix the hyperparameters due to the computational savings of avoiding hyperparameter optimization, naïvely fixing the hyperparameters can lead to poor performance.

Consider the comparison of the three batch ($M = 10$) closed-loop procedures in Figure 4-13. In the static (fixed) cases with dashed lines, the hyperparameters are optimized at the initial training step, but are not updated any further. Meanwhile, the other three solid lines show the average misclassification error when the hyperparameters are updated after each step, as was done with all the results shown up to this point. If the hyperparameters are fixed, the misclassification error actually increases between steps, even though there are more samples and these samples were actively chosen according to the indicated selection metric. This complete breakdown in prediction accuracy highlights the dangers of naïvely fixing the hyperparameters when the true values are unknown and demonstrates the limitations of similar procedures [93] based upon the assumption of fixed hyperparameters.

4.4.2 Robust Multi-Agent Task Allocation

The second example is a robust multi-agent task allocation problem. This task allocation problem is very different than the other examples considered in Chapters 3 and 4 and demonstrates data-driven statistical verification can be applied to a very broad range of systems with little-to-no modification. Here, the verification goal is to test whether a multi-agent system can successfully complete an ordered list of tasks while subject to parametric disturbances.

In particular, this problem considers aerial forest firefighting using UAVs in the presence of uncertain wind conditions [32]. Two sets of 2 UAVs of heterogeneous

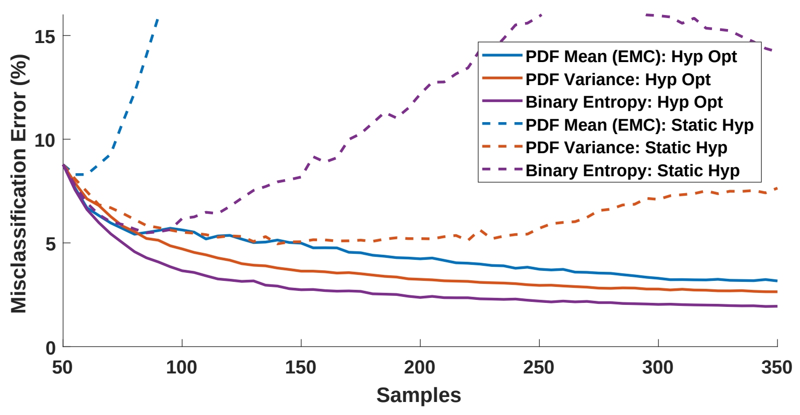


Figure 4-13: [Example 4.4.1] Average misclassification error of the closed-loop verification procedures with hyperparameter optimization (solid lines) versus the same procedures with static hyperparameters (dashed lines). Both sets start with the same initial training set and model, but the static versions do not update the hyperparameters during the retraining process.

capabilities are used to complete 30 surveillance tasks scattered throughout the world map. These surveillance tasks correspond to areas of possible forest fires that the UAVs must investigate. If a UAV spots a fire, it must map the perimeter of the fire at the location to provide the human fire commander with updates on the size, rate, and direction of fire expansion. The challenge with these fire surveillance tasks is that the tasks may take significantly longer to complete when the UAV maps the fire perimeter than if the UAV did not detect a fire and moved on to the next task in the list. Fire perimeter mapping is also a function of wind parameters (wind speed and direction) since the winds will affect the burn rate and spread the fire into different areas with varying terrain and vegetation combustibility. More details on the aerial forest firefighting problem and task allocation are found in Appendix B.

For the sake of simplicity, it is assumed the fixed control policy, the ordered list of tasks assigned to each UAV, has already been generated by a multi-agent task allocation procedure. This work uses the robust consensus-based bundle algorithm (CBBA) to produce the control policy [26, 132]. The realized score of the control policy is a function of wind speed and direction. As the firefighting tasks take shorter or longer to complete depending upon the wind conditions, tasks at the end of the ordered list could be performed significantly behind schedule and potentially not at all if the UAVs run out of fuel and have to return to base. Each task also has a time-decaying reward, so severely-delayed tasks will produce a lower score than originally intended. The cumulative reward of the given control policy will vary with the two wind parameters.

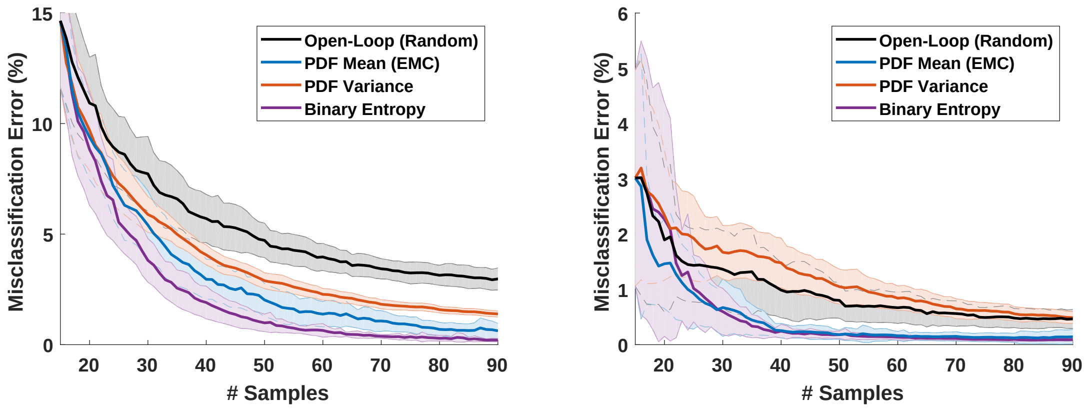
The verification goal of the process is to determine whether the control policy at a given set of wind conditions will reach a minimum score threshold. In order to test this with the statistical verification framework, a lattice of 16,641 points covers the feasible space between $\theta_1 : [0°, 359°]$ and $\theta_2 : [0, 40]$ (km/hr). Unlike the preceding MRAC example, this example utilizes the sequential form of the closed-loop procedure (Algorithm 3) and the competing open- and closed-loop approaches.

Figure 4-14 demonstrates the rate of misclassification error for the various verification procedures over 100 random initializations. The closed-loop framework with the entropy-based selection criteria outperforms the competing algorithms, depicted in Figures 4-14(a) and 4-15. Likewise, Figure 4-14(b) examines the error rate within the 95% confidence levels for $\widehat{\Theta}_{sat}$ and $\widehat{\Theta}_{fail}$. Just as in the last example, the error rate within the 95% confidence levels is significantly lower than in the full Θ_d. The results further highlight the utility of the prediction confidence to identify regions with high likelihood of misclassification errors.

4.4.3 Adaptive Control with Complex Temporal Specifications

The third example is an adaptive control system with complex spatial-temporal performance requirements. The example uses the same closed-loop state dynamics from Sections 3.5.2 and 4.4.1, but with the more complex requirements from Section 3.5.3. The requirement φ states the system must eventually satisfy all three of the following specifications ($\varphi = \varphi_1 \wedge \varphi_2 \wedge \varphi_3$) in order for a trajectory to be considered satisfactory. These three specifications are

$$\begin{aligned}
&\varphi_1 = \Diamond_{[2,3]}\ (x_1[t] - 0.7 \geq 0) \wedge \Diamond_{[2,3]}\ (1.3 - x_1[t] \geq 0),\\
&\varphi_2 = \Diamond_{[12,13]}\ (x_1[t] - 1.1 \geq 0) \wedge \Diamond_{[2,3]}\ (1.7 - x_1[t] \geq 0), \qquad (4.25)\\
\text{and}\quad &\varphi_3 = \Box_{[22.4,22.6]}\ (x_1[t] + 1.6 \geq 0) \wedge \Box_{[22.4,22.6]}\ (-1.2 - x_1[t] \geq 0).
\end{aligned}$$



(a) Total misclassification error

(b) Misclassification error in the 95% prediction confidence level

Figure 4-14: [Example 4.4.2] Rate of misclassification errors for Algorithm 3 and the competing approaches. The left plot shows the total misclassification error across all of Θ_d while the right-hand figure shows the error of the 95% confidence level. The standard deviation intervals correspond to 0.5σ bounds.

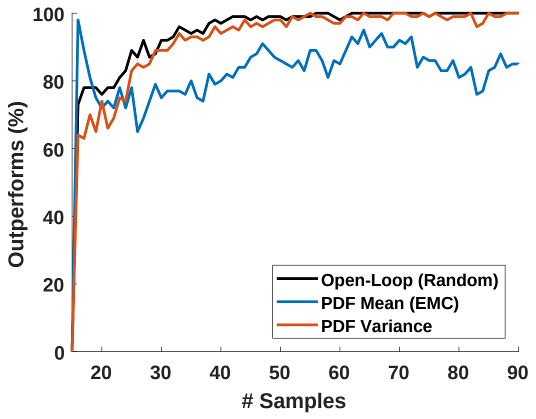


Figure 4-15: [Example 4.4.2] Ratio of runs where Algorithm 3 directly outperforms or matches the misclassification error rate of the indicated approaches. All strategies start with the same initial training set and thus each approach can be directly compared to the others with the same initialization.

The STL robustness degree for each trajectory is the minimum robustness degree encountered during the trajectory,

$$\rho^{\varphi}(\boldsymbol{\theta}) = \min \{\rho^{\varphi_1}(\boldsymbol{\theta}),\ \rho^{\varphi_2}(\boldsymbol{\theta}),\ \rho^{\varphi_3}(\boldsymbol{\theta})\}, \tag{4.26}$$

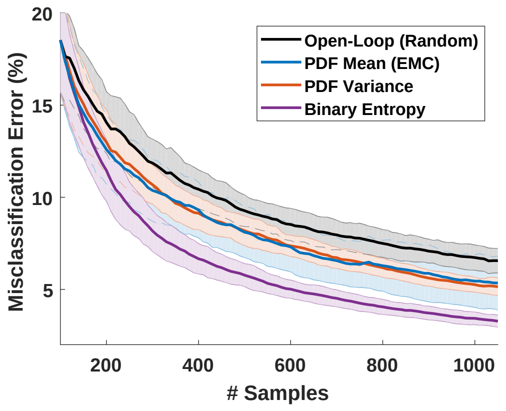
where

$$\rho^{\varphi_1}(\boldsymbol{\theta}) = \max_{t' \in [2,3]} \rho^{\varphi_1}[t'](\boldsymbol{\theta}), \qquad \rho^{\varphi_2}(\boldsymbol{\theta}) = \max_{t' \in [12,13]} \rho^{\varphi_2}[t'](\boldsymbol{\theta}),$$
$$\text{and } \rho^{\varphi_3}(\boldsymbol{\theta}) = \min_{t' \in [22.4,22.5]} \rho^{\varphi_3}[t'](\boldsymbol{\theta}). \tag{4.27}$$

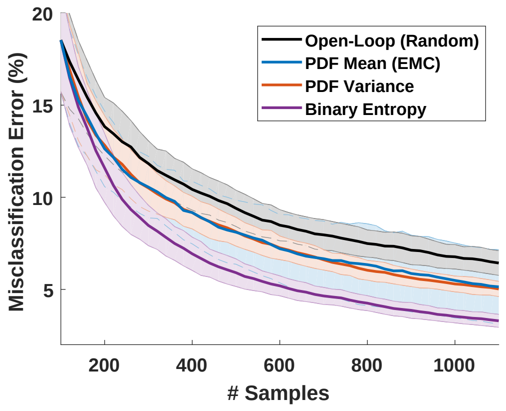
The continuous measurement $y(\boldsymbol{\theta})$ for each trajectory is the total STL robustness degree, $y(\boldsymbol{\theta}) = \rho^{\varphi}(\boldsymbol{\theta})$.

This example considers the same two uncertain parameters (θ_1, θ_2) from Section 4.4.1, but adds a third source of uncertainty θ_3 to capture uncertainty in the initial state $x_1(0)$. The statistical verification frameworks select trajectory conditions from a grid Θ_d of 214,221 possible sampling locations that cover $\theta_1 : [-5, 5]$, $\theta_2 : [-5, 5]$ and $\theta_3 : [-1, 1]$. Each of the open- and closed-loop verification procedures start with an initial training dataset of 100 trajectories at randomly-selected parameter settings. The algorithms operate in batch sizes of $M = 10$ and $M = 20$ up to a sampling budget of $N_{lim} = 1,050/1,060$ training points.

The total misclassification error produced by the statistical verification procedures is shown in Figure 4-16. For both batch sizes, the entropy- and variance-based closed-loop procedures have comparable levels of standard deviation, but the entropy-based approaches demonstrate a 35% improvement in average prediction error over PDF variance methods and a 33% improvement over the modified EMC algorithm. When the four approaches are directly compared against each other in Figure 4-17, Algorithm 6 consistently outperforms or matches the other approaches in the vast majority of the 100 randomly-initialized runs. At the completion of the verification procedure, Algorithm 6 has the lowest misclassification error rate in at least 95% of the $M = 10$ cases and 85% of the $M = 20$ cases. In contrast to Figure 4-10 from Example 4.4.1, the modified EMC algorithm takes the place as the nearest competitor to the entropy-based sampling approaches. This switch from the PDF variance-based approach to the EMC-based approach as the "next-best" approach highlights the inconsistent performance of those procedures. The EMC and PDF variance-based procedures will perform better in some problems and worse in others, while the binary classification entropy procedures are consistently the best. Ultimately, these results reinforce

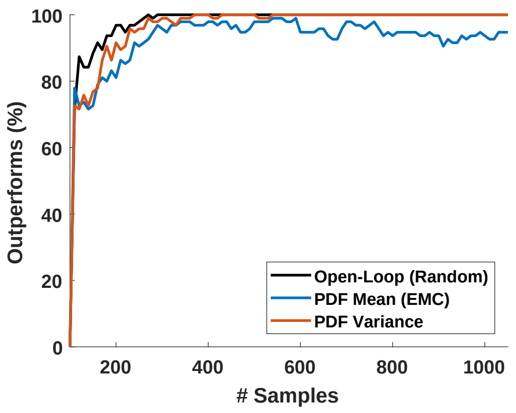


(a) Batch size $M = 10$

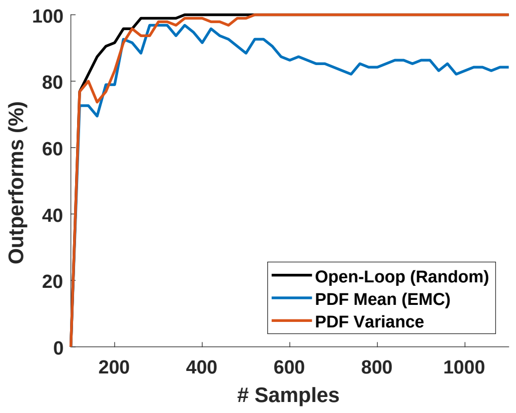


(b) Batch size $M = 20$

Figure 4-16: [Example 4.4.3] Misclassification error convergence of Algorithm 6 in comparison to the other approaches over the same 100 random initializations. The standard deviation intervals around the mean (solid lines) are given in traditional 1σ bounds.



(a) Batch size $M = 10$

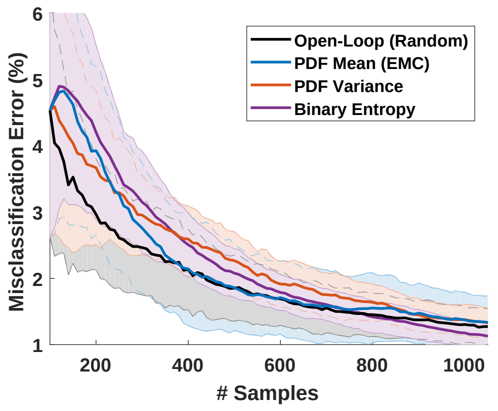


(b) Batch size $M = 20$

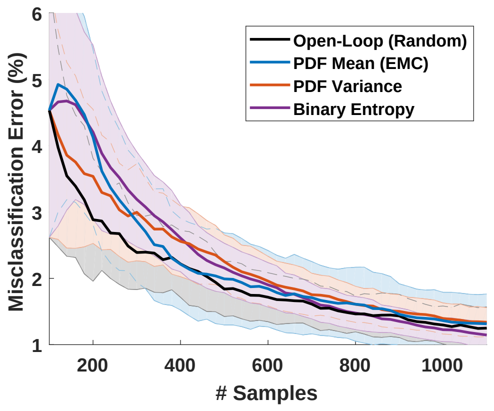
Figure 4-17: [Example 4.4.3] Ratio of runs where Algorithm 6 directly outperforms or matches the misclassification error rate of the indicated approaches. All strategies start with the same initial training set and thus each approach can be directly compared to the others with the same initialization.

the notion that entropy-based closed-loop verification produces the most accurate predictions for $\widehat{\Theta}_{sat}$ and $\widehat{\Theta}_{fail}$.

The results in Figure 4-21 also agree with the observations in the preceding examples. The rate of misclassification error within the 95% confidence level is lower than the rate over the full set, meaning the majority of incorrectly-labeled points possessed low prediction confidence. As was seen in Figure 4-14(b), the rate of misclassification



(a) Batch size $M = 10$



(b) Batch size $M = 20$

Figure 4-18: [Example 4.4.3] Rate of misclassification error in the 95% prediction confidence level. The standard deviation intervals around the mean (solid lines) are given in 1σ bounds.

errors within the 95% confidence interval is roughly the same between the different statistical verification procedures.

4.4.4 Lateral-Directional Autopilot

The last example verifies an aircraft's autopilot for controlling lateral-directional flight modes. In particular, the example examines the "heading-hold" autopilot mode used to turn the aircraft to and maintain a desired reference heading. The closed-loop aircraft dynamics are provided by the DeHavilland Beaver flight simulator in the Aerospace Blockset of Matlab/Simulink [133, 134]. This simulation model is significantly more complex than any of the previous examples as it includes numerous nonlinearities such as the full nonlinear 6 degree-of-freedom aircraft dynamics, nonlinear pitch, roll, and yaw controllers, and actuator models for each control surface with position and rate saturations. The autopilot itself includes various modes and switching logic that may affect the closed-loop response.

The heading-hold autopilot has various requirements that should be satisfied, but the dominating specification was found to be the altitude-hold requirement φ_{height} after a trade-space exploration. This requirement states the aircraft must remain within 35 feet of the initial altitude at every point in the turn [133, 134],

$$\varphi_{height} = \Box_{[0,50]}(35 - |h[t] - h[0]| \geq 0), \tag{4.28}$$

where $h[t]$ is aircraft altitude at time $0 \leq t \leq 50$ seconds. The continuous measurements are given by the STL robustness degree

$$y(\boldsymbol{\theta}) = \rho^{\varphi}(\boldsymbol{\theta}) = \min_{t' \in [0,50]} \rho^{\varphi_{height}}[t'](\boldsymbol{\theta}). \tag{4.29}$$

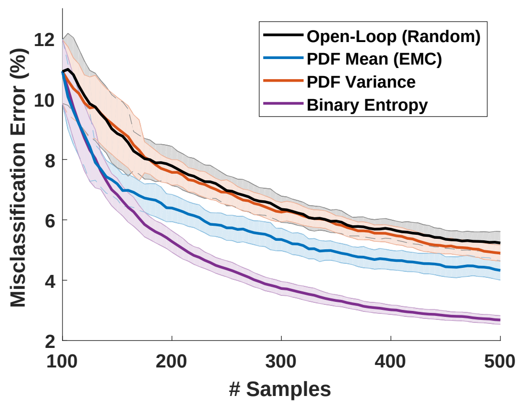
More detailed information about the example is found in Appendix C. The satisfaction of requirement φ_{height} is tested against four uncertain initial conditions: Euler angles for roll (θ_1), pitch (θ_2), and yaw (θ_3), as well as pitching moment of inertia I_{yy} (θ_4). The last parameter, moment of inertia I_{yy}, corresponds to uncertainty in the loading of the aircraft; the same amount of weight could be distributed towards the front or back of the aircraft and would affect the longitudinal dynamics. Interestingly enough, an initial trade space exploration did not indicate any sensitivity to the actual weight, only the moment of inertia. The space of allowable perturbations spans θ_1 : $[-60°, 60°]$, $\theta_2 : [4°, 19°]$, heading angle $\theta_3 : [75°, 145°]$, and $\theta_4 : [5430, 8430](kg \cdot m^2)$ with a desired reference heading angle of 112°. The discrete sampling grid Θ_d consists of a total of 937,692 possible sampling locations.

The results of the various statistical verification approaches over 100 random initializations are shown in Figures 4-19 and 4-20. The approaches start from an initial training dataset of 100 points and operate until a sampling budget of $N_{lim} = 500$ points is reached. The results are consistent with the observations from the preceding examples; the entropy-based algorithm outperforms the competing approaches in average misclassification error rate. For a batch size of $M = 5$, the entropy-based procedure demonstrates a 37% improvement in prediction error over the EMC-based closed-loop framework and an even higher rate over the other two approaches. In the $M = 10$ case, entropy-based sampling again outperforms the EMC-based competitor by 34% and the other two by roughly 44%. However, it is important to note that the results for the EMC algorithm shown in Figure 4-19, the EMC batch diversity term λ was heuristically optimized to minimize the prediction error. If the verification procedures are compared against the results for the original (non-optimized) λ term, the performance of the EMC approaches quickly degrade (not shown) and entropy-based sampling demonstrates a 50% improvement over the sub-optimal EMC algorithm. This discussion merely identifies an extra challenge faced with EMC algorithms, the best choice for λ, that does not affect the other three sampling strategies.

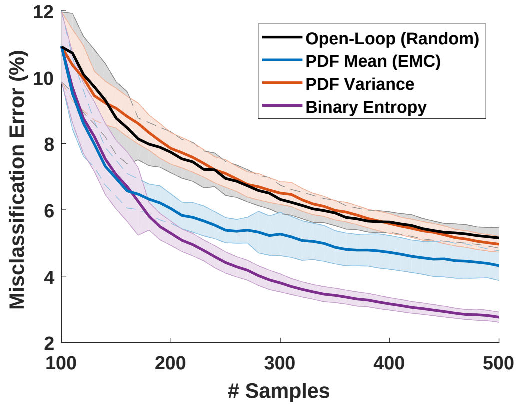
Beyond the mean prediction error rate, Figure 4-20 illustrates Algorithm 6's clear benefit over the other three approaches when directly compared against them in each of the 100 runs. Algorithm 6 has either a better or matching rate of misclassification error upon completion of the verification process for all of the 100 randomly-initialized runs. In contrast to the previous example problems, this autopilot example is a better representation of the complex systems considered in real-world industrial problems. This clear reduction in prediction error shown in Figures 4-19 and 4-20 highlights the large potential benefit of the entropy-based closed-loop statistical verification frameworks for more accurate predictions of robustness in industrial-level problems.

Additionally, the effects of hyperparameter optimization are more clearly visible in this example than in the previous three. In particular, the noticeable jumps in the standard deviation intervals for the binary entropy and EMC procedures from Figure 4-19(b) are due to hyperparameter optimization. Since nothing is known about the hyperparameters in advance, the hyperparameter optimization procedure can only rely upon the training dataset $\mathcal{L}$ to select the hyperparameters. Both of those jumps in total prediction error are caused by a drastic switch in the hyperparameter values which lower (binary entropy) or increase (EMC) the prediction error in the next iteration of the active sampling process. This sensitivity demonstrates the need for effective hyperparameter optimization to help minimize the prediction error.

Lastly, Figure 4-21 plots the misclassification error rate within the set of points of high prediction confidence. As seen in Figure 4-21 and the earlier examples, the ratio of misclassification errors is lower than the full set Θ_d once all the points with low prediction confidence have been removed. This sharp decrease reinforces the idea (4.9) is a useful tool to compute prediction error online without an external, independent validation dataset.

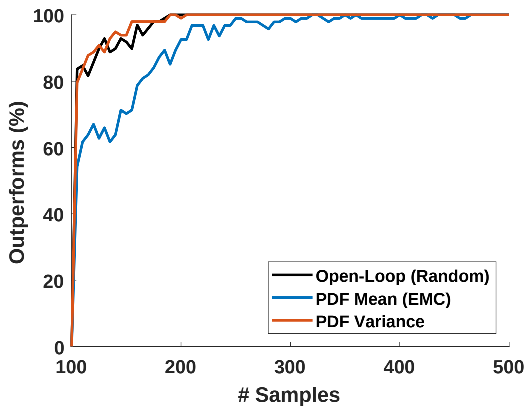


(a) Batch size $M = 5$

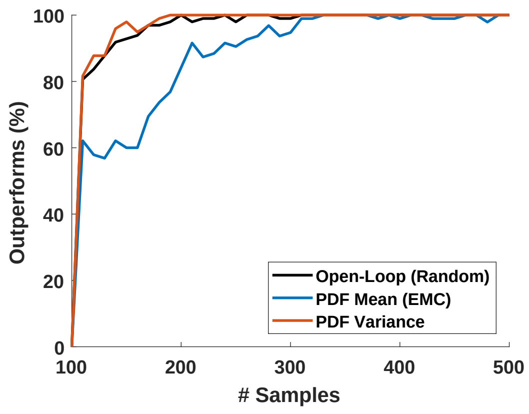


(b) Batch size $M = 10$

Figure 4-19: [Example 4.4.4] Misclassification error convergence of Algorithm 6 in comparison to the other approaches over the same 100 random initializations.. The standard deviation intervals around the mean (solid lines) are given in 0.5σ bounds for easier visualization.



(a) Batch size $M = 5$

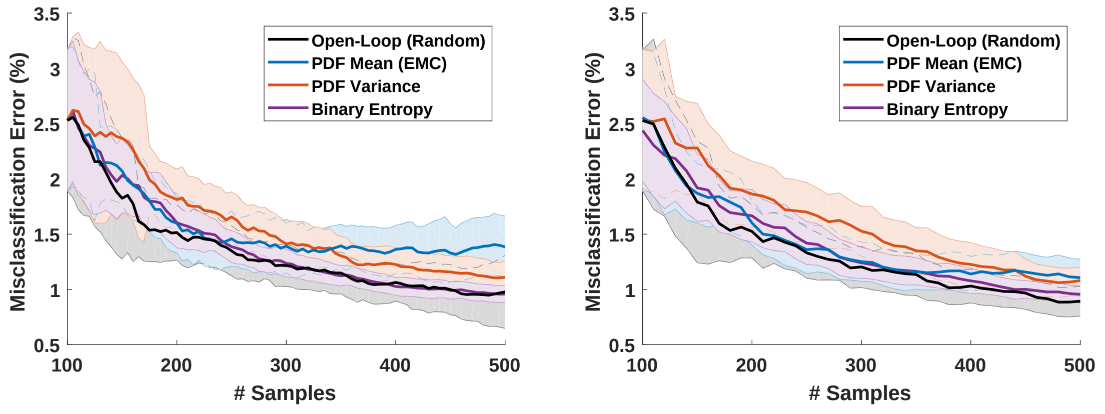


(b) Batch size $M = 10$

Figure 4-20: [Example 4.4.4] Ratio of runs where Algorithm 6 directly outperforms or matches the misclassification error rate of the indicated approaches. All strategies start with the same initial training set and thus each approach can be directly compared to the others with the same initialization.

4.5 Summary

This chapter introduced a substantial redevelopment of data-driven procedures for statistical verification of deterministic nonlinear systems. In comparison to Chapter 3, this chapter assumes non-binary performance measurements are available and constructs a Gaussian process regression model for binary classification/prediction. The



(a) Batch size $M = 10$ (b) Batch size $M = 20$

Figure 4-21: [Example 4.4.4] Rate of misclassification error in the 95% prediction confidence level. The standard deviation intervals around the mean (solid lines) are given in 0.5σ bounds for easier visualization.

main benefit of this new approach is the ability to compute prediction confidence online without relying upon sample-inefficient, external validation sets. Additionally, the new GP-based approach motivates a new set of closed-loop verification procedures that exploit binary classification entropy in order to minimize the prediction error. These procedures are demonstrated on multiple case studies covering a variety of controls applications. In the most complex examples, the proposed GP-based, closed-loop verification frameworks demonstrate a 30-40% reduction in misclassification error over the nearest competing approach.

Chapter 5

Stochastic Verification with Gaussian Distributions of Trajectory Robustness

This chapter details the development of data-driven approaches for statistical verification of stochastic systems. The previous two chapters demonstrated the benefits of data-driven verification for efficient verification of deterministic systems; however, these deterministic techniques fail to address additional challenges introduced by stochastic systems. Chapters 5 and 6 develop data-driven verification procedures specifically designed to combat these additional challenges.

In particular, this chapter extends the previous work in Chapter 4 to address the baseline problem in stochastic systems. Section 5.1 introduces the main challenge associated with verification of stochastic systems: the closed-loop system will no longer deterministically satisfy performance requirements. Stochasticity means there will be a distribution of robustness measurements when multiple trajectories are performed at the same parameter setting. The work in this chapter assumes a Gaussian distribution for these measurements and presents a new Gaussian process regression-based formulation in Section 5.2 to estimate the distribution. The main difference from the earlier work is that the Gaussian process model no longer segments Θ into two sets Θ_{sat} and Θ_{fail}, but rather computes the expected probability of satisfactory performance at every $\boldsymbol{\theta} \in \Theta$. The new formulation also necessitates significant changes to the closed-loop verification frameworks. Section 5.3 presents new sample-selection metrics specifically designed to minimize the prediction error between the true (unknown) distribution and the Gaussian process output. Sections 5.4 and 5.5 both describe extensions to the baseline approaches to include more complicated distributions. As in the previous chapters, the last section demonstrates these algorithms' improvements in prediction error over competing active learning and design of experiments procedures.

5.1 Problem Description

In this work, consider a closed-loop nonlinear system with a similar form to the deterministic system in (3.3),

$$\dot{\mathbf{x}}(t) = f_{cl}(\mathbf{x}(t), \boldsymbol{\theta}, \mathbf{w}(t)), \tag{5.1}$$

with state vector $\mathbf{x}(t) \in \mathbb{R}^n$ and parametric uncertainties $\boldsymbol{\theta} \in \mathbb{R}^p$. These parametric uncertainties are assumed to fall within the known, compact set Θ discussed in Assumption 3.1. Unlike the deterministic system in (3.3), the stochastic system (5.1) is corrupted by stochastic noise $\mathbf{w}(t)$.

The stochastic noise may be the byproduct of many different possible sources, the most common of which are process noise in the open-loop dynamics and measurement noise in the system output used by the control system to generate feedback control inputs. When the system is subject to process noise, the open-loop dynamics are corrupted by random disturbance $\mathbf{v}_q(t)$,

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t), \boldsymbol{\theta}, \mathbf{v}_q(t)). \tag{5.2}$$

Typically, the effects of random events such as wind gusts or other disturbances are modeled as additive Gaussian noise $\mathbf{v}_q(t) \sim \mathcal{N}(\bar{\mathbf{v}}_q, \epsilon_q^2)$ with mean $\bar{\mathbf{v}}_q$ and variance ϵ_q^2. The process noise causes the system to produce a random output $\dot{\mathbf{x}}(t)$ for the same set of inputs $\{\mathbf{x}, \mathbf{u}, \boldsymbol{\theta}\}$ and therefore no two trajectories will be the same, even with identical parameter settings and control inputs.

The second likely source of stochasticity is measurement noise $\mathbf{v}_r(t)$. Particularly in real-world, physical systems, the true state $\mathbf{x}(t) \in \mathbb{R}^n$ is often not directly measured and instead the states are inferred from measured output $\mathbf{z}(t) \in \mathbb{R}^{n_r}$,

$$\mathbf{z}(t) = h(\mathbf{x}(t), \mathbf{v}_r(t)). \tag{5.3}$$

Even in the simplest case where the open-loop dynamics are deterministic ($\mathbf{v}_q(t) = 0$) and $\mathbf{z}(t) = \mathbf{x}(t) + \mathbf{v}_r(t)$, the control system will not have direct access to $\mathbf{x}(t)$ and the only source of feedback is a noisy measurement $\mathbf{z}(t)$. This reliance upon $\mathbf{z}(t)$ introduces randomness into the closed-loop system and the end result is the same as with process noise: no two trajectories will be the same.

Although the examples in this thesis only consider process and measurement noise that enter the open-loop dynamics as additive zero-mean Gaussian distributions, nonlinearities in the closed-loop dynamics will complicate analysis. As a result, even those simple noise terms may appear nonlinearly in the closed-loop dynamics. The noise term $\mathbf{w}(t)$ merely captures the effect of any randomness in the closed-loop dynamics, regardless of its source. Ultimately, stochasticity breaks the underlying argument from the previous two chapters. The closed-loop trajectory $\Phi(\mathbf{x}(t)|\mathbf{x}_0, \boldsymbol{\theta})$ is no longer purely a deterministic function of known initial state $\mathbf{x}_0$ and parameters $\boldsymbol{\theta}$, but now includes a random element in its time evolution. The effect of stochasticity in the trajectories is demonstrated in Figure 5-1. Four trajectories all start from the same initial conditions, but quickly diverge from one another even though they share the same controller. As will be discussed in the next paragraphs, this randomness will have an effect upon the satisfaction of performance requirements at a given a $\mathbf{x}_0$ and $\boldsymbol{\theta}$. This can be seen in Trajectory 2 from the figure, where the state $x(t)$ actually exceeds the bound stipulated by the performance requirement $\varphi = \square_{[0,25]}\ (2.1 - x[t] > 0)$. Even though the other three trajectories all satisfy the requirement, they each demonstrate a varying degree of robustness.

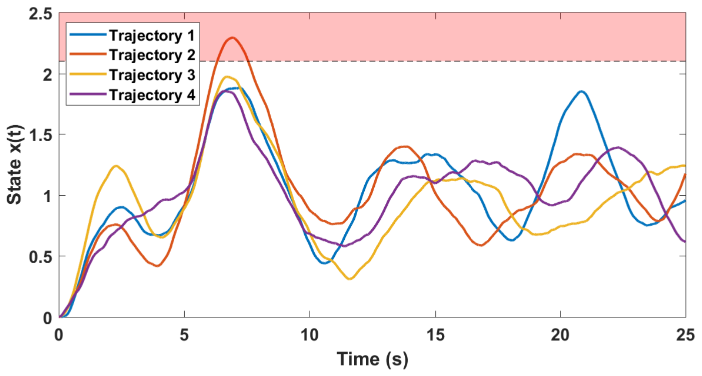


Figure 5-1: Illustration of the effects of stochasticity in the closed-loop response of four trajectories with the same initialization. The red box depicts a performance requirement for an upper bound on the state $x(t)$, $\varphi = \square_{[0,25]}\ (2.1 - x[t] > 0)$.

5.1.1 Distribution of Trajectory Robustness Measurements

The four example trajectories in Figure 5-1 highlight the biggest challenge with verification of stochastic systems: the satisfaction of a performance requirement is no longer deterministic. As in Assumption 4.1, continuous measurements $y(\boldsymbol{\theta}) \in \mathbb{R}$ indicate the robustness of the system's trajectory $\Phi(\mathbf{x}(t)|\mathbf{x}_0, \boldsymbol{\theta})$ to a pre-specified performance requirement.

Assumption 5.1. *There exists an expert or oracle which provides a deterministic, scalar output $y(\boldsymbol{\theta}) \in \mathbb{R}$ that measures a trajectory's minimum robustness to the specified performance requirement. As before in Assumption 4.1, the sign of y indicates satisfaction of the requirement where positive $y > 0$ corresponds to "satisfied" while $y \leq 0$ corresponds to "did not satisfy."*

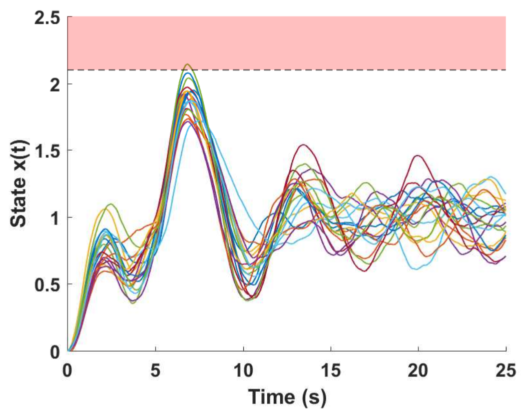
The expert or oracle will return a single robustness measurement for each observed trajectory to indicate that trajectory's satisfaction of the requirement.

Remark 5.2. *Even though the system is stochastic, these measurements are deterministic with respect to the exact trajectory $\Phi(\mathbf{x}(t)|\mathbf{x}_0, \boldsymbol{\theta})$; the expert/oracle will always return the same $y(\boldsymbol{\theta})$ given $\Phi(\mathbf{x}(t)|\mathbf{x}_0, \boldsymbol{\theta})$.*

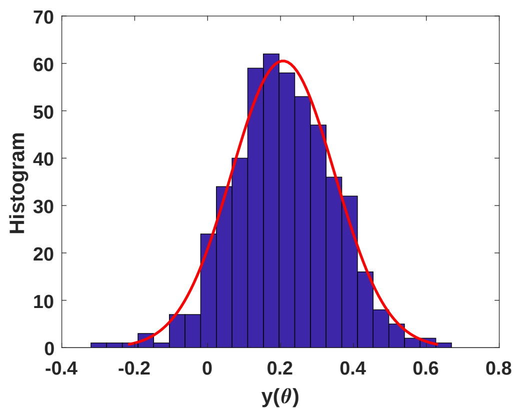
This aspect of Assumption 5.1 prevents any prospective measurement noise from effecting the evaluation of trajectory performance and does slightly limit the applicability of the approach. However, in many applications, the absence of measurement noise on $y(\boldsymbol{\theta})$ is not a restrictive assumption. In simulation environments, the true state $\mathbf{x}(t)$ may be hidden from the control system to replicate real-world conditions, but this state and its effects upon performance requirements would be completely observable to the certification oracle in the simulation model. In real-world laboratory experiments, external sources of information such as motion-capture systems [135] or other special testing apparatuses like air-data probes on flight-test aircraft [136, 137] would typically provide additional state information that may be hidden from the control system under test. Additionally, the satisfaction of the requirements might be completely observable even without perfect state information. For instance, failures to complete certain tasks or collisions with obstacles do not require noiseless state measurements to observe their fairly obvious effects. In situations where the measurements $y(\boldsymbol{\theta})$ are not deterministic with respect to $\Phi(\mathbf{x}(t)|\mathbf{x}_0, \boldsymbol{\theta})$, recent methods like probabilistic STL [105] can be employed to consider probabilistic predicates (requirement specifications), but are not discussed further.

Although $y(\boldsymbol{\theta})$ are deterministic with respect to the exact trajectory, these measurements are no longer deterministic with respect to the operating conditions $\boldsymbol{\theta}$, unlike Chapter 4. This complicates mapping parameters $\boldsymbol{\theta}$ to particular robustness levels since the same initialization could generate a different $y(\boldsymbol{\theta})$ for each repeated trajectory. This is seen in Figure 5-1 where all four of the trajectories produced different STL robustness degrees $\rho^{\varphi}(\boldsymbol{\theta})$. Figure 5-2 displays this same effect on an even greater number of trajectories. Figure 5-2(a) displays the trajectories for 20 repetitions of the same initial condition, each with their own different robustness degree measurement for $y(\boldsymbol{\theta})$. As the number of repetitions grows, the set of corresponding robustness measurements $y(\boldsymbol{\theta})$ empirically constructs a distribution for the likelihood of $y(\boldsymbol{\theta})$ given that same operating condition. In Figure 5-2(b), the histogram for 500 repetitions of the same system clearly illustrates a distribution over possible values for $y(\boldsymbol{\theta})$. In particular, the red line shows this set of 500 repetitions resembles a Gaussian distribution over $y(\boldsymbol{\theta})$.

Just like the histogram in Figure 5-2(b), the approach presented in this chapter assumes the true distribution of $y(\boldsymbol{\theta})$ at each query location is a Gaussian distribution.



(a) 20 trajectories with the same initialization



(b) Histogram of robustness measurements from a total of 500 trajectories

Figure 5-2: Distribution of robustness measurements for stochastic trajectories with the same initialization. Note that the performance requirement in (a) is the same bound from Figure 5-1 and the measurement $y(\boldsymbol{\theta})$ is the corresponding STL robustness degree. The red line in the right-hand figure displays a Gaussian distribution fit to the data.

Assumption 5.3. *The distribution of continuous measurements $y(\boldsymbol{\theta})$ at every $\boldsymbol{\theta} \in \Theta$ is a Gaussian distribution $y(\boldsymbol{\theta}) \sim \mathcal{N}(\bar{y}(\boldsymbol{\theta}), \epsilon_y^2)$ with spatially-varying mean $\bar{y}(\boldsymbol{\theta})$ and constant variance ϵ_y^2.*

This distribution is also known as a *homoscedastic* Gaussian distribution [138] as the variance ϵ_y^2 is independent of $\boldsymbol{\theta}$. In short, Assumption 5.3 effectively states any trajectory performed at arbitrary condition $\boldsymbol{\theta}$ will possess a corresponding robustness measurement $y(\boldsymbol{\theta})$ generated according to distribution $\mathcal{N}(\bar{y}(\boldsymbol{\theta}), \epsilon_y^2)$. Note that this distribution $y(\boldsymbol{\theta}) \sim \mathcal{N}(\bar{y}(\boldsymbol{\theta}), \epsilon_y^2)$ does not have to reflect the same level of noise $\mathbf{w}(t)$ in (5.1). Even if $\mathbf{w}(t)$ is a Gaussian distribution with $\mathbf{w}(t) \sim \mathcal{N}(\bar{\mathbf{w}}, \epsilon_w^2)$, nonlinearities in the closed-loop dynamics could lead to a completely different distribution for $y(\boldsymbol{\theta})$. Frequently, the variance levels may be different ($\epsilon_w \neq \epsilon_y$) due to control saturation or output feedback controllers.

It is important to highlight that Assumption 5.3 does restrict the set of stochastic systems considered in this chapter. Not only is the distribution assumed to be Gaussian, but the variance ϵ_y^2 must also remain constant across all $\boldsymbol{\theta} \in \Theta$. Although this assumption limits the wider applicability of the data-driven verification procedures, the Gaussian distribution still captures all the important aspects of stochastic verification and simplifies the problem for better clarity. Later extensions in Section 5.4 will relax this assumption and allow ϵ_y^2 to vary across Θ and even consider non-Gaussian distributions for $y(\boldsymbol{\theta})$, but ultimately build upon this initial work.

5.1.2 Satisfaction Probability Function

The stochasticity in the trajectories also modifies the verification objective since the disjoint regions of satisfaction and failure in Definitions 4.3 and 4.4 no longer exist. Given two different trajectories at the same $\boldsymbol{\theta} \in \Theta$, one trajectory may satisfy the performance requirement while the second may not. The likelihood an arbitrary simulation or experimental test will satisfy the requirement is defined by a Bernoulli distribution with probability parameter p_{sat}. This probability parameter is a function of $\boldsymbol{\theta}$ and is labeled the *satisfaction probability function*.

Definition 5.4. *The* satisfaction probability function $p_{sat}(\boldsymbol{\theta}) \in [0,1]$ *defines the likelihood an arbitrary simulation or experimental test initialized at* $\boldsymbol{\theta}$ *will satisfy the performance requirement. In this problem with continuous robustness measurements,* $\mathbb{P}(y(\boldsymbol{\theta}) > 0) = p_{sat}(\boldsymbol{\theta})$.

As indicated in Definition 5.4, the satisfaction probability function $p_{sat}(\boldsymbol{\theta})$ is the cumulative distribution of the Gaussian probability density function for robustness measurements $y(\boldsymbol{\theta})$,

$$p_{sat}(\boldsymbol{\theta}) = \mathbb{P}(y(\boldsymbol{\theta}) > 0) = \frac{1}{2} + \frac{1}{2}\,\mathrm{erf}\bigg(\frac{\bar{y}(\boldsymbol{\theta})}{\sqrt{2\epsilon_y^2}}\bigg). \tag{5.4}$$

For instance, consider the cumulative distribution pictured in Figure 5-3 taken from the Gaussian PDF in Figure 5-2(b). As was seen in the trajectory rollouts in Figure 5-2(a), the likelihood that $y(\boldsymbol{\theta}) > 0$ (and the trajectory satisfies the requirement) is high according to the PDF. The cumulative probability under this distribution is then $p_{sat}(\boldsymbol{\theta}) = 0.934$. For different values of $\boldsymbol{\theta}$, the center of the distribution $\bar{y}(\boldsymbol{\theta})$ will change and therefore the cumulative probability for $p_{sat}(\boldsymbol{\theta})$ will vary accordingly.

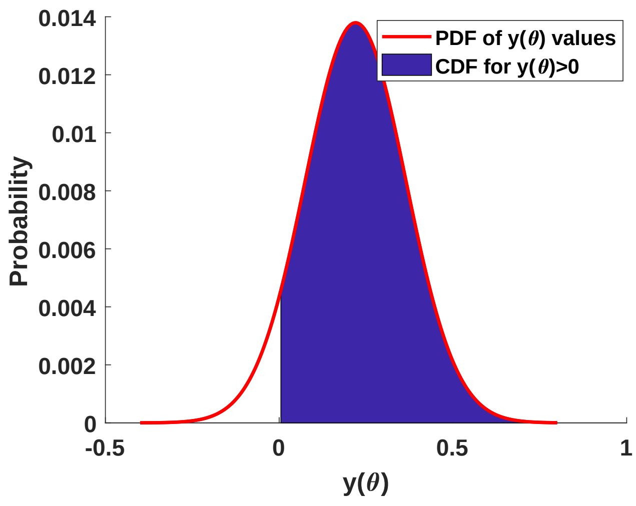


Figure 5-3: The satisfaction probability function is the cumulative distribution of the Gaussian PDF defining the likelihood of robustness measurements $y(\boldsymbol{\theta})$. This example uses the same Gaussian PDF $\mathcal{N}(0.212, 0.145)$ constructed from the histogram in Figure 5-2(b). The cumulative distribution is $p_{sat}(\boldsymbol{\theta}) = 0.934$.

Although the cumulative distribution (5.4) resembles the prediction confidence (4.9) from Section 4.2.2, these two Gaussian CDFs quantify different likelihoods. For one, the CDF in (4.9) does not measure the true satisfaction probability function, but rather the predictive uncertainty given the GP predictive mean and covariance. In the binary verification problem, the true satisfaction probability function is actually $p_{sat}(\boldsymbol{\theta}) \in \{0, 1\}$ since the trajectory will only produce one of two options. With the stochastic verification problem, even if the system's performance is perfectly modeled, the stochastic dynamics will result in a Bernoulli distribution with $p_{sat}(\boldsymbol{\theta}) \in [0, 1]$. Therefore, (4.9) and (5.4) both compute $\mathbb{P}(y(\boldsymbol{\theta}) > 0)$, but address completely different circumstances.

While the satisfaction probability function $p_{sat}(\boldsymbol{\theta})$ defines the true likelihood a trajectory will satisfy the requirements, this function will generally be unknown in advance. Without any historical information, it is not clear whether an arbitrary $\boldsymbol{\theta} \in \Theta$ maps to a region of high probability of satisfaction or not. Therefore, the new verification goal is to predict $p_{sat}(\boldsymbol{\theta})$ for all possible conditions in Θ.

Problem 5.1. *Given the stochastic closed-loop system (5.1), compute an estimated satisfaction probability function* $\widehat{p}_{sat}(\boldsymbol{\theta})$.

Unlike the deterministic verification objective in Problem 3.1, the stochastic verification objective in Problem 5.1 cannot be viewed as a binary classification problem. Later work in Chapter 7 will introduce a secondary stochastic verification problem that resembles the binary classification problem in Prob. 3.1 in order to separate Θ into two regions with $p_{sat}(\boldsymbol{\theta}) \geq p_{min}$ and $p_{sat}(\boldsymbol{\theta}) < p_{min}$, where $p_{min} \in (0, 1)$ is some minimum acceptable probability of success.

5.2 Regression-based Stochastic Verification

Given continuous measurements of trajectory robustness, regression-based approaches can be used to estimate $\widehat{p}_{sat}(\boldsymbol{\theta})$ from a finite collection of trajectories. Various regression techniques such as SVM regression models [81] or relevance vector machines (RVMs) [110] are possible, but this work uses Gaussian process regression models [85] for Bayesian estimation. Although most of the structure is the same as for the deterministic GPs in Section 4.2, the stochastic GP models include additional terms to capture the effect of ϵ_y in the measurements. The change in focus from binary sets $\widehat{\Theta}_{sat}$ and $\widehat{\Theta}_{fail}$ to satisfaction probability function $\widehat{p}_{sat}(\boldsymbol{\theta})$ also requires a different approach than Section 4.2.2 to quantify expected prediction accuracy. Section 5.2.2 introduces a new approach to quantify the uncertainty with the variance of the Gaussian CDF.

5.2.1 Gaussian Process Regression Model

The Gaussian process regression model used to compute $\widehat{p}_{sat}(\boldsymbol{\theta})$ follows the same training procedure discussed earlier in Section 2.2 and closely parallels the process from Section 4.2.1. Given a finite collection of observed trajectories, a trained GP model defines a distribution over possible $y(\boldsymbol{\theta})$ measurements at all $\boldsymbol{\theta} \in \Theta$. This collection of observed trajectories forms a training dataset $\mathcal{L}$ consisting of perturbation conditions $\mathcal{D} = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_N\}$ and corresponding measurements $\mathbf{y} = [y(\boldsymbol{\theta}_1), y(\boldsymbol{\theta}_2), \ldots, y(\boldsymbol{\theta}_N)]^T$. Unlike Section 4.2.1, the elements of vector $\mathbf{y}$ are noisy measurements.

Instead of learning the distribution directly, the Gaussian process regression model predicts the latent mean $\bar{y}(\boldsymbol{\theta})$ and couples it with standard deviation ϵ_y to define the distribution over $y(\boldsymbol{\theta})$. The standard deviation ϵ_y may or may not be known in advance. In the latter case, the Gaussian process model does not estimate ϵ_y itself, but instead relies upon numerical approximation or hyperparameter optimization to determine ϵ_y. Once this information has been provided, the GP regression model relies upon Bayesian inference to construct the predictions for $\bar{y}(\boldsymbol{\theta})$.

The prior probability distribution encapsulates all prior information about the latent mean. Gaussian process regression models assume this prior distribution is a joint multivariate Gaussian distribution between all the training points in set $\mathcal{L}$. This chapter's work assumes nothing is known about the shape of $\bar{y}(\boldsymbol{\theta})$ in advance; therefore, the prior distribution initializes with a zero mean to avoid biases in the predictions. This prior probability distribution is written as

$$\mathbb{P}(\bar{\mathbf{y}}|\mathcal{D}, \psi) = \mathcal{N}(\bar{\mathbf{y}}|\mathbf{0}, \mathbf{K}), \tag{5.5}$$

where $\mathbf{K}$ is the covariance matrix with $\mathbf{K}_{ij} = \kappa(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j)$ and $\bar{\mathbf{y}}$ is the $N \times 1$ vector for the values of the latent mean function corresponding to the noisy measurements in $\mathbf{y}$. This approach also uses the common squared exponential kernel with automatic relevance determination detailed in (2.14). The hyperparameters for the kernels are given by ψ.

While the Gaussian process models the latent mean $\bar{y}(\boldsymbol{\theta})$, these values are not directly available and the GP can only base its predictions upon noisy measurements $\mathbf{y}$. The GP training process explicitly incorporates the fact that the available measurements are noisy representations of $\bar{\mathbf{y}}$ with a likelihood model $\mathbb{P}(\mathbf{y}|\bar{\mathbf{y}}, \vartheta)$. This likelihood model introduces a second set of hyperparameters, set ϑ, which represents the standard deviation ϵ_y, i.e. $\vartheta = \epsilon_y$. Conveniently, the likelihood model can be factorized amongst the N training points with

$$\mathbb{P}(\mathbf{y}|\bar{\mathbf{y}}, \vartheta) = \prod_{i=1}^{N} \mathbb{P}(y(\boldsymbol{\theta}_i)|\bar{y}(\boldsymbol{\theta}_i), \vartheta) = \mathcal{N}(\mathbf{y}|\bar{\mathbf{y}}, \epsilon_y^2\mathbf{I}). \tag{5.6}$$

Previously in Chapter 4, the measurements were noise-free observations. The likelihood model for those noise-free measurements was a Dirac delta distribution, which can be viewed similar to a Gaussian distribution with variance 0. As the standard deviation ϵ_y of the noisy measurements shrinks, the likelihood model (5.6) begins to more closely approximate the Dirac delta distribution as noisy observations are more tightly clustered around their latent $\bar{y}(\boldsymbol{\theta})$ values.

The end result of the training process is a posterior distribution $\mathbb{P}(\bar{\mathbf{y}}|\mathcal{L}, \psi, \vartheta)$ for the latent mean $\bar{\mathbf{y}}$ corresponding to each of the training measurements. The real power of the GP regression model is its ability to compute the posterior predictive distribution for $\bar{y}(\boldsymbol{\theta})$ at unobserved locations in Θ. This posterior predictive distribution at an arbitrary location $\boldsymbol{\theta}_* \in \Theta$ is a Gaussian distribution

$$\mathbb{P}(\bar{y}(\boldsymbol{\theta}_*)|\mathcal{L}, \boldsymbol{\theta}_*, \psi, \vartheta) = \mathcal{N}\big(\mu(\boldsymbol{\theta}_*), \Sigma(\boldsymbol{\theta}_*)\big) \tag{5.7}$$

with posterior predictive mean $\mu(\boldsymbol{\theta}_*)$ and covariance $\Sigma(\boldsymbol{\theta}_*)$. These two terms are computed by

$$\begin{aligned} \mu(\boldsymbol{\theta}_*) &= \mathbf{K}_*^T(\mathbf{K} + \epsilon_y^2\mathbf{I})^{-1}\mathbf{y} \\ \Sigma(\boldsymbol{\theta}_*) &= \mathbf{K}_{**} - \mathbf{K}_*^T(\mathbf{K} + \epsilon_y^2\mathbf{I})^{-1}\mathbf{K}_* \end{aligned} \tag{5.8}$$

where scalar $\mathbf{K}_{**} = \kappa(\boldsymbol{\theta}_*, \boldsymbol{\theta}_*)$ and $\mathbf{K}_*$ is the $N \times 1$ vector of kernel function $\kappa(\boldsymbol{\theta}_*, \boldsymbol{\theta}_i)$ for $i = 1, \dots, N$. Note that (5.8) appears very similar to (4.5) except for the inclusion of $\epsilon_y^2\mathbf{I}$ from the likelihood model to reflect noisy $\mathbf{y}$.

For stochastic verification, the predictions for $\bar{y}(\boldsymbol{\theta})$ are really only one step in the process to estimate the CDF for $\mathbb{P}(y(\boldsymbol{\theta}) > 0)$. Next, the posterior predictive distribution for arbitrary $y(\boldsymbol{\theta}_*)$ is obtained by augmenting (5.7) with ϵ_y,

$$\mathbb{P}(y(\boldsymbol{\theta}_*)|\mathcal{L}, \boldsymbol{\theta}_*, \psi, \vartheta) = \mathcal{N}\big(\mu(\boldsymbol{\theta}_*), \Sigma(\boldsymbol{\theta}_*) + \epsilon_y^2\big). \tag{5.9}$$

Just as the PDF of $y(\boldsymbol{\theta})$ defines $p_{sat}(\boldsymbol{\theta})$, the PDF for $y(\boldsymbol{\theta})$ in (5.9) leads to the following posterior predictive CDF to estimate $p_{sat}(\boldsymbol{\theta}_*)$,

$$\widehat{p}_{sat}(\boldsymbol{\theta}_*) = \mathbb{P}(y(\boldsymbol{\theta}_*) > 0 | \mathcal{L}, \boldsymbol{\theta}_*, \psi, \vartheta) = \frac{1}{2} + \frac{1}{2}\,\mathrm{erf}\Big(\frac{\mu(\boldsymbol{\theta}_*)}{\sqrt{2(\Sigma(\boldsymbol{\theta}_*) + \epsilon_y^2)}}\Big). \tag{5.10}$$

Definition 5.5. *The* predicted satisfaction probability function $\widehat{p}_{sat}(\boldsymbol{\theta}) \in [0, 1]$ *predicts the likelihood an arbitrary simulation or experimental test initialized at* $\boldsymbol{\theta}$ *will satisfy the performance requirement given the evidence provided by training dataset* $\mathcal{L}$*.*

Although (5.10) is similar in appearance to (5.4), the CDF also includes $\Sigma(\boldsymbol{\theta})$ due to the uncertainty over the true value of $\bar{y}(\boldsymbol{\theta})$. This presents an equivalent, alternative perspective for the predicted satisfaction probability function in (5.10).

Remark 5.6. *The posterior predictive CDF in (5.10) is the expected value of (5.4) over different possible CDFs given the current observations in* $\mathcal{L}$*.*

This can be seen through the CDF in (5.4) marginalized over different possible values of $\bar{y}(\boldsymbol{\theta}_*)$:

$$\widehat{p}_{sat}(\boldsymbol{\theta}_*) = \mathbb{E}_{\bar{y}(\boldsymbol{\theta}_*)}\Big[\mathbb{P}(y(\boldsymbol{\theta}_*) > 0 | \bar{y}(\boldsymbol{\theta}_*), \vartheta)\Big] \tag{5.11}$$

$$= \int \mathbb{P}(y(\boldsymbol{\theta}_*) > 0 | \bar{y}(\boldsymbol{\theta}_*), \vartheta)\ \mathbb{P}(\bar{y}(\boldsymbol{\theta}_*) | \mathcal{L}, \boldsymbol{\theta}_*, \psi, \vartheta)\ d\bar{y}(\boldsymbol{\theta}_*) \tag{5.12}$$

$$= \int \Big(\frac{1}{2} + \frac{1}{2}\,\mathrm{erf}\Big(\frac{\bar{y}(\boldsymbol{\theta}_*)}{\sqrt{2\epsilon_y^2}}\Big)\Big)\ \mathcal{N}\big(\bar{y}(\boldsymbol{\theta}_*) | \mu(\boldsymbol{\theta}_*), \Sigma(\boldsymbol{\theta}_*)\big)\ d\bar{y}(\boldsymbol{\theta}_*) \tag{5.13}$$

$$= \frac{1}{2} + \frac{1}{2}\int \mathrm{erf}\Big(\frac{\bar{y}(\boldsymbol{\theta}_*)}{\sqrt{2\epsilon_y^2}}\Big)\ \mathcal{N}\big(\bar{y}(\boldsymbol{\theta}_*) | \mu(\boldsymbol{\theta}_*), \Sigma(\boldsymbol{\theta}_*)\big)\ d\bar{y}(\boldsymbol{\theta}_*). \tag{5.14}$$

The solution to the final integral in (5.14) has been shown in the table of integrals in Section 2.5.2 of [139] and ultimately reduces to (5.10).

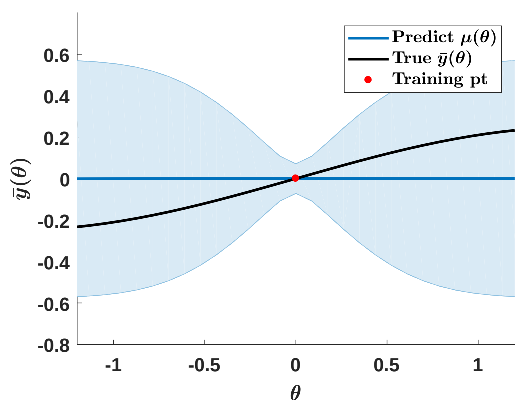
Selection of Hyperparameters

The choice of kernel hyperparameters ψ still has a substantial effect upon the GP predictions and the resulting $\widehat{p}_{sat}(\boldsymbol{\theta})$. In most applications, the true or "ideal" hyperparamters that replicate the true shape of $\bar{y}(\boldsymbol{\theta})$ are unknown and must be estimated online from training dataset $\mathcal{L}$. Same as before, this work uses maximum likelihood estimation (Section 2.2.3) for efficient computation of locally-optimum hyperparameters ψ^*.

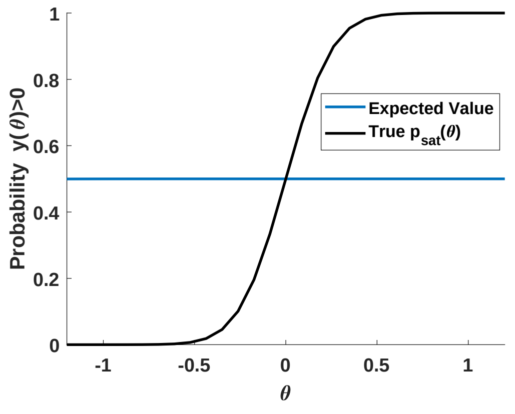
While set ψ defines the hyperparameters for the kernel function $\kappa(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j)$, stochasticity introduces a second set of hyperparameters ϑ for the likelihood model in (5.6). For Gaussian distributions, Assumption 5.3 means the hyperparameters ϑ only need to estimate constant ϵ_y. In the simplest case, this variance is already known and $\vartheta = \epsilon_y$; however, this will not always be true. When ϵ_y is unknown, it can be estimated through a number of methods. For one, the noise hyperparameter ϑ can be included in the hyperparameter optimization procedure in Section 2.2.3. The resulting locally-optimum ϑ^* then approximates ϵ_y given the current training data $\mathcal{L}$. Another straightforward option is to obtain a number of repeated trajectories at the same $\boldsymbol{\theta}$ and compute the sample variance.

5.2.2 Measuring Prediction Accuracy

While (5.9) computes the expected satisfaction probability function $\widehat{p}_{sat}(\boldsymbol{\theta})$, there will likely be some level of prediction error $\widetilde{p}_{sat}(\boldsymbol{\theta})$ between the predicted $\widehat{p}_{sat}(\boldsymbol{\theta})$ and the true (but unknown) function $p_{sat}(\boldsymbol{\theta})$, defined as $\widetilde{p}_{sat}(\boldsymbol{\theta}) = p_{sat}(\boldsymbol{\theta}) - \widehat{p}_{sat}(\boldsymbol{\theta})$. Unfortunately, the prediction confidence from (4.9) will no longer measure the prediction error/confidence due to stochastic measurements. The prediction confidence in (4.9) assumed the true satisfaction probability function had only two options, $p_{sat}(\boldsymbol{\theta}) \in \{0, 1\}$, and therefore a $\mathbb{P}(y(\boldsymbol{\theta}) > 0) \neq 0$ or 1 signified uncertainty in the predictions. Now, the true $p_{sat}(\boldsymbol{\theta}) \in [0, 1]$ and the same reasoning will not apply. For instance, consider the GP prediction output and resulting $\widehat{p}_{sat}(\boldsymbol{\theta})$ in Figure 5-4. The predictions near the training point $\boldsymbol{\theta} = 0$ compute $\widehat{p}_{sat}(\boldsymbol{\theta} = 0) = 0.5$. According to the prediction confidence in (4.9), this point would have low confidence in the accuracy of the predictions; however, this point is actually the *most* accurate prediction. This discrepancy identifies the need for a new metric to quantify prediction accuracy or at least indicate where prediction errors are likely to occur.



(a) GP output



(b) Predicted $\mathbb{P}(y(\boldsymbol{\theta}) > 0)$

Figure 5-4: Illustration of prediction error resulting from uncertainties in $\bar{y}(\boldsymbol{\theta})$. The GP prediction mean $\mu(\boldsymbol{\theta})$ and 1σ variance $\Sigma(\boldsymbol{\theta})$ bounds (blue) are shown against the true $\bar{y}(\boldsymbol{\theta})$ (black). The corresponding expected value $\widehat{p}_{sat}(\boldsymbol{\theta})$ and true $p_{sat}(\boldsymbol{\theta})$ are shown on the right. Since the mean is a constant $\mu(\boldsymbol{\theta}) = 0$, the expected value is also a constant $\widehat{p}_{sat}(\boldsymbol{\theta}) = 0.5$.

Probabilistic inequalities provide theoretically-justified methods to bound the prediction error. In particular, Chebyshev's inequality bounds the probability the prediction error $\widetilde{p}_{sat}(\boldsymbol{\theta})$ will be greater than some value when the stochastic verification problem is viewed from a Bayesian perspective [140]. In this Bayesian perspective, Chebyshev's inequality states the probability the absolute error between $p_{sat}(\boldsymbol{\theta}_*)$ and the expected value of the distribution, $\mathbb{E}_{\bar{y}(\boldsymbol{\theta}_*)}\Big[\mathbb{P}(y(\boldsymbol{\theta}_*) > 0|\bar{y}(\boldsymbol{\theta}_*), \vartheta)\Big]$, is greater than a constant $a > 0$ will be bounded by the variance of the distribution, $\mathbb{V}_{\bar{y}(\boldsymbol{\theta}_*)}\Big[\mathbb{P}(y(\boldsymbol{\theta}_*) > 0|\bar{y}(\boldsymbol{\theta}_*), \vartheta)\Big]$, and the constant a [140],

$$\mathbb{P}\Big(\big|p_{sat}(\boldsymbol{\theta}_*) - \mathbb{E}_{\bar{y}(\boldsymbol{\theta}_*)}\Big[\mathbb{P}(y(\boldsymbol{\theta}_*) > 0|\bar{y}(\boldsymbol{\theta}_*), \vartheta)\Big]\big| \geq a\Big) \leq \frac{\mathbb{V}_{\bar{y}(\boldsymbol{\theta}_*)}\Big[\mathbb{P}(y(\boldsymbol{\theta}_*) > 0|\bar{y}(\boldsymbol{\theta}_*), \vartheta)\Big]}{a^2}. \tag{5.15}$$

Note that the predicted probability of satisfaction $\widehat{p}_{sat}(\boldsymbol{\theta}_*)$ was already shown to be the expected value $\mathbb{E}_{\bar{y}(\boldsymbol{\theta}_*)}\Big[\mathbb{P}(y(\boldsymbol{\theta}_*) > 0|\bar{y}(\boldsymbol{\theta}_*), \vartheta)\Big]$ in (5.14). In other words, Chebyshev's inequality produces a prediction interval for the difference between true $p_{sat}(\boldsymbol{\theta}_*)$ and estimate $\widehat{p}_{sat}(\boldsymbol{\theta}_*)$. Lower variance will translate to a lower probability the error $\widetilde{p}_{sat}(\boldsymbol{\theta}_*)$ will be large and means a higher confidence in the accuracy of $\widehat{p}_{sat}(\boldsymbol{\theta}_*)$. The key assumption with this probability bound is that the variance is explicitly known.

The main issue with the Chebyshev inequality (5.15) and its application to the stochastic verification problem is the lack of a closed-form solution for the variance of a Gaussian CDF. Therefore, the true CDF variance is not explicitly known. Instead, the variance can be approximated using a 1st or 2nd order Taylor series expansion [141] centered around the expected value for $\mathbb{P}(y(\boldsymbol{\theta}_*) > 0|\bar{y}(\boldsymbol{\theta}_*), \vartheta)$, estimate $\widehat{p}_{sat}(\boldsymbol{\theta}_*)$. For an arbitrary nonlinear random function $Y = g(X)$, the 1st order Taylor approximation is

$$\mathbb{V}[Y] \approx \mathbb{V}[X]\big(\frac{\partial g}{\partial X}\big)^2, \tag{5.16}$$

while the 2nd order expansion is

$$\begin{aligned}\mathbb{V}[Y] \approx & \mathbb{V}[X]\big(\frac{\partial g}{\partial X}\big)^2 - \frac{1}{4}\Big[\mathbb{V}[X]\Big]^2\big(\frac{\partial^2 g}{\partial X^2}\big)^2 + \mathbb{E}\Big[\big(X - \mathbb{E}[X]\big)^3\Big]\frac{\partial g}{\partial X}\frac{\partial^2 g}{\partial X^2} \\ & + \frac{1}{4}\mathbb{E}\Big[\big(X - \mathbb{E}[X]\big)^4\Big]\big(\frac{\partial^2 g}{\partial X^2}\big)^2.\end{aligned} \tag{5.17}$$

Conveniently, $\mathbb{E}\big[\big(X - \mathbb{E}[X]\big)^3\big] = 0$ and $\mathbb{E}\big[\big(X - \mathbb{E}[X]\big)^4\big] = 0$ when X is a Gaussian random variable so the last two terms in (5.17) drop out. The 1st and 2nd order approximations for the variance $\mathbb{V}_{\bar{y}(\boldsymbol{\theta}_*)}\big[\mathbb{P}(y(\boldsymbol{\theta}_*) > 0|\bar{y}(\boldsymbol{\theta}_*), \vartheta)\big]$ are then written as

$$\mathbb{V}_{\bar{y}(\boldsymbol{\theta}_*)}\big[\mathbb{P}(y(\boldsymbol{\theta}_*) > 0|\bar{y}(\boldsymbol{\theta}_*), \vartheta)\big] \approx \frac{1}{2\pi\epsilon_y^2}e^{-\mu(\boldsymbol{\theta}_*)^2/\epsilon_y^2}\ \Sigma(\boldsymbol{\theta}_*), \tag{5.18}$$

and

$$\mathbb{V}_{\bar{y}(\boldsymbol{\theta}_*)}\big[\mathbb{P}(y(\boldsymbol{\theta}_*) > 0|\bar{y}(\boldsymbol{\theta}_*), \vartheta)\big] \approx \frac{1}{2\pi\epsilon_y^2}e^{-\mu(\boldsymbol{\theta}_*)^2/\epsilon_y^2}\ \Sigma(\boldsymbol{\theta}_*) - \frac{1}{2\pi\epsilon_y^6}\mu(\boldsymbol{\theta}_*)^2 e^{-\mu(\boldsymbol{\theta}_*)^2/\epsilon_y^2}\ \Sigma(\boldsymbol{\theta}_*)^2. \tag{5.19}$$

For simplicity and ease of writing, (5.18) will be used in place of (5.19) for the remainder of this chapter.

Although (5.18) approximates the true CDF variance, the accuracy of a Taylor series approximation of a nonlinear function is limited. More importantly, the lack of the exact value for the CDF variance violates the key assumption of the Chebyshev inequality. Therefore, the use of (5.18) for the probabilistic bounds in (5.15) is not advisable and it is not possible to quantify the prediction accuracy in the same manner

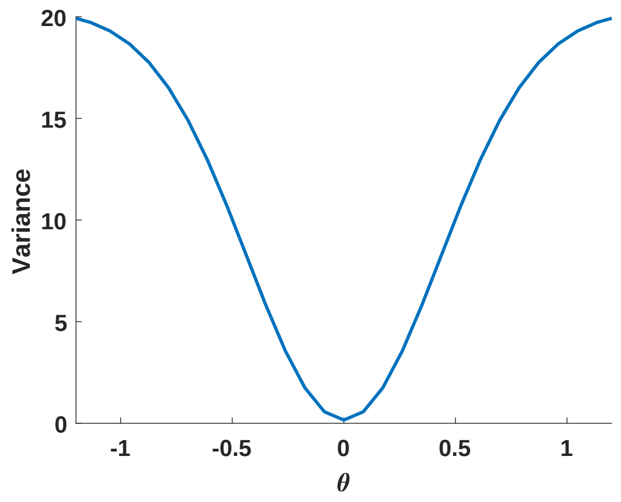


Figure 5-5: 1st order approximation of the CDF variance in the example from Figure 5-4. Note that regions of high variance around $\boldsymbol{\theta} = \pm 1$ correspond to the areas of high prediction error in Figure 5-4(b).

as (4.9). While it's not accurate enough for use in (5.15) to explicitly *quantify* the prediction accuracy, approximate CDF variance (5.18) is still a perfect metric for *qualifying* the prediction accuracy and identifying regions of Θ where the accuracy of $\widehat{p}_{sat}(\boldsymbol{\theta})$ is likely to suffer. Figure 5-5 displays the 1st order approximation of CDF variance for the previous example in Figure 5-4. The magnitude of the approximate CDF variance is unrealistically high, but it does correctly identify the regions in Θ where the prediction error is high. For instance, the prediction error $\widetilde{p}_{sat}(\boldsymbol{\theta})$ was highest near points $\boldsymbol{\theta} = \pm 1$ and lowest around $\boldsymbol{\theta} = 0$. Figure 5-5 displays the exact same behavior, demonstrating that the approximate CDF variance is a useful metric for at least identifying likely areas of prediction error, if not actually bounding the error itself. The approximate variance will be used extensively in the next section to rederive a closed-loop verification framework for stochastic systems.

5.3 Closed-Loop Statistical Verification

The changes to the Gaussian process regression model and the quantification of prediction confidence also motivate new changes to the closed-loop verification framework to adapt it to stochastic systems. The base of the framework remains almost entirely unchanged. Although the predictions have shifted from $\widehat{\Theta}_{sat}$ and $\widehat{\Theta}_{fail}$ to $\widehat{p}_{sat}(\boldsymbol{\theta})$, the high level goal is still to provide the most accurate predictions given a finite sampling budget of N_{lim} trajectories. The main change is a set of new sample selection metrics specifically tailored to the stochastic verification problem. Section 5.2.2 already previewed the limitations of the previous metrics from Section 4.3 in the discussion of the unsuitability of (4.9) to quantify prediction confidence in stochastic systems. The new metrics will redevelop their deterministic equivalents from Section 4.3 in order to replicate their advantages and improvements in prediction accuracy.

5.3.1 Sample-Selection Criteria

As before, active learning [117] forms the basis of closed-loop, stochastic verification procedures. The center of all active learning procedures is the sample selection criteria used to define the "best" future sample location for the improvement of the predictions. Not surprisingly, there exists a number of different possible selection criteria to guide the choice of sample locations and the evolution of the GP-based prediction model. Section 4.3 already described many of these possible selection metrics, but the following paragraphs will briefly repeat the most relevant techniques for the sake of clarity and to highlight their strengths and limitations.

Existing Approaches

Existing selection criteria decomposes into two general groups either focused on the PDF's predictive mean $\mu(\boldsymbol{\theta})$ or covariance $\Sigma(\boldsymbol{\theta})$. The former is an extension of the earlier SVM-based procedures in Chapter 3. The objective is to rank points according to their proximity to $\mu(\boldsymbol{\theta}) = 0$,

$$\overline{\boldsymbol{\theta}} = \text{ argmin } |\mu(\boldsymbol{\theta})|. \tag{5.20}$$

Although the original motivation for these approaches in Section 3.4 no longer applies, discussions later in this chapter will highlight how this metric inadvertently captures one of the two major influences upon prediction accuracy in stochastic verification.

The latter group focused on PDF variance $\Sigma(\boldsymbol{\theta})$ captures the second of the two major influences. These PDF variance-based approaches [92,94,117,125] are by far the most common technique and have been shown to work for stochastic systems [92, 94, 125]. Their goal is to select the sample location with the highest variance (or some subset of) to most improve the mutual information of the training set,

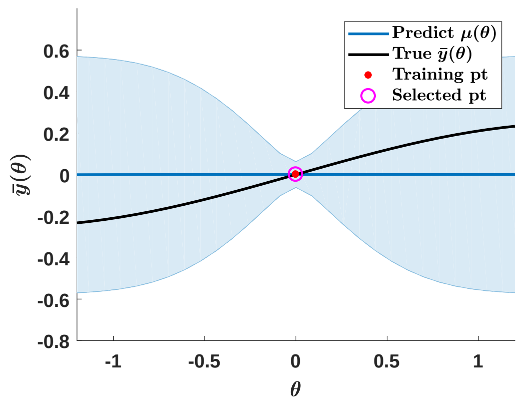
$$\overline{\boldsymbol{\theta}} = \ \text{argmax}\ \Sigma(\boldsymbol{\theta}). \tag{5.21}$$

Various extensions of these two general groupings exist [93, 117], but none of them explicitly address the stochastic verification problem.

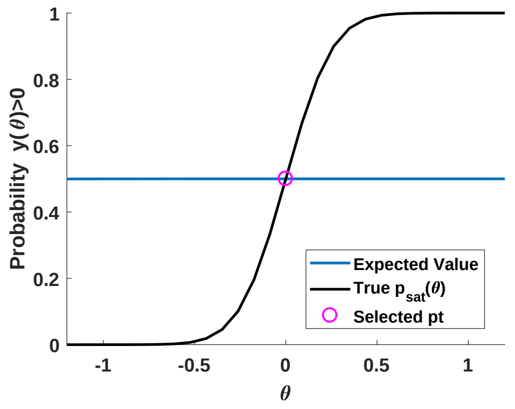
Limitation of Binary Classification Entropy

Not surprisingly, a third set of sample selection criteria to discuss are the binary classification entropy approaches from Chapter 4. The results in Section 4.4 clearly demonstrated the value of binary classification entropy-based selection criteria in deterministic verification; however, this does not necessarily translate to stochastic systems. The binary classification entropy criterion in (4.12) is a nonlinear function that emphasizes locations with $\mathbb{P}(y(\boldsymbol{\theta}) > 0|\mathcal{L}, \psi, \vartheta) \approx 0.5$ and deemphasizes those with $\mathbb{P}(y(\boldsymbol{\theta}) > 0|\mathcal{L}, \psi, \vartheta) \approx 0$ or 1. This criterion worked well for deterministic systems since the true $\mathbb{P}(y(\boldsymbol{\theta}) > 0)$ was either 0 or 1 and thus $\mathbb{P}(y(\boldsymbol{\theta}) > 0|\mathcal{L}, \psi, \vartheta) = 0.5$ indicated high uncertainty. Now with true $p_{sat}(\boldsymbol{\theta})$ falling anywhere between 0 and 1, including $p_{sat}(\boldsymbol{\theta}) = 0.5$, binary classification entropy no longer necessarily indicates uncertainty in the predictions.

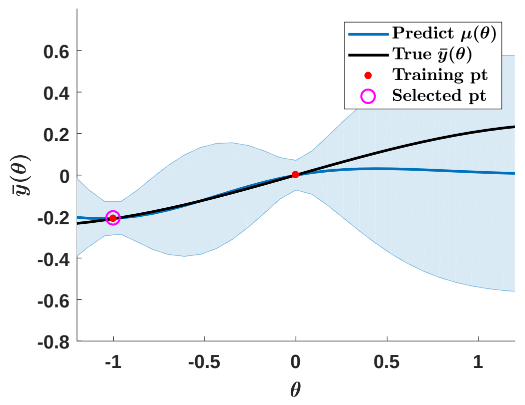
The change in suitability of $E(\boldsymbol{\theta}|\mathcal{L}, \psi, \vartheta)$ is visible in the example from Figure 5-4. All the $\boldsymbol{\theta}$ points in the figure have an expected $\mathbb{P}(y(\boldsymbol{\theta}) > 0\mathcal{L}, \psi, \vartheta) = 0.5$ and will therefore have the same binary classification entropy, but future training samples at all these points would not have an equal effect upon posterior predictions. If an additional simulation is performed at $\boldsymbol{\theta} = 0$, the resulting posterior GP model would have zero change in the predictions and prediction error, as seen in Figures 5-6(a) and (b). The lack of change is not surprising as there was already a sample at $\boldsymbol{\theta} = 0$,



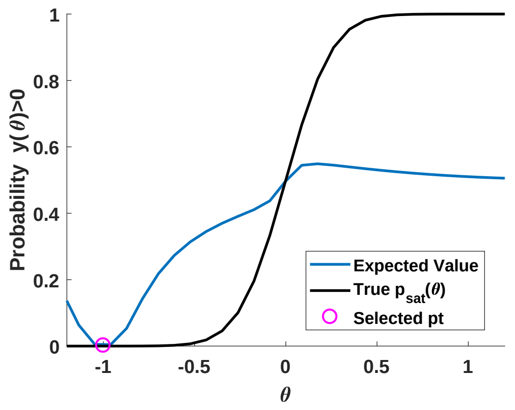
(a) GP output after a simulation test at $\boldsymbol{\theta} = 0$



(b) Predicted $\mathbb{P}(y(\boldsymbol{\theta}) > 0)$ after a simulation test at $\boldsymbol{\theta} = 0$



(c) GP output after a simulation test at $\boldsymbol{\theta} = -1$



(d) Predicted $\mathbb{P}(y(\boldsymbol{\theta}) > 0)$ after a simulation test at $\boldsymbol{\theta} = -1$

Figure 5-6: Binary classification entropy fails to quantify prediction uncertainty in stochastic systems. All the points along $\boldsymbol{\theta} : [-1, 1]$ possess the same $\widehat{p}_{sat}(\boldsymbol{\theta}) = 0.5$ and therefore would have the same binary classification entropy. According to the algorithms from Chapter 4, all these points would have the same ranking. However, a sample at $\boldsymbol{\theta} = -1$ (and also $\boldsymbol{\theta} = +1$) would produce a drastically different effect upon $\widehat{p}_{sat}(\boldsymbol{\theta})$ than a test at $\boldsymbol{\theta} = 0$, suggesting the binary entropy-based approaches from Chapter 4 are not perfectly suited to the stochastic verification problem.

but this fact is not reflected in the entropy. Instead, if a simulation is performed at $\boldsymbol{\theta} = -1$, which has the same entropy as $\boldsymbol{\theta} = 0$, the change in the posterior model in Figures 5-6(c) and (d) is much more apparent. The fact that two points with the same value of binary classification entropy produce very different results points to the limitation of binary classification entropy for selection criteria in stochastic closed-loop verification.

Reduction in Cumulative Distribution Function Variance

In place of binary classification entropy, this section presents new selection criteria based upon the variance of the cumulative distribution function.For ease of viewing, the CDF variance $\mathbb{V}_{\bar{y}(\boldsymbol{\theta})}\big[\mathbb{P}(y(\boldsymbol{\theta}) > 0|\bar{y}(\boldsymbol{\theta}), \vartheta)\big]$ will be written as $V(\boldsymbol{\theta}|\mathcal{L}, \psi, \vartheta)$. Although CDF variance replaces binary entropy $E(\boldsymbol{\theta}|\mathcal{L}, \psi, \vartheta)$ in the selection criteria of interest, it borrows many of the same concepts and motivations from Section 4.3. Additionally, the examination of the new selection criteria will also highlight that although binary classification entropy will sometimes fail to correctly delineate between certain sample locations as in Figure 5-6, it will inadvertently agree with the new CDF variance-based criteria in many instances. Since the GP prediction model is dependent upon the hyperparameters ψ and ϑ, the CDF variance will also vary with the hyperparameters. The true CDF variance is determined by marginalizing over the distribution of possible hyperparameters, but this is computationally intractable. Instead, the CDF variance uses the MLE hyperparameters ψ^* and ϑ^*.

According to (5.15), the likelihood of high prediction error is coupled to the CDF variance. Therefore, the overall goal of closed-loop verification is to minimize variance $V(\boldsymbol{\theta}|\mathcal{L}, \psi, \vartheta)$ in order to reduce the probabilistic bounds on prediction error. In an ideal scenario, the best sample $\overline{\boldsymbol{\theta}}$ would either minimize the cumulative posterior CDF variance

$$\overline{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}_*}{\operatorname{argmin}}\ V(\Theta_d|\mathcal{L}^+, \psi, \vartheta), \tag{5.22}$$

or minimize the maximum level of posterior CDF variance

$$\overline{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}_*}{\operatorname{argmin}}\ \Big(\max_{\boldsymbol{\theta}} V(\boldsymbol{\theta}|\mathcal{L}^+, \psi, \vartheta)\Big), \tag{5.23}$$

where $\mathcal{L}^+$ is the posterior training set $\mathcal{L}^+ = \mathcal{L} \cup \{\boldsymbol{\theta}_*, y(\boldsymbol{\theta}_*)\}$ after a simulation or experiment at $\boldsymbol{\theta}_*$ has been performed and measurement $y(\boldsymbol{\theta}_*)$ obtained. Just as before, a large, finite sampling set Θ_d approximates the uncountable set Θ with a fine discretization. All the calculations are performed on this set as a replica of Θ and samples are chosen from the remaining available locations $\mathcal{U} = \Theta_d \setminus \mathcal{D}$.

Unfortunately, the posterior CDF variance $V(\boldsymbol{\theta}|\mathcal{L}^+, \psi, \vartheta)$ is unavailable since $\mathcal{L}^+$ requires measurements $y(\boldsymbol{\theta}_*)$ to be known before a trajectory has been performed at that parameter setting. Instead, the infeasible posterior variance $V(\boldsymbol{\theta}|\mathcal{L}^+, \psi, \vartheta)$ in (5.22) and (5.23) can be replaced with the *expected* posterior CDF variance, $\widehat{V}(\boldsymbol{\theta}|\widehat{\mathcal{L}}^+, \psi, \vartheta)$, computed with the estimated posterior training dataset $\widehat{\mathcal{L}}^+ = \mathcal{L} \cup \{\boldsymbol{\theta}_*, \mu(\boldsymbol{\theta}_*)\}$ since $\mathbb{E}[y(\boldsymbol{\theta}_*)] = \mu(\boldsymbol{\theta}_*)$.

Even though the approximate forms of (5.22) and (5.23) are feasible since they replace $\mathcal{L}^+$ with $\widehat{\mathcal{L}}^+$, these selection metrics are computationally intractable for most verification problems with large Θ_d. The expected posterior CDF variance requires the GP regression model to be retrained for every prospective sample location in Θ_d in order to correctly compute the expected change in $\mu(\boldsymbol{\theta})$ and $\Sigma(\boldsymbol{\theta})$. As the inversion of the resulting $(N+1) \times (N+1)$ kernel matrix $\mathbf{K}$ is a $\mathcal{O}((N+1)^3)$ operation, $\mathcal{O}((N+1)^2)$ at best, it is very computationally demanding to re-invert the matrix for every prospective $\boldsymbol{\theta} \in \Theta_d$.

A more computationally tractable approach is to maximize the local improvement in CDF variance,

$$\overline{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}_*}{\operatorname{argmax}}\, \widetilde{V}(\boldsymbol{\theta}_*|\mathcal{L}, \psi, \vartheta), \tag{5.24}$$

where

$$\widetilde{V}(\boldsymbol{\theta}_*|\mathcal{L}, \psi, \vartheta) = V(\boldsymbol{\theta}_*|\mathcal{L}, \psi, \vartheta) - \widehat{V}(\boldsymbol{\theta}_*|\widehat{\mathcal{L}}^+, \psi, \vartheta). \tag{5.25}$$

In general, measurements at arbitrary location $\boldsymbol{\theta}_*$ will drastically decrease the resulting posterior variance, but (5.24) ensures samples with large prior variance $V(\boldsymbol{\theta}_*|\mathcal{L}, \psi, \vartheta)$ are ranked higher since they will see the larger amount of variance reduction. Additionally, the local decrease in CDF variance can be efficiently computed without actually having to recompute $\widehat{V}(\boldsymbol{\theta}_*|\widehat{\mathcal{L}}^+, \psi, \vartheta)$ at each prospective sample location. Assume the posterior GP predictive distribution at $\boldsymbol{\theta}_*$ after a measurement there is given by

$$\mathbb{P}(y(\boldsymbol{\theta}_*)|\mathcal{L}^+, \psi, \vartheta) = \mathcal{N}(\mu(\boldsymbol{\theta}_*)^+, \Sigma(\boldsymbol{\theta}_*)^+ + \epsilon_y^2). \tag{5.26}$$

Using the same nomenclature from (5.8), the posterior predictive mean and covariance after the measurement are

$$\mu(\boldsymbol{\theta}_*)^+ = \begin{bmatrix}\mathbf{K}_*^T & \mathbf{K}_{**}\end{bmatrix} \begin{bmatrix}\mathbf{K}+\epsilon_y^2\mathbf{I} & \mathbf{K}_* \\ \mathbf{K}_*^T & \mathbf{K}_{**}+\epsilon_y^2\end{bmatrix}^{-1} \begin{bmatrix}\mathbf{y} \\ y(\boldsymbol{\theta}_*)\end{bmatrix} \tag{5.27}$$

$$\Sigma(\boldsymbol{\theta}_*)^+ = \mathbf{K}_{**} - \begin{bmatrix}\mathbf{K}_*^T & \mathbf{K}_{**}\end{bmatrix} \begin{bmatrix}\mathbf{K}+\epsilon_y^2\mathbf{I} & \mathbf{K}_* \\ \mathbf{K}_*^T & \mathbf{K}_{**}+\epsilon_y^2\end{bmatrix}^{-1} \begin{bmatrix}\mathbf{K}_* \\ \mathbf{K}_{**}\end{bmatrix} \tag{5.28}$$

While (5.27) and (5.28) appear unwieldy, the Woodbury matrix inversion identity [126] reduces (5.27) to

$$\begin{aligned}\mu(\boldsymbol{\theta}_*)^+ = \mathbf{K}_*^T(\mathbf{K}+\epsilon_y^2\mathbf{I})^{-1}\mathbf{y} + \Big[\mathbf{K}_*^T(\mathbf{K}+\epsilon_y^2\mathbf{I})^{-1}\mathbf{K}_* - \mathbf{K}_{**}\Big] \\ \Big[\mathbf{K}_{**}+\epsilon_y^2 - \mathbf{K}_*^T(\mathbf{K}+\epsilon_y^2\mathbf{I})^{-1}\mathbf{K}_*\Big]^{-1}\Big(\mathbf{K}_*^T(\mathbf{K}+\epsilon_y^2\mathbf{I})^{-1}\mathbf{y} - y(\boldsymbol{\theta}_*)\Big).\end{aligned} \tag{5.29}$$

This further simplifies to a function of the current posterior predictive distribution,

$$\mu(\boldsymbol{\theta}_*)^+ = \mu(\boldsymbol{\theta}_*) - \Sigma(\boldsymbol{\theta}_*)\Big(\Sigma(\boldsymbol{\theta}_*)+\epsilon_y^2\Big)^{-1}\big(\mu(\boldsymbol{\theta}_*) - y(\boldsymbol{\theta}_*)\big). \tag{5.30}$$

Meanwhile, the the same matrix inversion steps that produced (5.30) reduce covariance $\Sigma(\boldsymbol{\theta}_*)^+$ to

$$\Sigma(\boldsymbol{\theta}_*)^+ = \Sigma(\boldsymbol{\theta}_*) - \Sigma(\boldsymbol{\theta}_*)\Big(\Sigma(\boldsymbol{\theta}_*)+\epsilon_y^2\Big)^{-1}\Sigma(\boldsymbol{\theta}_*) \tag{5.31}$$

$$= \Sigma(\boldsymbol{\theta}_*)\Big(1 - \frac{\Sigma(\boldsymbol{\theta}_*)}{\Sigma(\boldsymbol{\theta}_*)+\epsilon_y^2}\Big). \tag{5.32}$$

The end result is the posterior CDF variance can be approximated by

$$V(\boldsymbol{\theta}_*|\mathcal{L}^+,\psi,\vartheta) = \frac{1}{2\pi\epsilon_y^2}e^{-\mu(\boldsymbol{\theta}_*)^{+^2}/\epsilon_y^2}\;\Sigma(\boldsymbol{\theta}_*)^+, \tag{5.33}$$

and the expected posterior CDF variance is simply

$$\widehat{V}(\boldsymbol{\theta}_*|\widehat{\mathcal{L}}^+,\psi,\vartheta) = \mathbb{E}_{\bar{y}(\boldsymbol{\theta}_*)}\Big[V(\boldsymbol{\theta}_*|\mathcal{L}^+,\psi,\vartheta)\Big] \tag{5.34}$$

$$= \frac{1}{2\pi\epsilon_y^2}e^{-\mu(\boldsymbol{\theta}_*)^2/\epsilon_y^2}\;\Sigma(\boldsymbol{\theta}_*)\Big(1 - \frac{\Sigma(\boldsymbol{\theta}_*)}{\Sigma(\boldsymbol{\theta}_*)+\epsilon_y^2}\Big) \tag{5.35}$$

since $\mathbb{E}[y(\boldsymbol{\theta}_*)] = \mu(\boldsymbol{\theta}_*)$. Ultimately, the local change in posterior CDF variance (5.24) can be written purely in terms of current CDF variance and GP output as

$$\widetilde{V}(\boldsymbol{\theta}_*|\mathcal{L}, \psi, \vartheta) = \frac{1}{2\pi\epsilon_y^2} e^{-\mu(\boldsymbol{\theta}_*)^2/\epsilon_y^2}\ \Sigma(\boldsymbol{\theta}_*) \left(\frac{\Sigma(\boldsymbol{\theta}_*)}{\Sigma(\boldsymbol{\theta}_*) + \epsilon_y^2} \right), \tag{5.36}$$

meaning (5.24) can be written as

$$\overline{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}^*}{\text{argmax}} \left(\frac{1}{2\pi\epsilon_y^2} e^{-\mu(\boldsymbol{\theta}_*)^2/\epsilon_y^2}\ \Sigma(\boldsymbol{\theta}_*) \left(\frac{\Sigma(\boldsymbol{\theta}_*)}{\Sigma(\boldsymbol{\theta}_*) + \epsilon_y^2} \right) \right). \tag{5.37}$$

An examination of the new, theoretically-motivated selection metric (5.37) highlights the sensitivity of the selection criterion to the predictive mean and covariance and also provides insights into the suitability of the other existing selection metrics. From the approximation in (5.37), the selection criterion ranks points with both $|\mu(\boldsymbol{\theta})| \approx 0$ and $\Sigma(\boldsymbol{\theta}) \gg 0$ as the ones with the highest uncertainty. Locations with $|\mu(\boldsymbol{\theta})| \approx 0$ ensures the term $e^{-\mu(\boldsymbol{\theta})^2/\epsilon_y^2}$ is maximized while $\Sigma(\boldsymbol{\theta})$ multiplies with this term. Large variances $\Sigma(\boldsymbol{\theta}) \gg 0$ will have the dual effect of multiplying $e^{-\mu(\boldsymbol{\theta})^2/\epsilon_y^2}$ by a larger number as well as maximizing $\frac{\Sigma(\boldsymbol{\theta}_*)}{\Sigma(\boldsymbol{\theta}_*)+\epsilon_y^2} \to 1$, assuming $\Sigma(\boldsymbol{\theta}) \gg \epsilon_y^2$.

The selection criteria in (5.37) also explains the limitations of the existing active sampling approaches when applied to the stochastic verification problem. The selection metric in (5.20) derived from the earlier expected model change (EMC) metric in Chapters 3 and 4 focuses on minimizing the mean of the PDF, $\mu(\boldsymbol{\theta})$. This selection metric emphasizes points with $|\mu(\boldsymbol{\theta})| \approx 0$, which is one of the two trends that maximizes the local reduction in CDF variance. Therefore, even though the theoretical motivation for the PDF mean-focused metric in (5.20) from Chapter 3 no longer applies, it inadvertently captures one of the two main sources of CDF variance reduction. Likewise, the PDF variance-focused metric in (5.21) captures the second of the two trends, $\Sigma(\boldsymbol{\theta}) \gg 0$. While they each independently emphasize one of the two trends that lead to large reductions in CDF variance, they will also accidentally select uninformative samples due to incomplete information. For instance, the PDF mean-focused metric in (5.20) will rank points with low $\Sigma(\boldsymbol{\theta})$ as "informative" as long as their $\mu(\boldsymbol{\theta}) \approx 0$. Similarly, PDF variance from (5.21) will also incorrectly label points with $|\mu(\boldsymbol{\theta})| \gg 0$ as "informative" as long as the corresponding covariance $\Sigma(\boldsymbol{\theta}) \gg 0$. Both of these metrics lack the full picture and may fall into avoidable traps, but can still produce favorable results since they do emphasize one of the two important trends.

Lastly, the binary classification entropy metric from (4.12) combines aspects from both the PDF mean- and variance-based selection metrics in (5.20) and (5.21). As such, binary classification entropy does heavily weight points with $\mu(\boldsymbol{\theta}) \approx 0$, but also incorrectly emphasizes points further away from $\mu(\boldsymbol{\theta}) \approx 0$ if they have large $\Sigma(\boldsymbol{\theta})$. The example in Figure 5-6 has already highlighted the limitations and pitfalls of binary classification entropy for selection criteria in stochastic verification problems.

5.3.2 Sampling Algorithms

The new CDF variance reduction metric in (5.37) forms the basis of new closed-loop sampling algorithms. All versions of the closed-loop framework first require an initial training dataset $\mathcal{L}$ of passively-selected $\boldsymbol{\theta}$ locations and measurements from the resulting trajectories. Various open-loop, passive procedures like design of experiment techniques [73, 74] will generally produce a training dataset of adequate diversity and informativeness. This initial training dataset of size $|\mathcal{L}| = N_0$ will then produce an initial GP regression model to enable the start of active sampling. The active sampling procedures will operate until the limit on the number of simulation or experimental tests, N_{lim}, has been reached.

Sequential Sampling

The simplest closed-loop verification framework is the sequential procedure in Algorithm 7 which selects one sample at a time between GP retraining steps. Aside from the new selection criterion, the overall process remains almost unchanged from Algorithm 3. Given the initial GP regression model and required inputs (Steps 1 and 2), the procedure choses a sample location $\overline{\boldsymbol{\theta}}$ from the set of available locations

$\mathcal{U} = \Theta_d \setminus \mathcal{D}$ according to the selection metric (5.37) (Step 4). Once the algorithm performs a simulation or experiment at this location (Step 5), the procedure adds the noisy measurement $y(\overline{\boldsymbol{\theta}})$ to the training dataset, removes $\overline{\boldsymbol{\theta}}$ from $\mathcal{U}$ (Step 6), and updates the GP regression model (Step 7). The process repeats until the number of trajectories reaches the cap of $T = N_{lim} - N_0$ additional trajectories. Once it has reached this budget, the procedure terminates and returns the expected satisfaction probability function $\widehat{p}_{sat}(\boldsymbol{\theta})$ and variance $V(\boldsymbol{\theta}|\mathcal{L}, \psi, \vartheta)$ for all $\boldsymbol{\theta} \in \Theta_d$ (Step 9). In the competing PDF mean- and variance-based approaches, Step 4 is replaced by the respective sample selection metric of choice, but the rest of the algorithm remains the same. The total computational complexity is the same as for the deterministic framework in Algorithm 3.

Algorithm 7 Sequential closed-loop stochastic verification framework using GP regression models

```
1: Input: initial training set $\mathcal{L} = \{\mathcal{D}, \mathbf{y}\}$, available sample locations $\mathcal{U}$, max # of additional samples $T$
2: Initialize: train GP regression model
3: for $i = 1, 2, \ldots, T$ do
4:     Select $\overline{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \mathcal{U}}{\text{argmax}}\ \widetilde{V}(\boldsymbol{\theta}|\mathcal{L}, \psi, \vartheta)$
5:     Perform test at $\overline{\boldsymbol{\theta}}$, obtain measurement $y(\overline{\boldsymbol{\theta}})$
6:     Add $\{\overline{\boldsymbol{\theta}}, y(\overline{\boldsymbol{\theta}})\}$ to training set $\mathcal{L}$, remove $\overline{\boldsymbol{\theta}}$ from $\mathcal{U}$
7:     Retrain model with updated $\mathcal{L}$
8: end for
9: Return: expected value of the satisfaction probability function $\widehat{p}_{sat}(\boldsymbol{\theta})$ and corresponding variance $V\big(\boldsymbol{\theta}|\mathcal{L}, \psi, \vartheta\big)$
```

Batch Sampling

In many verification problems, it is advantageous to select batches of samples between retraining steps. Batch sampling lowers the computational cost by reducing the number of retraining steps (for the same N_{lim}) and is also better suited to applications with the ability to perform multiple trajectories in parallel. Again, the main challenge associated with batch sampling strategies is to encourage adequate diversity amongst the points in each batch in order to avoid redundancy.

Section 4.3.3 introduced importance-weighted probability distributions and determinantal point processes (DPPs) [130, 131] as practical methods for batch sampling with low computational overhead. In particular, the latter approach combines efficient importance sampling Monte Carlo methods [59, 127, 128] with a correlation matrix to discourage similarities among the points selected for each batch. In comparison to baseline importance-weighted random sampling, sampling with k-DPPs still steers the selection of points towards regions of high probability/informativeness, but also avoids clustering the points of the batch in close proximity to one another. As a result, the strategy more evenly distributes training points across regions of high informativeness, as was seen in Figure 4-5. This same approach is easily adopted for the stochastic verification problem as well.

One of the main aspects of the k-DPP approach is the probability distribution used to weight the informativeness of points. This work uses the CDF variance reduction selection criteria from (5.24) to form a probability distribution

$$\mathbb{P}_V(\boldsymbol{\theta}) = \frac{1}{Z_V}\widetilde{V}(\boldsymbol{\theta}|\mathcal{L},\psi,\vartheta), \tag{5.38}$$

where $Z_V = \sum_{i=1}^{|\Theta_d|}\widetilde{V}(\boldsymbol{\theta}_i|\mathcal{L},\psi,\vartheta)$. Parameter settings with a large reduction $\widetilde{V}(\boldsymbol{\theta}|\mathcal{L},\psi,\vartheta)$ will have a higher likelihood than those with low $\widetilde{V}(\boldsymbol{\theta}|\mathcal{L},\psi,\vartheta)$. The k-DPP utilizes this probability distribution to construct a correlation matrix to encourage spatial dispersion in the M points selected for the batch set $\mathcal{S}$.

Algorithm 8 details the new batch sampling algorithm for stochastic closed-loop verification. Note that this algorithm closely resembles the DPP-based procedure for deterministic systems shown earlier in Algorithm 6. Once the initial GP regression model has been obtained in Step 2, the batch selection process (Steps 3-11) actively choses the remaining $N_{lim} - N_0$ trajectories. Step 5 transforms the variance reduction into the modified probability distribution $\mathbb{P}_V(\boldsymbol{\theta})$ needed to construct the k-DPP in Step 6. The resulting k-DPP will produce a set $\mathcal{S}$ of M points for the next batch of tests (Step 7). Once these simulations or experiments have completed (Step 8), the algorithm incorporates their robustness measurements into the training dataset

Algorithm 8 Batch closed-loop stochastic verification framework using determinantal point processes

```
1:  Input: initial training set $\mathcal{L} = \{\mathcal{D}, \mathbf{y}\}$, available sample locations $\mathcal{U}$, # of iterations $T$, batch size $M$
2:  Initialize: train GP regression model
3:  for $i = 1, 2, \ldots, T$ do
4:      Initialize: $\mathcal{S} = \emptyset$
5:      Transform $\widetilde{V}(\boldsymbol{\theta}|\mathcal{L}, \psi, \vartheta)$ into probability distribution $P_V(\boldsymbol{\theta})$
6:      Generate $M_T$ random samples from $P_V(\boldsymbol{\theta})$, construct k-DPP
7:      Generate $M$ random samples from k-DPP, add to $\mathcal{S}$
8:      Perform tests $\forall \boldsymbol{\theta} \in \mathcal{S}$, obtain measurements $\mathbf{y}_{\mathcal{S}}$
9:      Add $\{\mathcal{S}, \mathbf{y}_{\mathcal{S}}\}$ to training set $\mathcal{L}$
10:     Retrain model with updated $\mathcal{L}$
11: end for
12: Return: expected value of the satisfaction probability function $\widehat{p}_{sat}(\boldsymbol{\theta})$ and corresponding variance $V\big(\boldsymbol{\theta}|\mathcal{L}, \psi, \vartheta\big)$
```

$\mathcal{L}$ (Step 9), retrains the GP model (Step 10), and repeats the iterative process. At the conclusion of T iterations, the new procedure returns the predicted satisfaction probability function $\widehat{p}_{sat}(\boldsymbol{\theta})$ (Step 12). It also provides the CDF variance $V(\boldsymbol{\theta}|\mathcal{L}, \psi, \vartheta)$ to indicate regions where the accuracy of these predictions is likely to suffer. The total computational complexity is the same as Algorithm 6.

5.4 Extension: Heteroscedastic Gaussian Distributions

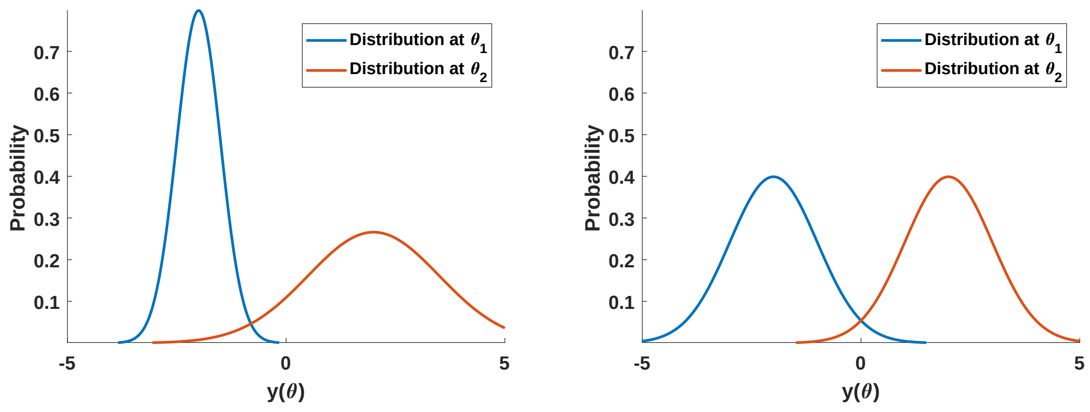
The earlier sections in this chapter addressed Gaussian distributions of $y(\boldsymbol{\theta})$ and introduced the data-driven stochastic verification framework as it applied to these problems. In practice, Assumption 5.3 will not hold for all systems, which restricts the applicability of the techniques in Sections 5.2 and 5.3. The following section will extend the stochastic verification framework and slightly modify the approach to include a wider class of systems with spatially-varying Gaussian noise. Later work in Section 5.5 will propose further modifications to the framework in order to address non-Gaussian distributions.

The major change in this extension is the relaxation of Assumption 5.3 to include heteroscedastic Gaussian distributions. In this new version of the stochastic verification problem, the true distribution of continuous measurements $y(\boldsymbol{\theta})$ is still a Gaussian distribution, but the width of the distribution is allowed to vary with the parameters $\boldsymbol{\theta}$. This is formally defined in the following assumption which replaces Assumption 5.3.

Assumption 5.7. *The distribution of continuous measurements $y(\boldsymbol{\theta})$ at every $\boldsymbol{\theta}$ is a Gaussian distribution $y(\boldsymbol{\theta}) \sim \mathcal{N}(\bar{y}(\boldsymbol{\theta}), \epsilon_y^2(\boldsymbol{\theta}))$ with spatially-dependent mean $\bar{y}(\boldsymbol{\theta})$ and spatially-dependent variance $\epsilon_y^2(\boldsymbol{\theta})$.*

Assumption 5.3 restricted the distribution of $y(\boldsymbol{\theta})$ to homoscedastic Gaussian distributions, where ϵ_y was independent of $\boldsymbol{\theta}$. Now, Assumption 5.7 expands the set of allowable distributions to include *heteroscedastic* Gaussians [138], where $\epsilon_y(\boldsymbol{\theta})$ is an explicit function of parameters $\boldsymbol{\theta}$. Such distributions are possible in many nonlinear systems. For instance, stochastic wind turbulence models like the common Dryden wind field model [15] are highly nonlinear functions of multiple parameters like altitude and wind-shear intensity and the resulting effect of the stochastic wind disturbance will vary with those parameters. An initial exploration of a stochastic version of the previous lateral-directional autopilot example combined with the Dryden wind field model has empirically observed the resulting performance measurements $y(\boldsymbol{\theta})$ are distributed as heteroscedastic Gaussians. This example demonstrates that spatially-varying Gaussian distributions are present in complex nonlinear systems and are a relevant extension to consider.

A comparison of homoscedastic and heteroscedastic Gaussian distributions is shown in Figure 5-7. The left-hand plot pictures two Gaussian distributions centered at $\bar{y}(\boldsymbol{\theta}_1) = -2$ and $\bar{y}(\boldsymbol{\theta}_2) = 2$ with variances $\epsilon_y^2(\boldsymbol{\theta}_1) = 0.5$ and $\epsilon_y^2(\boldsymbol{\theta}_2) = 1.5$. From the figure, it is clear the second distribution $\mathcal{N}(\bar{y}(\boldsymbol{\theta}_2), \epsilon_y^2(\boldsymbol{\theta}_2))$ has a much larger width and the first distribution $\mathcal{N}(\bar{y}(\boldsymbol{\theta}_1), \epsilon_y^2(\boldsymbol{\theta}_1))$ has virtually no probability of satisfactory performance, i.e. $\mathbb{P}(y(\boldsymbol{\theta}_2) > 0) \approx 0$. However, when both of these distributions are approximated with homoscedastic distributions in Figure 5-7(b), the approximate dis-



(a) Heteroscedastic Gaussian distributions (b) Approximation as homoscedastic Gaussians

Figure 5-7: Comparison of heteroscedastic Gaussian distributions against approximations using homoscedastic Gaussians. In the right-hand plot, both approximations overestimate the probability of $y(\boldsymbol{\theta}) > 0$, which is dangerous in verification applications.

tributions clearly fail to replicate the true distributions. Here, the homoscedastic variance is the average between the two heteroscedastic variances, $\epsilon_y^2(\boldsymbol{\theta}_1) = \epsilon_y^2(\boldsymbol{\theta}_2) = 1$.

More importantly, the homoscedastic approximations in Figure 5-7(b) misrepresent the true satisfaction probability function $p_{sat}(\boldsymbol{\theta})$. When the variance is an explicit function of $\boldsymbol{\theta}$, the satisfaction probability function from (5.4) requires a slight modification,

$$p_{sat}(\boldsymbol{\theta}) = \mathbb{P}(y(\boldsymbol{\theta}) > 0) = \frac{1}{2} + \frac{1}{2}\,\text{erf}\left(\frac{\bar{y}(\boldsymbol{\theta})}{\sqrt{2\epsilon_y^2(\boldsymbol{\theta})}}\right). \tag{5.39}$$

If the true heteroscedastic distributions are approximated with homoscedastic Gaussians, this will have a corresponding effect upon $p_{sat}(\boldsymbol{\theta})$. As mentioned earlier, the true distribution for $y(\boldsymbol{\theta}_1)$ has virtually no likelihood of satisfactory performance. This no longer holds for the homoscedastic approximation in the right-hand plot and it predicts a nonzero likelihood of success. The same overapproximation of $\mathbb{P}(y(\boldsymbol{\theta}_2) > 0)$ occurs for the second distribution. From a verification prospective, these overapproximations are unsafe since they predict a higher likelihood of success than reality, and could have easily been avoided by modeling the distributions as heteroscedastic Gaussians.

5.4.1 Heteroscedastic Gaussian Process Regression Model

When the variance $\epsilon_y^2(\boldsymbol{\theta})$ varies wildly across Θ, not only will the approximations of $\mathbb{P}(y(\boldsymbol{\theta}))$ and $p_{sat}(\boldsymbol{\theta})$ suffer, but the underlying non-homoscedastic Gaussian distribution behind noisy measurements $y(\boldsymbol{\theta})$ will also corrupt a standard GP regression model from (5.7) trained on this data. Luckily, heteroscedastic Gaussian processes (HGPs) [138, 142–144] were specifically developed to avoid these problems and model Gaussian distributions with spatially-dependent variance. The overall structure of these HGPs is almost entirely the same as before. For simplicity, say the spatially-dependent variance is a nonlinear function $\epsilon_y^2(\boldsymbol{\theta}) = r(\boldsymbol{\theta})$. Given the function for $r(\boldsymbol{\theta})$, the posterior predictive distribution for $y(\boldsymbol{\theta}_*)$ at arbitrary $\boldsymbol{\theta}_*$ is $\mathbb{P}(y(\boldsymbol{\theta}_*)|\mathcal{L}, \boldsymbol{\theta}_*, \psi, \vartheta) = \mathcal{N}(\mu(\boldsymbol{\theta}_*), \Sigma(\boldsymbol{\theta}_*) + r(\boldsymbol{\theta}_*))$, where

$$\begin{aligned} \mu(\boldsymbol{\theta}_*) &= \mathbf{K}_*^T(\mathbf{K}+\mathbf{R})^{-1}\mathbf{y} \\ \Sigma(\boldsymbol{\theta}_*) &= \mathbf{K}_{**} - \mathbf{K}_*^T(\mathbf{K}+\mathbf{R})^{-1}\mathbf{K}_* \end{aligned} \tag{5.40}$$

and $\mathbf{R}$ is a diagonal matrix of $r(\boldsymbol{\theta})$ values at each of the training points. The problem is that $r(\boldsymbol{\theta})$ is typically not known in advance and thus it will be impossible to actually compute $\mathbb{P}(y(\boldsymbol{\theta}_*)|\mathcal{L}, \boldsymbol{\theta}_*, \psi, \vartheta)$ according to (5.40). Even if the matrix $\mathbf{R}$ were available by repeating simulations or experiments at each of the training locations, the predicted variance $r(\boldsymbol{\theta}_*)$ at unobserved locations is still completely unknown. The unknown shape of $r(\boldsymbol{\theta})$ is a major implementation challenge.

Heteroscedastic Gaussian process models introduce a second GP specifically for the purpose of modeling the unknown variance function $r(\boldsymbol{\theta})$. Rather than model $r(\boldsymbol{\theta})$ directly since $r(\boldsymbol{\theta}) > 0$, the second GP models its logarithm, $s(\boldsymbol{\theta}) = log(r(\boldsymbol{\theta}))$. The two GP models will be trained in parallel: one for latent mean $\bar{y}(\boldsymbol{\theta})$ and one for the variance logarithm $s(\boldsymbol{\theta})$. Both of these GPs are given suitable Gaussian priors,

$$\bar{y}(\boldsymbol{\theta}) \sim \mathcal{N}\big(0, \kappa_y(\boldsymbol{\theta}, \boldsymbol{\theta}')\big) \tag{5.41}$$

for $\bar{y}(\boldsymbol{\theta})$ and

$$s(\boldsymbol{\theta}) \sim \mathcal{N}\big(\mu_0, \kappa_s(\boldsymbol{\theta}, \boldsymbol{\theta}')\big) \tag{5.42}$$

for $s(\boldsymbol{\theta})$. The two GPs both employ squared exponential kernels, with $\kappa_y(\boldsymbol{\theta}, \boldsymbol{\theta}')$ and $\kappa_s(\boldsymbol{\theta}, \boldsymbol{\theta}')$ delineating between the two in order to avoid confusion. Each kernel function also requires a different set of hyperparameters. Set ψ still refers to the hyperparameters for the kernel function for $\bar{y}(\boldsymbol{\theta})$, $\kappa_y(\boldsymbol{\theta}, \boldsymbol{\theta}')$. However, since the likelihood model from (5.6) is replaced with the second GP for $s(\boldsymbol{\theta})$, hyperparameters ϑ now refer to the set of hyperparameters for $\kappa_s(\boldsymbol{\theta}, \boldsymbol{\theta}')$ plus the additional nonzero-mean prior μ_0. At the conclusion of the training process, the posterior predictive distribution for measurement $y(\boldsymbol{\theta})$ is given by the integral

$$\mathbb{P}(y(\boldsymbol{\theta}_*)|\mathcal{L}, \boldsymbol{\theta}_*, \psi, \vartheta) = \int\int \mathbb{P}(y(\boldsymbol{\theta}_*)|\mathcal{L}, \boldsymbol{\theta}_*, \mathbf{s}, s(\boldsymbol{\theta}_*), \psi)\ \mathbb{P}(\mathbf{s}, s(\boldsymbol{\theta}_*)|\mathcal{L}, \boldsymbol{\theta}_*, \vartheta)\ d\mathbf{s}\ ds(\boldsymbol{\theta}_*), \tag{5.43}$$

where $\mathbb{P}(\mathbf{s}, s(\boldsymbol{\theta}_*)|\mathcal{L}, \boldsymbol{\theta}_*, \vartheta)$ is the second GP's output for the joint distribution of logarithmic variance at each of the training points (vector $\mathbf{s}$) and query location $\boldsymbol{\theta}_*$.

Variational Heteroscedastic GP Regression

Although (5.43) is the true distribution for $y(\boldsymbol{\theta})$, the integral cannot be solved analytically and the HGP approaches approximate this result. The main difference between the various HGP approaches is their handling of an approximate solution to (5.43). Early work [144] treated the integral with a fully Bayesian, Markov chain Monte Carlo (MCMC) approximation which proved to be quite computationally demanding. Later work [143] instead took a maximum a posteriori (MAP) approach to reduce the computational overhead, but was susceptible to overfitting. The work in this extension utilizes one of the most recent techniques, variational heteroscedastic Gaussian processes (VHGPs) [138, 142], which exploits variational inference to lower bound the marginal likelihood and produce an approximation that is both accurate and efficient.

A more detailed description of the VHGP process is found in its source material [138], but the central ideal is to lower bound the marginal log-likelihood (model evidence) of the HGP with analytically tractable variational approximations. The following function F lower bounds the marginal log-likelihood ($\log \mathbb{P}(\mathbf{y})$) for any choice of variational PDFs $q(\bar{\mathbf{y}})$ and $q(\mathbf{s})$,

$$F\big(q(\bar{\mathbf{y}}), q(\mathbf{s})\big) = \log \mathbb{P}(\mathbf{y}) - KL\big(q(\bar{\mathbf{y}}), q(\mathbf{s})||\mathbb{P}(\bar{\mathbf{y}}, \mathbf{s}|\mathbf{y})\big). \tag{5.44}$$

The training process is more complex than for a standard homoscedastic GP regression model, but the posterior predictive output of the VHGP approach is still defined by Gaussian distributions. At an arbitrary location $\boldsymbol{\theta}_*$, the predictive distribution for log variance $s(\boldsymbol{\theta}_*)$ is

$$q(s(\boldsymbol{\theta}_*)) = \mathcal{N}\big(s(\boldsymbol{\theta}_*)|\mu_s(\boldsymbol{\theta}_*), \Sigma_s(\boldsymbol{\theta}_*)\big), \tag{5.45}$$

while the predictive distribution for $\bar{y}(\boldsymbol{\theta}_*)$ is

$$q(\bar{y}(\boldsymbol{\theta}_*)) = \mathcal{N}\big(\bar{y}(\boldsymbol{\theta}_*)|\mu_{\bar{y}}(\boldsymbol{\theta}_*), \Sigma_{\bar{y}}(\boldsymbol{\theta}_*)\big). \tag{5.46}$$

However, the computation of the posterior predictive mean and covariance of $s(\boldsymbol{\theta}_*)$ is noticeably different due to the variational parameters, diagonal matrix $\boldsymbol{\Lambda}$ and scalar μ_0,

$$\begin{aligned} \mu_s(\boldsymbol{\theta}_*) &= \mathbf{K}_{s*}^T(\boldsymbol{\Lambda} - 0.5\mathbf{I})\mathbf{1} + \mu_0 \\ \Sigma_s(\boldsymbol{\theta}_*) &= \mathbf{K}_{s**} - \mathbf{K}_{s*}^T(\mathbf{K}_s + \boldsymbol{\Lambda}^{-1})^{-1}\mathbf{K}_{s*}\ , \end{aligned} \tag{5.47}$$

where scalar $\mathbf{K}_{s**} = \kappa_s(\boldsymbol{\theta}_*, \boldsymbol{\theta}_*)$, vector $\mathbf{K}_{s*} = \kappa_s(\boldsymbol{\theta}_*, \boldsymbol{\theta}_i)$ for $i = 1, \ldots, N$, and $\mathbf{K}_s$ is the $N \times N$ matrix of κ_s evaluated across all N training locations. The variational parameters $\boldsymbol{\Lambda}$, as well as the kernel hyperparameters ψ and ϑ, are determined through maximum likelihood estimation of the variational bound F. Once the distribution for $q(s(\boldsymbol{\theta}_*))$ has been obtained, the computation for $q(\bar{y}(\boldsymbol{\theta}_*))$ follows the standard (non-variational) GP format with

$$\begin{aligned}\mu_{\bar{y}}(\boldsymbol{\theta}_*) &= \mathbf{K}_{\bar{y}*}^T(\mathbf{K}_{\bar{y}} + \mathbf{R})^{-1}\mathbf{y} \\ \Sigma_{\bar{y}}(\boldsymbol{\theta}_*) &= \mathbf{K}_{\bar{y}**} - \mathbf{K}_{\bar{y}*}^T(\mathbf{K}_{\bar{y}} + \mathbf{R})^{-1}\mathbf{K}_{\bar{y}*}\ ,\end{aligned} \tag{5.48}$$

where $\mathbf{R}$ is a diagonal matrix with $[\mathbf{R}]_{ii} = e^{[\mu_s]_i - 0.5[\Sigma_s]_{ii}}$. Ultimately, the variational approximation of the posterior predictive distribution for noisy measurement $y(\boldsymbol{\theta}_*)$ is

$$q(y(\boldsymbol{\theta}_*)) = \int\int \mathbb{P}\big(y(\boldsymbol{\theta}_*)|\bar{y}(\boldsymbol{\theta}_*), s(\boldsymbol{\theta}_*)\big)\ q(\bar{y}(\boldsymbol{\theta}_*))\ q(s(\boldsymbol{\theta}_*))\ d\bar{y}(\boldsymbol{\theta}_*)\ ds(\boldsymbol{\theta}_*) \tag{5.49}$$

$$= \int \mathcal{N}\big(y(\boldsymbol{\theta}_*)|\mu_{\bar{y}}(\boldsymbol{\theta}_*), \Sigma_{\bar{y}}(\boldsymbol{\theta}_*) + e^{s(\boldsymbol{\theta}_*)}\big)\ \mathcal{N}\big(s(\boldsymbol{\theta}_*)|\mu_s(\boldsymbol{\theta}_*), \Sigma_s(\boldsymbol{\theta}_*)\big)\ ds(\boldsymbol{\theta}_*). \tag{5.50}$$

Although the two GPs both produce Gaussian distributions, the resulting predictive distribution $q(y(\boldsymbol{\theta}_*))$ is not Gaussian and the integral in (5.50) is not analytically tractable. Fortunately, the mean and variance of $q(y(\boldsymbol{\theta}_*))$ can be computed analytically as

$$\mathbb{E}[q(y(\boldsymbol{\theta}_*))] = \mu_{\bar{y}}(\boldsymbol{\theta}_*) \qquad \text{and} \qquad \mathbb{V}[q(y(\boldsymbol{\theta}_*))] = \Sigma_{\bar{y}}(\boldsymbol{\theta}_*) + e^{\mu_s(\boldsymbol{\theta}_*) + 0.5\Sigma_s(\boldsymbol{\theta}_*)}. \tag{5.51}$$

5.4.2 Modifications to the Stochastic Verification Framework

The stochastic verification framework incorporates the VHGP regression model in the same manner as the standard GP model. Unlike the previous VHGP applications [138, 142] which focused on the posterior predictive distribution for $y(\boldsymbol{\theta})$, stochastic verification needs the cumulative distribution for $\mathbb{P}(y(\boldsymbol{\theta}) > 0)$ rather than the PDF of $y(\boldsymbol{\theta})$. Redevelopments of the closed-loop verification algorithms require both the expected value and variance of this cumulative distribution. Since the integral in (5.50) is not analytical and the distribution is non-Gaussian, no closed-form solution for the cumulative distribution for $\mathbb{P}(y(\boldsymbol{\theta}_*) > 0)$ exists. Instead, the expected value of the cumulative distribution in (5.39) can be approximated as a Gaussian CDF from

$\mathbb{E}[q(y(\boldsymbol{\theta}_*))]$ and $\mathbb{V}[q(y(\boldsymbol{\theta}_*))]$. The resulting expected value of $p_{sat}(\boldsymbol{\theta})$ is

$$\widehat{p}_{sat}(\boldsymbol{\theta}) = \mathbb{E}\Big[\mathbb{P}(y(\boldsymbol{\theta}_*) > 0|\boldsymbol{\mathcal{L}}, \boldsymbol{\theta}_*, \psi, \vartheta)\Big] \tag{5.52}$$

$$= \frac{1}{2} + \frac{1}{2}\,\mathrm{erf}\Big(\frac{\mu(\boldsymbol{\theta}_*)}{\sqrt{2(\Sigma(\boldsymbol{\theta}_*) + e^{\mu_s(\boldsymbol{\theta}_*)+0.5\Sigma_s(\boldsymbol{\theta}_*)})}}\Big), \tag{5.53}$$

which closely parallels the the original homoscedastic predictions in (5.10).

Likewise, closed-loop verification algorithms need the variance of the cumulative distribution for the selection criteria. This variance had no closed-form solution with homoscedastic Gaussian distributions and will not have one now. The same Taylor series approximate [141] from (5.18) will be used to compute the variance of the predictions for (5.39). Due to the addition of the second GP for $s(\boldsymbol{\theta})$, the approximation is slightly more complicated; however, the distributions of $q(s(\boldsymbol{\theta}_*))$ and $q(\bar{y}(\boldsymbol{\theta}_*))$ can be treated as independent random variables. The first-order Taylor expansion of an arbitrary nonlinear random function $Y = g(X_1, X_2)$ is then

$$\mathbb{V}[Y] \approx \sum_{i=1}^{2} \Big(\frac{\partial g}{\partial X_i}\Big)^2\Big|_{E[X_1],E[X_2]} \mathbb{V}[X_i]. \tag{5.54}$$

Given the partial derivatives of (5.39) with respect to $\bar{y}(\boldsymbol{\theta}_*)$ and $s(\boldsymbol{\theta}_*)$ are

$$\frac{\partial \mathbb{P}(y(\boldsymbol{\theta}_*)|\bar{y}(\boldsymbol{\theta}_*), s(\boldsymbol{\theta}_*)}{\partial \bar{y}(\boldsymbol{\theta}_*)} = \frac{\partial}{\partial \bar{y}(\boldsymbol{\theta}_*)}\Big(\frac{1}{2} + \frac{1}{2}\,\mathrm{erf}\Big(\frac{\bar{y}(\boldsymbol{\theta}_*)}{\sqrt{2s(\boldsymbol{\theta}_*)}}\Big)\Big) \tag{5.55}$$

$$= \frac{1}{\sqrt{2\pi}}\frac{1}{\sqrt{e^{s(\boldsymbol{\theta}_*)}}} e^{-0.5e^{-s(\boldsymbol{\theta}_*)}\bar{y}(\boldsymbol{\theta}_*)^2} \tag{5.56}$$

and

$$\frac{\partial \mathbb{P}(y(\boldsymbol{\theta}_*)|\bar{y}(\boldsymbol{\theta}_*), s(\boldsymbol{\theta}_*)}{\partial s(\boldsymbol{\theta}_*)} = \frac{\partial}{\partial s(\boldsymbol{\theta}_*)}\Big(\frac{1}{2} + \frac{1}{2}\,\mathrm{erf}\Big(\frac{\bar{y}(\boldsymbol{\theta}_*)}{\sqrt{2s(\boldsymbol{\theta}_*)}}\Big)\Big) \tag{5.57}$$

$$= \frac{-1}{2\sqrt{2\pi}}\frac{\bar{y}(\boldsymbol{\theta}_*)}{\sqrt{e^{s(\boldsymbol{\theta}_*)}}} e^{-0.5e^{-s(\boldsymbol{\theta}_*)}\bar{y}(\boldsymbol{\theta}_*)^2}, \tag{5.58}$$

the first-order Taylor series approximation of the CDF variance is

$$\mathbb{V}\Big[\mathbb{P}(y(\boldsymbol{\theta}_*) > 0|\mathcal{L}, \boldsymbol{\theta}_*, \psi, \vartheta)\Big] = \frac{1}{2\pi} e^{-\mu_{\bar{y}}(\boldsymbol{\theta}_*)^2 e^{-\mu_s(\boldsymbol{\theta}_*)} - \mu_{\bar{y}}} \Sigma_{\bar{y}}(\boldsymbol{\theta}_*) \\ + \frac{1}{8\pi} \mu_{\bar{y}}(\boldsymbol{\theta}_*)^2 e^{-\mu_{\bar{y}}(\boldsymbol{\theta}_*)^2 e^{-\mu_s(\boldsymbol{\theta}_*)} - \mu_{\bar{y}}} \Sigma_s(\boldsymbol{\theta}_*). \tag{5.59}$$

A second-order Taylor expansion can also be used in place of the first-order approximation in (5.59). For closed-loop verification with heteroscedastic distributions, the expected value (5.53) and variance (5.59) equations directly replace their homoscedastic equivalents in the closed-loop algorithms from Section 5.3 and no further modifications are required. The final example in Section 5.6 will examine closed-loop verification with VHGPs.

5.5 Discussion: Non-Gaussian Distributions

Although the extension in Section 5.4 expands the class of systems considered by the data-driven, stochastic verification approaches presented in this chapter, there still exists a class of relevant systems that have not been addressed yet. Both Assumptions 5.3 and 5.7 restrict the class of applicable systems to those with Gaussian likelihoods for the distribution of noisy measurements $y(\boldsymbol{\theta})$, i.e.

$$y(\boldsymbol{\theta}) = \bar{y}(\boldsymbol{\theta}) + w_y \qquad w_y \sim \mathcal{N}(0, \epsilon_y^2); \tag{5.60}$$

however, the distribution of $y(\boldsymbol{\theta})$ is non-Gaussian in many applications. For instance, nonlinearities in the closed-loop dynamics might produce a multi-modal distribution for $y(\boldsymbol{\theta})$. This non-Gaussian likelihood model for $y(\boldsymbol{\theta})$ greatly complicates the implementation of Gaussian process regression for predictive inference and the non-Gaussian distribution of measurements can lead standard GP methods to become sensitive to outliers [145].

A variety of regression techniques have been developed for modeling and inference with non-Gaussian likelihoods [145–149]. The vast majority of these approaches [145–147] are concerned with regression in the presence of Student's t-distributions, which produce similar distributions to Gaussian likelihood models, but with heavier tails. One of the more recent techniques for regression with non-Gaussian likelihood models is Meta-GPs [149], which uses mixtures of Gaussians to capture non-Gaussian distributions. This mixture enables Meta-GPs to consider not just Student's t-distribution, but other relevant distributions such as multi-modal distributions, and do so without extra modification.

The central assumption of Meta-GPs is that the non-Gaussian likelihood can be accurately modeled as a mixture of Gaussians. Where previously a standard GP assumed

$$\mathbb{P}(y(\boldsymbol{\theta})|\bar{y}(\boldsymbol{\theta})) = \mathcal{N}(y(\boldsymbol{\theta})|\bar{y}(\boldsymbol{\theta}), \epsilon_y^2), \tag{5.61}$$

now the Meta-GP assumes

$$\mathbb{P}(y(\boldsymbol{\theta})|\bar{y}(\boldsymbol{\theta})) = \sum_{i=1}^{K} \pi_i \mathcal{N}(\mu_i + \bar{y}(\boldsymbol{\theta}), \sigma_L^2), \tag{5.62}$$

where π_i is the weighting term associated with each Gaussian distribution centered at $\mu_i + \bar{y}(\boldsymbol{\theta})$ with bandwidth σ_L. Given a mixture of K Gaussians, the sum will be able to approximate nearly any possible likelihood model. The Meta-GP approach in its current form relies upon the sparse-GP [150] representation with a fixed cap on the number of training points, but can be rewritten to instead incorporate the baseline, non-sparse GP regression model representations used in this thesis.

One important limitation of Meta-GPs, and all the surveyed methods for non-Gaussian likelihoods, is that they do not consider spatially-varying likelihood distributions. Just as homoscedastic GPs assume the variance ϵ_y remains constant, Meta-GPs assume the mixture of Gaussians is also independent of $\boldsymbol{\theta}$. Non-Gaussian regression techniques lack an equivalent to heteroscedastic GP regression. Data-driven stochastic verification would benefit immensely from a comprehensive method for regression in the presence of spatially-varying non-Gaussian likelihood models, particularly one with the wide applicability of Meta-GPs. However, such a technique does not exist currently. Once one becomes available, the open- and closed-loop stochastic verification frameworks should be modified to exploit that technique as it would enable the verification procedures to handle virtually any possible distribution for noisy, continuous measurements.

Due to the relative infancy of practical non-Gaussian regression techniques, namely Meta-GPs [149], this thesis does not delve any further into the implementation of those techniques nor apply them to example problems. Future work in Chapter 8 discusses application of these new techniques to versions of the stochastic systems shown in the next section. This section was included to provide background on an obvious limitation of the GP-based methods presented in this chapter and identify a clear roadmap towards addressing that limitation.

5.6 Simulation Results

The closed-loop verification algorithms are demonstrated on various systems with stochastic dynamics. The first example is a stochastic variant of the example from Section 4.4.1 and will be used as the primary case study to highlight the various aspects of stochastic verification. The subsequent examples will reaffirm the observations and discussions from the first example.

5.6.1 Concurrent Learning Model Reference Adaptive Controller

The first example is the same concurrent learning model reference adaptive control (CL-MRAC) system from Section 4.4.1 and 3.5.2. An uncertain linear system is subject to two uncertain parameters $\boldsymbol{\theta} = [\theta_1, \theta_2]^T$ which are estimated by the CL-MRAC adaptive law. However, unlike those previous two systems, the open-loop dynamics are corrupted by an additive process noise term $\mathbf{w}(t)$,

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -0.2+\theta_1 & -0.2+\theta_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) + \mathbf{w}(t) \ . \tag{5.63}$$

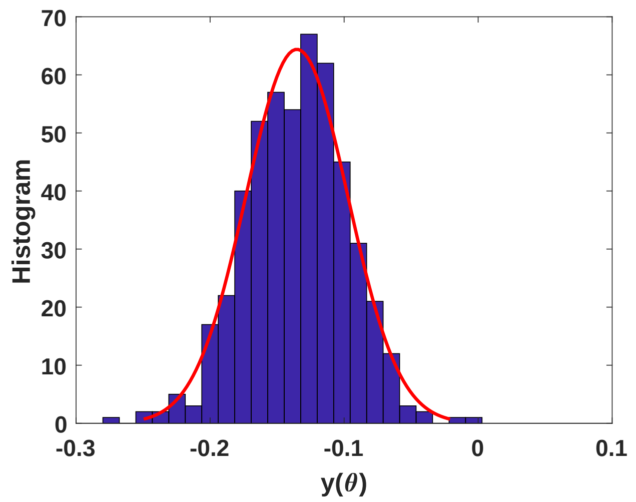


Figure 5-8: [Example 5.6.1] Histogram of robustness measurements from 500 repeated trajectories at arbitrary location $\boldsymbol{\theta} = [6, 1.8]^T$. The distribution of the robustness measurements is roughly Gaussian as a Gaussian fit (red line) to the distribution closely matches the histogram data.

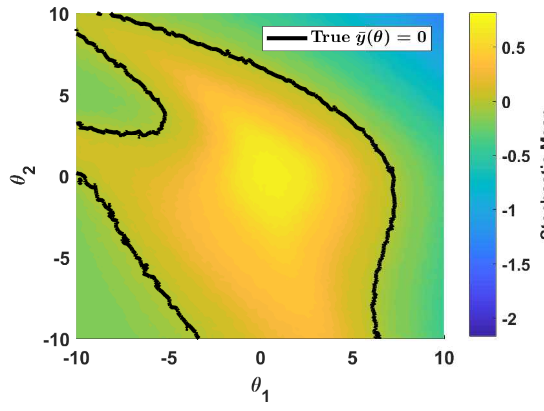
This process noise is a zero-mean Gaussian $\mathbf{w}(t) \sim \mathcal{N}(\mathbf{0}, \Sigma_w)$ with covariance matrix $\Sigma_w = \text{diag}([5, 5])$. This process noise is the sole source of stochasticity in the closed-loop system dynamics.

The performance requirement and measurement techniques are also the same as Section 4.4.1. The performance requirement states the actual state $x_1(t)$ must remain within a unit ball of reference state $x_{m_1}(t)$ along the entire trajectory. This requirement is naturally expressed in signal temporal logic (STL) format, as shown previously in (4.23), and the corresponding trajectory robustness is measured with the STL robustness degree $\rho^{\varphi}(\boldsymbol{\theta})$. Unlike the deterministic example, stochasticity leads to a distribution of possible robustness degrees at an arbitrary $\boldsymbol{\theta}$ location, illustrated in Figure 5-8. In this figure, 500 trajectories are performed at the same arbitrary location $\boldsymbol{\theta} = [6, 1.8]^T$ to demonstrate the effect of stochasticity upon the robustness measurements $y(\boldsymbol{\theta})$. Note that the distribution of measurements is roughly Gaussian, with the fitted Gaussian (red line) closely matching the histogram data.

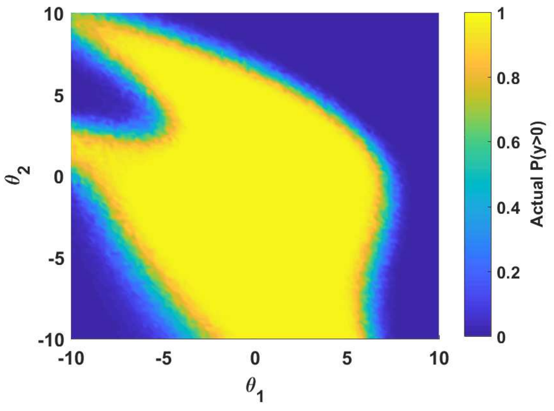
This example assumes the $y(\boldsymbol{\theta})$ measurements follow a homoscedastic Gaussian distribution with a constant standard deviation of $\epsilon_y = 0.0682$. This standard deviation ϵ_y was empirically computed from a "ground truth" dataset of 50 trajectories performed at each of the locations in Θ_d. In order to avoid any potential heteroscedastic distributions in this example, the standard deviations at each location were averaged to produce ϵ_y. Likewise, the distribution mean $\bar{y}(\boldsymbol{\theta})$ is simply the average of the 50 distributions at each $\boldsymbol{\theta} \in \Theta_d$. This true mean $\bar{y}(\boldsymbol{\theta})$ and resulting cumulative distribution $p_{sat}(\boldsymbol{\theta})$ are displayed in Figure 5-9. Interestingly enough, the mean $\bar{y}(\boldsymbol{\theta})$ does not necessarily equal the deterministic measurement value for (deterministic) $y(\boldsymbol{\theta})$ from Section 4.4.1. These deterministic measurement values are redisplayed in Figure 5-10 and highlight that the stochastic mean $\bar{y}(\boldsymbol{\theta})$ is different than the deterministic values. This disagreement can be due to a number of factors, including the nonlinearity of the adaptive law making the closed-loop system particularly susceptible to random noise. Later work in Chapter 7 will revisit this disagreement between deterministic and stochastic data, but the main takeaway from Figure 5-10 is to show that $\bar{y}(\boldsymbol{\theta})$ does not have to match the output from a deterministic system. It would be unwise to blindly assume the stochastic mean equals the deterministic measurement without actually running simulations or experiments of the closed-loop system.

The statistical verification algorithms select training sample locations $\boldsymbol{\theta}$ from a discrete sampling set Θ_d of 40,401 locations between $|\theta_1| \leq 10$ and $|\theta_2| \leq 10$. Each algorithm starts with an initial training set of 50 randomly chosen trajectories and then selects additional training locations until a budget of 450 total trajectories is reached. Figure 5-11 displays one of the initial training datasets and the resulting GP regression model and predictions. The initial GP model roughly approximates the true $\bar{y}(\boldsymbol{\theta})$ and $p_{sat}(\boldsymbol{\theta})$, but would obviously improve with additional trajectory data.

This example examines two versions of Algorithm 8 with batch sizes $M = 5$ and $M = 10$. Figure 5-12(a) plots the selection criterion, CDF variance reduction $\widetilde{V}(\boldsymbol{\theta}|\mathcal{L}, \psi, \vartheta)$, and the selected batch $\mathcal{S}$ of 5 points produced by the k-DPP in Algorithm 8. These samples are spread out across areas of large perceived reductions in CDF variance and are mostly concentrated in regions where $\widehat{p}_{sat}(\boldsymbol{\theta}) \approx 0.5$, as shown in Figure 5-12(b). Figures 5-12(c) and (d) also compare the sample set $\mathcal{S}$ and CDF variance reduction $\widetilde{V}(\boldsymbol{\theta}|\mathcal{L}, \psi, \vartheta)$ against the PDF predictive mean $\bar{y}(\boldsymbol{\theta})$ and covariance $\Sigma(\boldsymbol{\theta})$. These two plots illustrate the earlier insights in Section 5.3 that CDF



(a) Mean $\bar{y}(\boldsymbol{\theta})$



(b) Satisfaction probability $p_{sat}(\boldsymbol{\theta})$

Figure 5-9: [Example 5.6.1] True mean $\bar{y}(\boldsymbol{\theta})$ and satisfaction probability function $p_{sat}(\boldsymbol{\theta})$ for the stochastic CL-MRAC system.

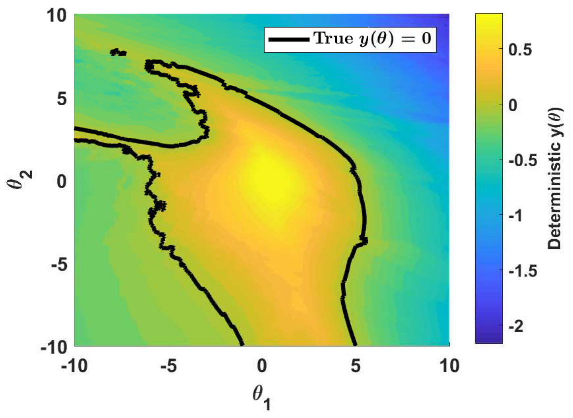
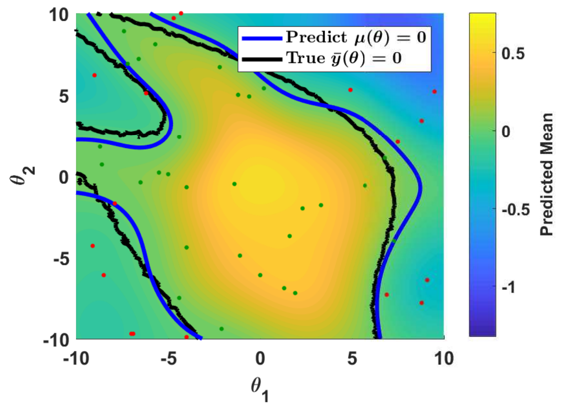


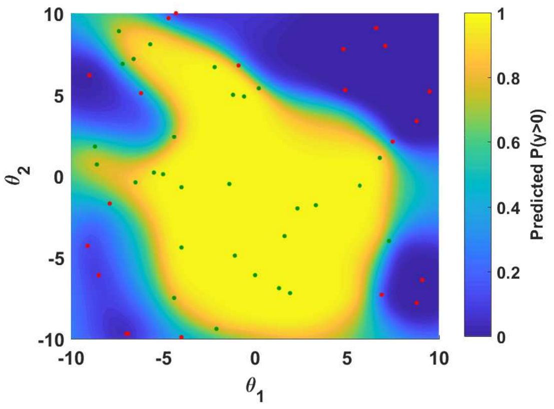
Figure 5-10: [Example 5.6.1] Deterministic measurements from Example 4.4.1 for comparison to the mean $\bar{y}(\boldsymbol{\theta})$ of the noisy measurements in this stochastic variation of the problem. Notice that the deterministic measurements do not necessarily equal the stochastic mean $\bar{y}(\boldsymbol{\theta})$.

variance reduction (and CDF variance itself) is a function of both $\mu(\boldsymbol{\theta})$ and $\Sigma(\boldsymbol{\theta})$. The expected CDF variance reduction is high in regions with $\mu(\boldsymbol{\theta}) \approx 0$, but not all areas with $\mu(\boldsymbol{\theta}) = 0$ are weighted equally. Likewise, all the areas of high $\widetilde{V}(\boldsymbol{\theta}|\mathcal{L}, \psi, \vartheta)$ correspond to regions of moderate-to-high PDF variance $\Sigma(\boldsymbol{\theta})$, but not all points with high $\Sigma(\boldsymbol{\theta})$ have a large expected reduction in CDF variance.

The sampling process will continue to select parameter settings for future trajectories until the sampling budget is reached. An intermediate result halfway through the process after 250 samples is shown in Figure 5-13. The GP prediction for mean



(a) Predicted mean $\mu(\boldsymbol{\theta})$



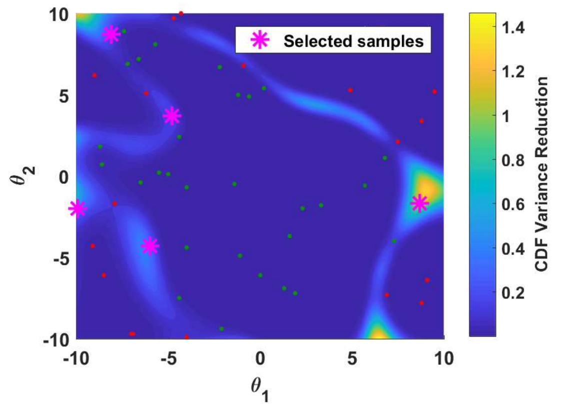
(b) Predicted satisfaction probability $\widehat{p}_{sat}(\boldsymbol{\theta})$

Figure 5-11: [Example 5.6.1] Predicted satisfaction probability function $\widehat{p}_{sat}(\boldsymbol{\theta})$ for the CL-MRAC systems at the initial training step.

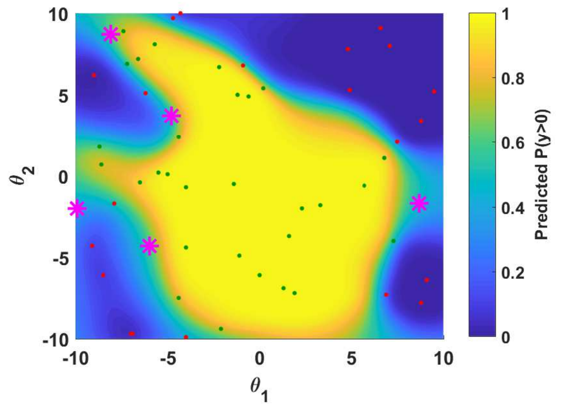
$\bar{y}(\boldsymbol{\theta})$ and the estimated $\widehat{p}_{sat}(\boldsymbol{\theta})$ are already fairly accurate approximations of the true values from Figure 5-9. The chosen training samples (red and green dots) are clumped in close proximity to the region of $\widehat{p}_{sat}(\boldsymbol{\theta}) \approx 0.5$ as this is where the predictions are expected to change the most. The final prediction model after all 450 trajectories have been selected is shown in Figure 5-14 and is almost identical to the intermediate prediction model in Figure 5-13.

Figure 5-15 compares the performance of Algorithm 8 (labeled "CDF Variance") against competing active sampling strategies using the PDF mean-focused selection metric (5.20) and PDF variance-focused metric (5.21), as well as an open-loop strategy using random sampling. Unlike the earlier figures in Chapters 3 and 4, these figures measure the prediction accuracy with mean absolute error (MAE) between predicted $\widehat{p}_{sat}(\boldsymbol{\theta})$ and true $p_{sat}(\boldsymbol{\theta})$ over all $\boldsymbol{\theta} \in \Theta_d$. The figure displays the mean and 0.5σ error bounds for MAE over 100 random initializations of each algorithm. At the conclusion of the process after 450 simulations have been performed, Algorithm 8 with batch size $M = 5$ demonstrates a 23% improvement in average MAE over the closest competitor (PDF mean), a 24% improvement over PDF variance-based sampling, and 27% improvement over open-loop, random sampling. Similar results are seen for the larger batch size $M = 10$.

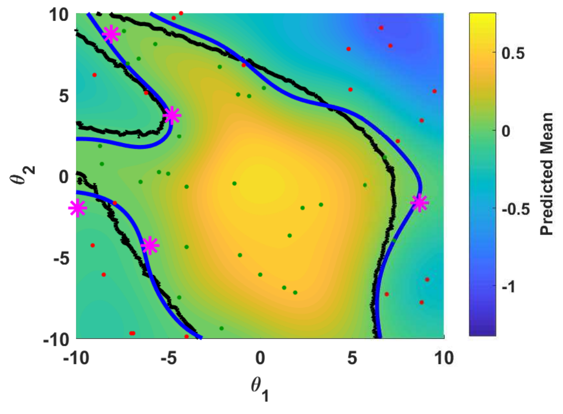
Additionally, Algorithm 8 still outperforms the other algorithms when the true hy-



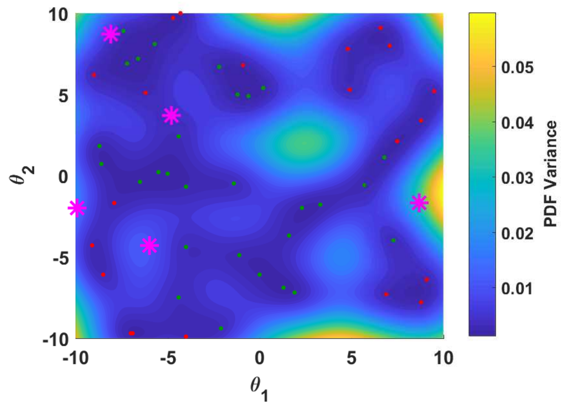
(a) CDF variance reduction $\widetilde{V}(\boldsymbol{\theta}|\mathcal{L}, \psi, \vartheta)$



(b) Predicted $\widehat{p}_{sat}(\boldsymbol{\theta})$



(c) PDF mean $\mu(\boldsymbol{\theta})$

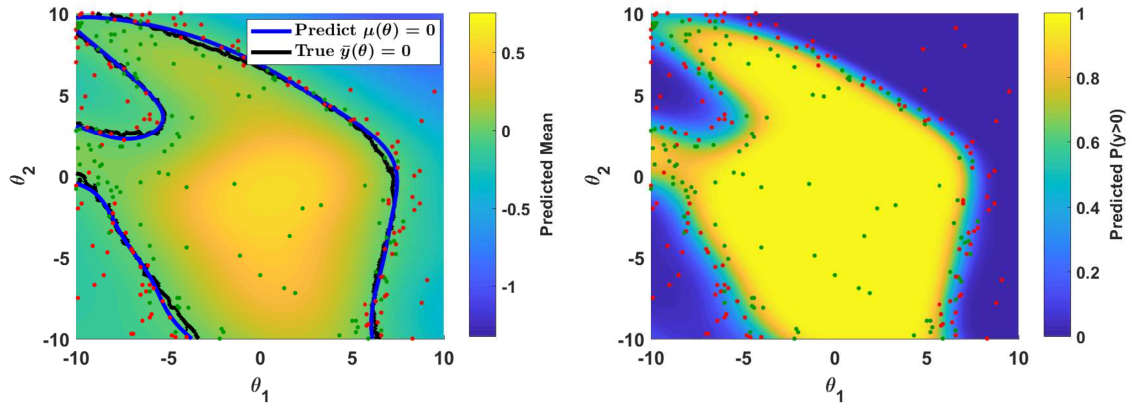


(d) PDF variance $\Sigma(\boldsymbol{\theta})$

Figure 5-12: [Example 5.6.1] Illustration of the CDF variance selection criterion and the chosen set of future training locations. For comparison to other selection metrics, the selected points are overlayed on top of $\bar{y}(\boldsymbol{\theta})$, $\widehat{p}_{sat}(\boldsymbol{\theta})$, and $\Sigma(\boldsymbol{\theta})$.

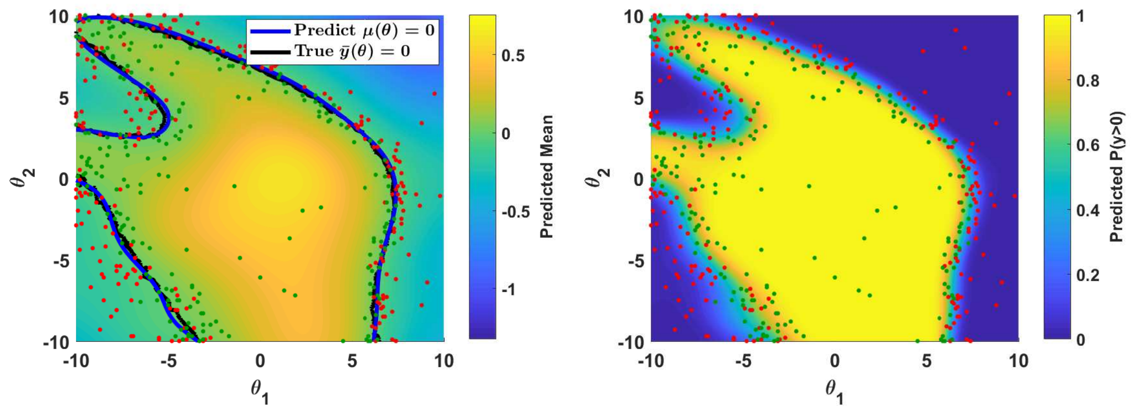
perparameters were already known. The results in Figure 5-15 considered the usual case where the true hyperparameters are not known and (ψ, ϑ) are estimated online using hyperparameter optimization. Figure 5-16 considers the same verification algorithms, but assumes the true hyperparameters are known in advance and (ψ, ϑ) are fixed to those values. Even with this advanced knowledge, Algorithm 8 demonstrates a comparable level of improvement over the existing sampling strategies.

Although Figures 5-15 and 5-16 illustrated a 20-27% improvement in average MAE rate over existing sampling strategies, there is still a distribution associated with the MAE convergence. There is no guarantee Algorithm 8 and the other approaches will



(a) Predicted mean $\mu(\boldsymbol{\theta})$ (b) Predicted satisfaction probability $\widehat{p}_{sat}(\boldsymbol{\theta})$

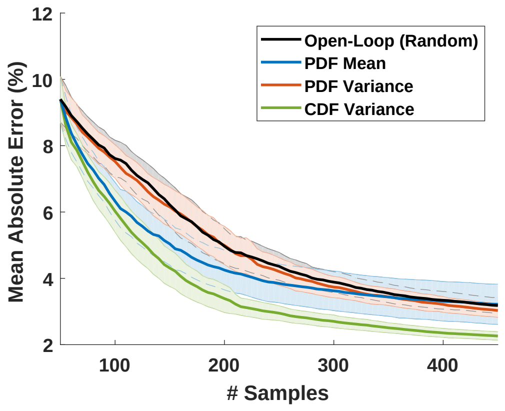
Figure 5-13: [Example 5.6.1] Predicted satisfaction probability function $\widehat{p}_{sat}(\boldsymbol{\theta})$ halfway through the verification process after 250 trajectories have been selected.



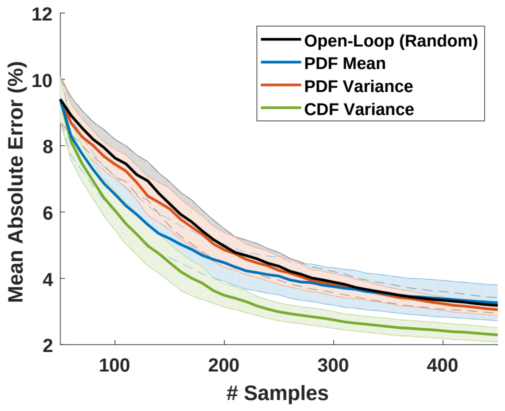
(a) Predicted mean $\mu(\boldsymbol{\theta})$ (b) Predicted satisfaction probability $\widehat{p}_{sat}(\boldsymbol{\theta})$

Figure 5-14: [Example 5.6.1] Predicted satisfaction probability function $\widehat{p}_{sat}(\boldsymbol{\theta})$ at the end of the verification process after all 450 trajectories have been selected.

always achieve that level of MAE, particularly since the closed-loop system is inherently random. In order to address whether Algorithm 8 is consistently better than the existing approaches, the four sampling procedures are started from the same random initialization and with the same noisy measurements for each of the 100 repeated test runs. This allows for the approaches to be directly compared against one another as they all have the same exact initial $\mathcal{L}$ and noisy data. Figure 5-17 demonstrates that at the completion of the 450 trajectory samples, Algorithm 8 directly outperforms or matches the MAE rate of all the competing approaches roughly 92-100% of the

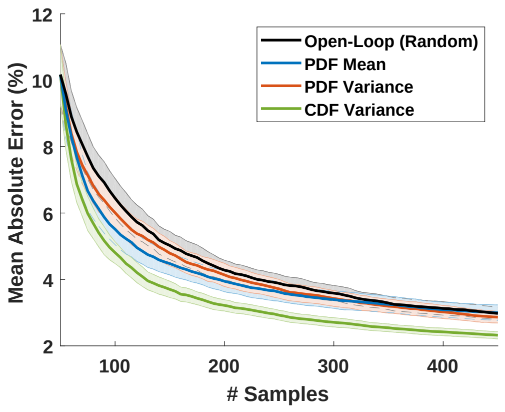


(a) Batch size $M = 5$

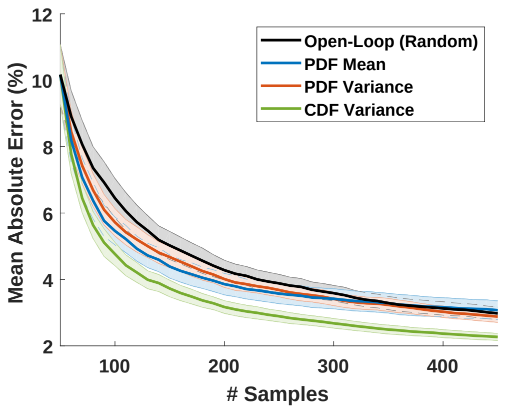


(b) Batch size $M = 10$

Figure 5-15: [Example 5.6.1] Mean absolute error (MAE) convergence of Algorithm 8 in comparison to the other sampling strategies. In this case, the hyperparameters are dynamically updated with hyperparameter optimization after each iteration. The standard deviation intervals around the mean (solid lines) are given by the 0.5σ bound.



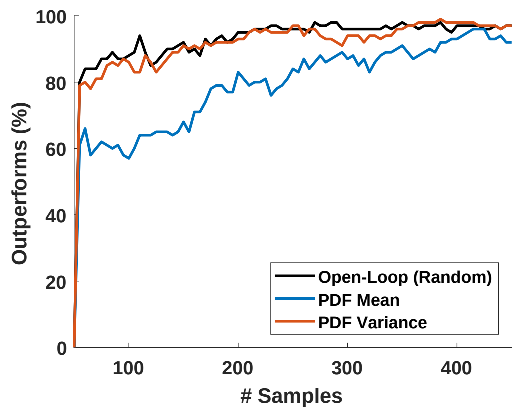
(a) Batch size $M = 5$



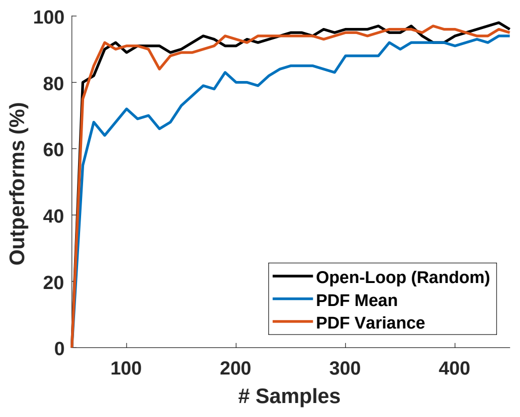
(b) Batch size $M = 10$

Figure 5-16: [Example 5.6.1] Mean absolute error (MAE) convergence for the different sampling strategies with known (static) hyperparameters. The standard deviation intervals around the mean (solid lines) are given by the 0.5σ bound.

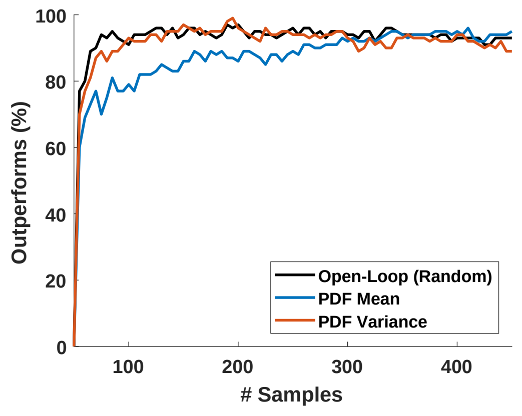
time, regardless of the batch size and whether the true hyperparameters are known or not. Therefore, Algorithm 8 consistently produces the lowest rate of mean absolute prediction error.



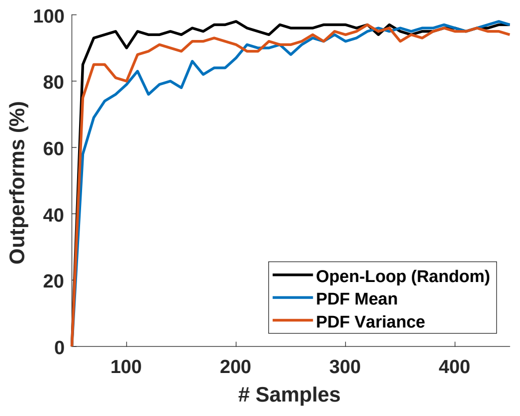
(a) Hyperparameter opt., $M = 5$



(b) Hyperparameter opt., $M = 10$



(c) Static hyperparameters, $M = 5$



(d) Static hyperparameters, $M = 10$

Figure 5-17: [Example 5.6.1] Ratio of runs where Algorithm 8 directly outperforms or matches the mean absolute error rate of the competing sampling strategies. All strategies start with the same initial training set and noisy measurements and thus each approach can be directly compared to the others with the same initialization.

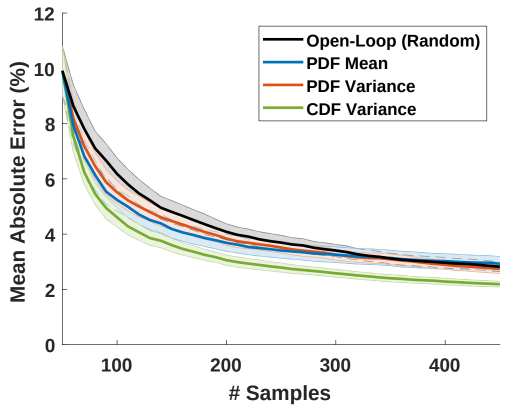
CDF Variance to Identify Regions of High Prediction Error

While Figures 5-15 and 5-16 examine the mean absolute prediction error for the various sampling strategies, these values are not known during the actual verification process because $p_{sat}(\boldsymbol{\theta})$ is unknown. In order to address this limitation, Section 5.2.2 presented a novel method for online quantification of prediction accuracy using CDF variance. CDF variance bounds the Chebyshev inequality, but more practically, identifies regions in Θ where new training samples could possibly induce a large change in $\widehat{p}_{sat}(\boldsymbol{\theta})$. Although the true CDF variance has no closed-form solution, the approximations in (5.18) and (5.19) are still useful tools for indicating where CDF variance is high and therefore where confidence in the predicted $\widehat{p}_{sat}(\boldsymbol{\theta})$ is low.

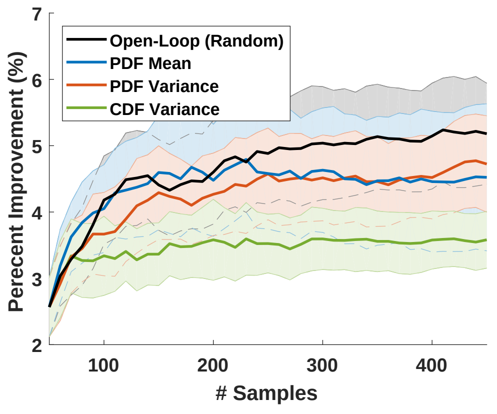
Figures 5-18 and 5-19 demonstrate the use of CDF variance to identify areas of likely prediction error. Additionally, these results show CDF variance will correctly identify these areas independent of the actual sampling strategy. It does not matter whether trajectories are selected using Algorithm 8 or the other three strategies. In each of the four strategies, all $\boldsymbol{\theta} \in \Theta_d$ are sorted according to their approximate CDF variance from (5.18) and the points with the top 1% and 5% of CDF variance are removed. Figures 5-18 and 5-19 then show the recomputed MAE values for all four sampling strategies with batch size $M = 10$ and static hyperparameters after the top 1% and 5% have been removed. In comparison to the original plot in Figure 5-16(b), the average MAE reduces by up to 14-20% when the top 5% of points are removed, illustrating the concentration of points with high prediction error within those points with the top 5% of CDF variance. Although it does not explicitly quantify prediction error as (4.9) did in Chapter 4, approximate CDF variance (5.18) does identify regions where large prediction error $\widetilde{p}_{sat}(\boldsymbol{\theta})$ is likely to occur.

Effect of Changes in the Distribution of $y(\boldsymbol{\theta})$

Lastly, Figures 5-20, 5-21, and 5-22 examine the performance of the sampling algorithms after changes in the distribution of $y(\boldsymbol{\theta})$. These figures consider changes in the process noise $\mathbf{w}(t)$ of the stochastic dynamics, which ultimately manifest as changes in the distribution of noisy robustness measurements $y(\boldsymbol{\theta})$. When the process noise covariance matrix Σ_w increases or decreases, the distribution of measurements and the resulting satisfaction probability $p_{sat}(\boldsymbol{\theta})$ will vary. Figure 5-20 depicts the changes to the shape of the true $p_{sat}(\boldsymbol{\theta})$ surface as a result of varying the level of process noise. When the process noise decreases, the average $\bar{y}(\boldsymbol{\theta})$ remains roughly the same, but the standard deviation ϵ_y decreases and causes a sharper gradient in $p_{sat}(\boldsymbol{\theta})$ at points near $p_{sat}(\boldsymbol{\theta}) \approx 0.5$. Likewise, when the process noise is increased, the standard deviation also increases and leads to a more gradual slope in Figure 5-20(b). These changes in $p_{sat}(\boldsymbol{\theta})$ will affect the performance of the sampling approaches.

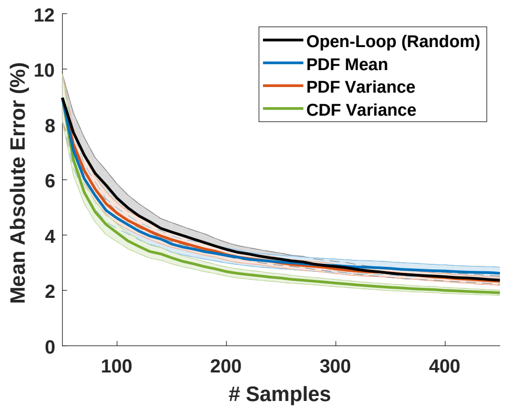


(a) MAE without points in the top 1% of CDF variance

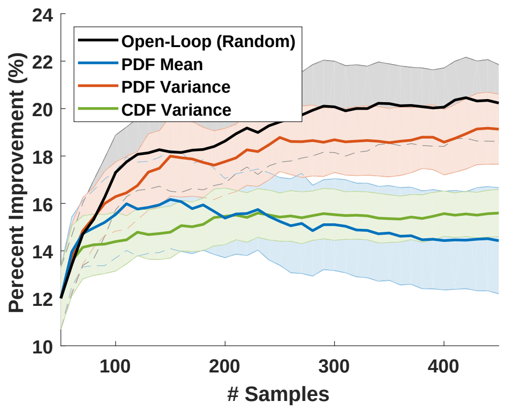


(b) Percent reduction vs Figure 5-16(b)

Figure 5-18: [Example 5.6.1] Concentration of prediction error in points with the top 1% of CDF variance. These plots illustrate the reduction in MAE after the points with the top 1% of CDF variance have been removed.



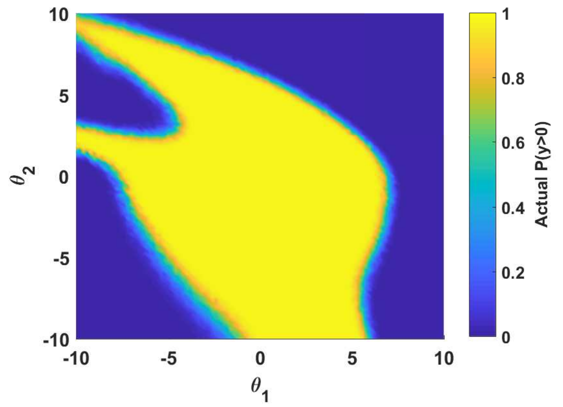
(a) MAE without points in the top 5% of CDF variance



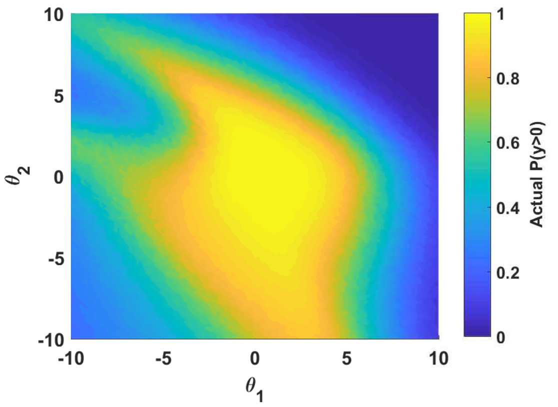
(b) Percent reduction vs Figure 5-16(b)

Figure 5-19: [Example 5.6.1] Concentration of prediction error in points with the top 5% of CDF variance. These plots illustrate the reduction in MAE after the points with the top 5% of CDF variance have been removed.

When the process noise decreases and ϵ_y shrinks, the distribution of $y(\boldsymbol{\theta})$ across Θ resembles a binary problem with large areas where $p_{sat}(\boldsymbol{\theta}) \approx 0$ or $p_{sat}(\boldsymbol{\theta}) \approx 1$ and very few points where $p_{sat}(\boldsymbol{\theta}) \approx 0.5$. Figure 5-21 plots the average MAE convergence of the four sampling algorithms for this new level of ϵ_y. The performance of each of the algorithms is roughly the same as it was for the nominal problem in Figures 5-15(b) and 5-16(b). Algorithm 8 outperforms the other approaches with between a 29-35%



(a) True $p_{sat}(\boldsymbol{\theta})$ with low ϵ_y

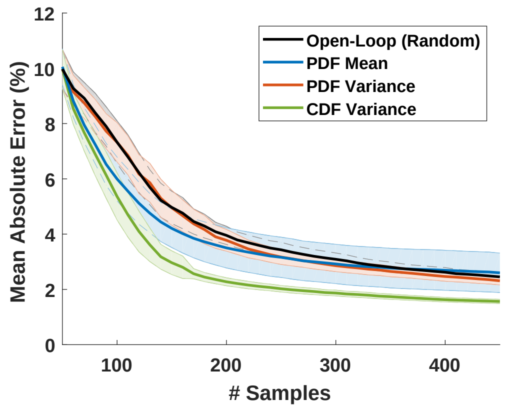


(b) True $p_{sat}(\boldsymbol{\theta})$ with high ϵ_y

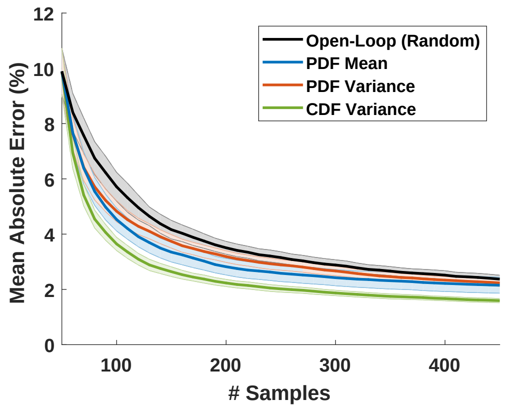
Figure 5-20: [Example 5.6.1] Changes in $p_{sat}(\boldsymbol{\theta})$ associated with increases and decreases in process noise. The change in process noise ultimately results in changes to the distribution of $y(\boldsymbol{\theta})$ which defines $p_{sat}(\boldsymbol{\theta})$.

improvement in average MAE for the case with online hyperparameter optimization and a 22-32% improvement for the case with static hyperparameters.

As the process noise increases and standard deviation ϵ_y grows, the distribution is much wider and the gradient of $p_{sat}(\boldsymbol{\theta})$ is much lower. This wider distribution means a higher ratio of points in Θ will be located in the regions around $p_{sat}(\boldsymbol{\theta}) \approx 0.5$ where prediction error is likely to be high. In Figure 5-22, the average MAE starts higher than before in Figure 5-21 due to the difficulty in modeling $p_{sat}(\boldsymbol{\theta})$. In these plots, the PDF mean-focused sampling strategy has noticeable difficulty with $\widetilde{p}_{sat}(\boldsymbol{\theta})$ and its average MAE converges much more slowly than it did before. This degraded performance is due to the effect of larger ϵ_y on the cumulative distribution for $\mathbb{P}(y(\boldsymbol{\theta}) > 0)$. Points further away from $\bar{y}(\boldsymbol{\theta})$ will have more sizable cumulative distributions for $\mathbb{P}(y(\boldsymbol{\theta}) > 0)$ and therefore clumping samples around $\mu(\boldsymbol{\theta}) \approx 0$ will have less of a positive effect as it did when ϵ_y was small. Likewise, the large ϵ_y term also causes the information gain associated with each additional sample to decrease and explains the similarity in performance between PDF variance, open-loop sampling, and Algorithm 8. Due to the high standard deviation ϵ_y, the $\epsilon_y^2\mathbf{I}$ term in the inverted matrix from (5.8) will lessen the impact of each update. As each additional sample will have a significantly reduced effect upon the predictions, $\Sigma(\boldsymbol{\theta})$ will not decrease

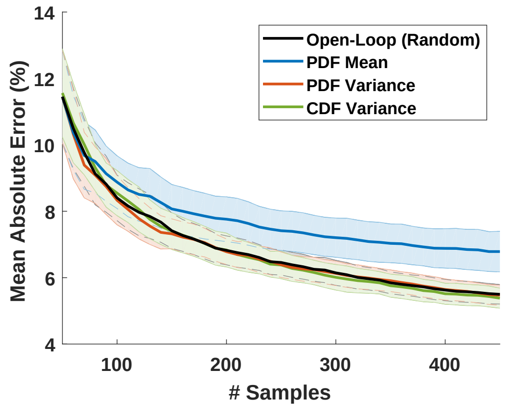


(a) Hyperparameter opt.

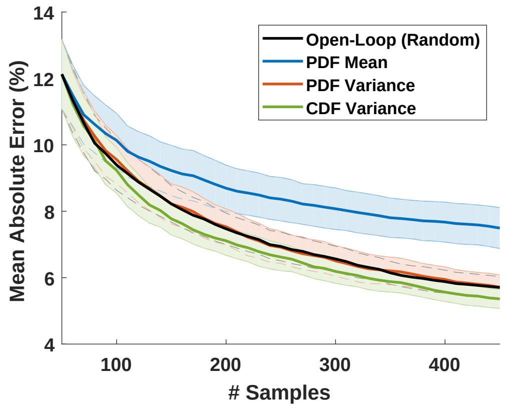


(b) Static hyperparameters

Figure 5-21: [Example 5.6.1] Mean absolute error convergence of the four sampling strategies ($M = 10$) when the process noise has been reduced to $\Sigma_w = \text{diag}([1, 1])$.



(a) Hyperparameter opt.



(b) Static hyperparameters

Figure 5-22: [Example 5.6.1] Mean absolute error convergence of the four sampling strategies ($M = 10$) when the process noise has been increased to $\Sigma_w = \text{diag}([7, 7])$.

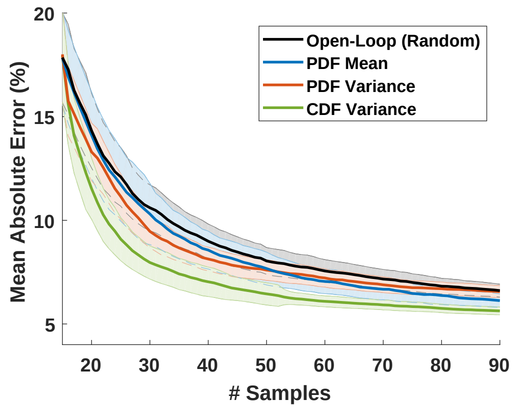
as much and the variance will be comparable across large portions of Θ - almost like a uniform random distribution. This decrease in the posterior change of $\Sigma(\boldsymbol{\theta})$ with each additional sample will also lessen the expected posterior CDF variance reduction and degrades Algorithm 8's edge over the other approaches. The main takeaway from Figures 5-21 and 5-22 is that Algorithm 8 will often outperform the existing sampling strategies, but will do no less in worst-case scenarios. This emphasizes the better *all-around* performance of Algorithm 8 for stochastic verification purposes.

5.6.2 Robust Multi-Agent Task Allocation

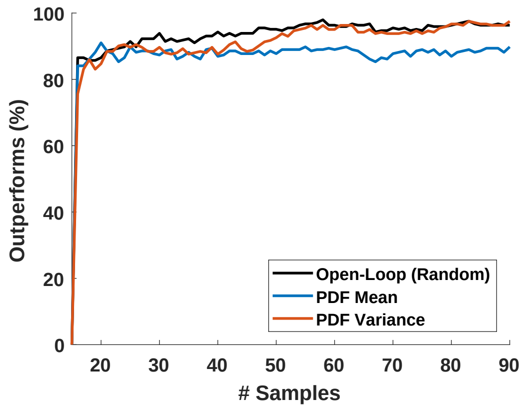
The second example examines the same robust multi-agent task allocation problem from Section 4.4.2 subjected to additional stochasticity. In this stochastic version of the problem, the time it takes a UAV to complete a surveillance task is still a nonlinear functions of wind parameters $\boldsymbol{\theta} = [\theta_1, \theta_2]^T$, but the task duration is also corrupted by zero-mean Gaussian noise which may cause a task to take longer or shorter than planned even if wind parameters $\boldsymbol{\theta}$ are known. The uncertainty and stochasticity in the task durations will have a cumulative effect upon the realized mission score as longer-than-planned task durations will cause the UAVs to miss the completion of tasks within their assigned window and performance worse than expected.

As before, the verification goal is to determine whether multi-agent system will reach a minimum score threshold across all the possible wind conditions given the assigned control policy, the ordered list of tasks assigned to each UAV. The set of feasible wind conditions, $\theta_1 : [0°, 359°]$ and $\theta_2 : [0, 40]$ (km/hr), is covered with a lattice Θ_d of 16,641 possible parameter settings for the trajectories. Each training point consists of a trajectory rollout of the multi-agent system in the forest fire simulation model at the given wind conditions and the trajectory rollout's realized mission score. Just like the previous example, the simulation results will also explore the effect of different levels of Gaussian noise in the system.

Figures 5-23 and 5-24 compare the sequential active sampling approach (Algorithm 7) against the other sampling strategies over 250 random initializations. In both figures, Algorithm 7 performs as well as, if not better than, the other algorithms the vast majority of the time. As was seen in Example 5.6.1, Algorithm 7's improvement over the other strategies is higher when the standard deviation ϵ_y is lower, but is still fine when ϵ_y increases. Ultimately the CDF variance reduction procedure in Algorithm 7 consistently performs well whereas the other algorithms' rate of prediction error shifts depending on the distribution. Since the standard deviation of the distribution will generally be unknown in advance, these results reinforce the notion that CDF variance-based selection criteria presents the best all-around option for closed-loop

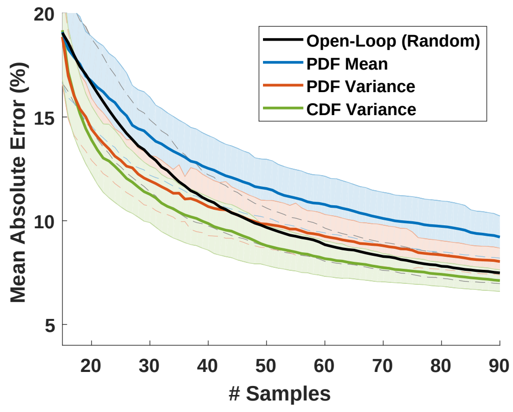


(a) MAE convergence

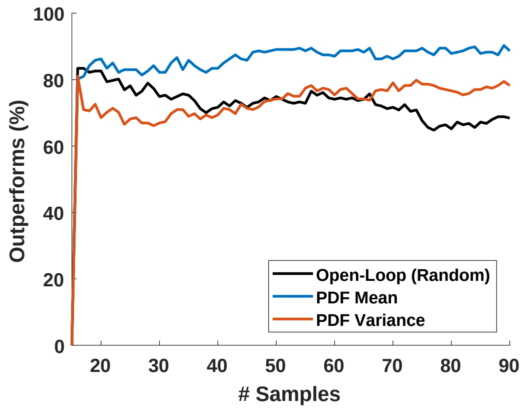


(b) Ratio of tests where Algorithm 7 outperforms or matches existing sampling strategies

Figure 5-23: [Example 5.6.2] Comparison of mean absolute error (MAE) performance with low measurement variance ϵ_y^2 over 250 random initializations of the sampling strategies. The right-hand plot displays the ratio of these randomly-initialized runs where Algorithm 7 directly outperforms or matches the other sampling strategies.



(a) MAE convergence

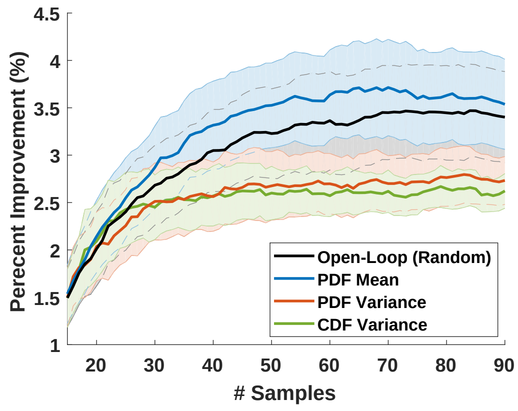


(b) Ratio of tests where Algorithm 7 outperforms or matches existing sampling strategies

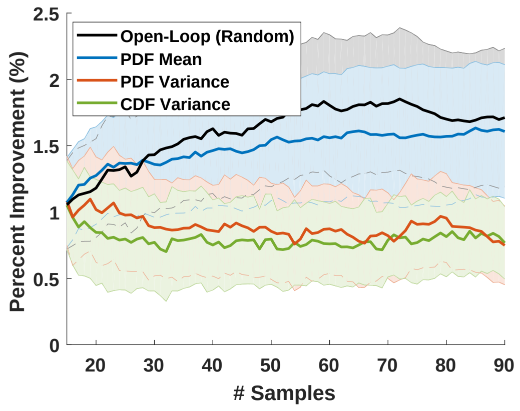
Figure 5-24: [Example 5.6.2] Comparison of mean absolute error (MAE) performance with high measurement variance ϵ_y^2 over 250 random initializations of the sampling strategies. The right-hand plot displays the ratio of these randomly-initialized runs where Algorithm 7 directly outperforms or matches the other sampling strategies.

statistical verification.

Lastly, Figures 5-25 and 5-26 examine the use of CDF variance to identify regions of likely prediction error. Both figures consider the low-variance case from Figure 5-23 and recompute the MAE after points with high CDF variance have been removed. In

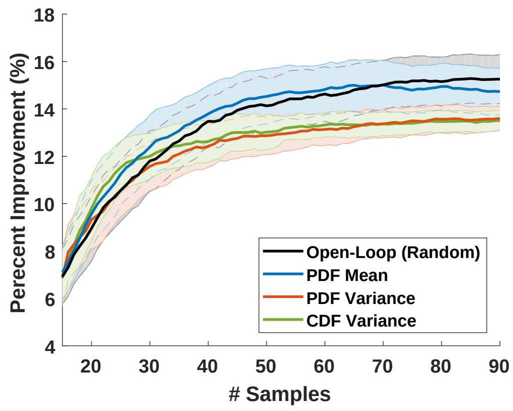


(a) Percent reduction vs Figure 5-23(a)

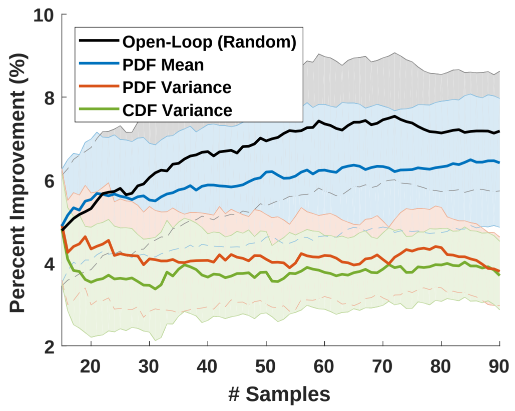


(b) Percent reduction vs Figure 5-24(a)

Figure 5-25: [Example 5.6.2] Concentration of prediction error in points with the top 1% of CDF variance. These plots illustrate the reduction in MAE after the points with the top 1% of CDF variance have been removed.



(a) Percent reduction vs Figure 5-23(a)



(b) Percent reduction vs Figure 5-24(a)

Figure 5-26: [Example 5.6.2] Concentration of prediction error in points with the top 5% of CDF variance. These plots illustrate the reduction in MAE after the points with the top 5% of CDF variance have been removed.

Figure 5-25 after the points with the top 1% of CDF variance have been removed, the MAE drops by 1-2% and even further by 4-7% in Figure 5-26 after the top 5% have been removed. These reductions in MAE again demonstrate the value of approximate CDF variance (5.18) for online identification of points with likely high prediction error, irrespective of the exact sampling strategy.

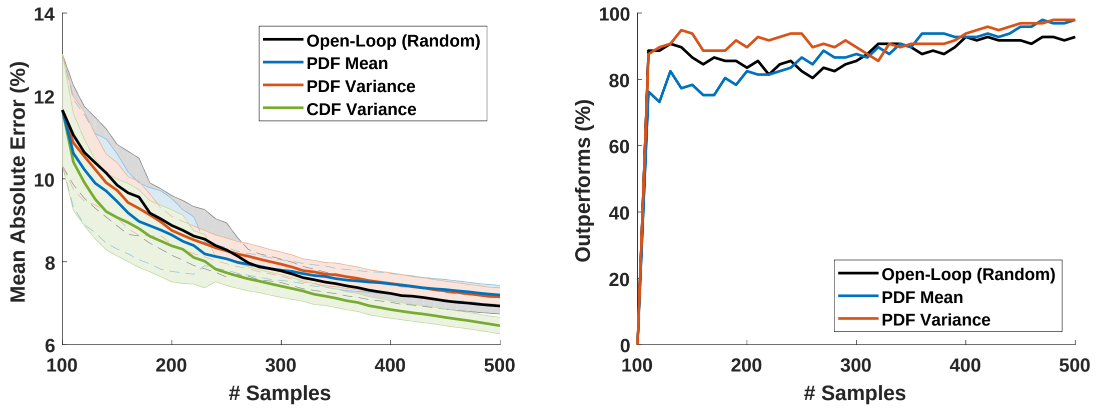
5.6.3 Lateral-Directional Autopilot

The last example adds a stochastic wind field to the lateral-directional autopilot from Section 4.4.4. In particular, the open-loop aircraft dynamics include a stochastic Dryden wind field model [15], an aerospace standard for modeling random wind turbulence. The wind turbulence model was initialized with a turbulence scale length of 533.4 meters and a high-altitude intensity exceedance probability of 10^{-1}, common settings for low-altitude, light wind turbulence. The rest of the aircraft simulation model and control system remain unchanged.

The dominating performance requirement for the heading-hold autopilot is still the altitude-hold requirement and the nominal specification remains unchanged: the aircraft's altitude must remain within 35 feet of the initial altitude at every point along the trajectory [133,134]. Unlike Example 4.4.4, this stochastic example will also explore the effect of loosening the 35 foot window as it drastically changes the $p_{sat}(\boldsymbol{\theta})$ function and the performance of the active sampling strategies. The satisfaction of the altitude-hold performance requirement is tested against the four uncertain initial conditions $\boldsymbol{\theta} = [\theta_1, \theta_2, \theta_3, \theta_4]^T$ corresponding to the Euler angles for roll, pitch, and yaw and the longitudinal moment of inertia I_{yy}. The sampling grid Θ_d consists of 937,692 points that span $\theta_1 : [-60°, 60°]$, $\theta_2 : [4°, 19°]$, heading $\theta_3 : [75°, 145°]$, and inertia $\theta_4 : [5430, 8430](kg \cdot m^2)$, with a desired heading angle of 112°.

The first part of this example assumes the STL robustness measurements $y(\boldsymbol{\theta})$ follow a homoscedastic Gaussian distribution with a constant standard deviation ϵ_y. Each sample location in Θ_d was tested offline 200 times to determine the empirical mean $\bar{y}(\boldsymbol{\theta})$ and standard deviation $\epsilon_y(\boldsymbol{\theta})$ of a Gaussian distribution fit to that data. For the homoscedastic problem, the spatially-independent ϵ_y was created by averaging $\epsilon_y(\boldsymbol{\theta})$ across all $\boldsymbol{\theta} \in \Theta_d$. This will be relaxed to heteroscedastic distributions during the second portion of the example.

Figures 5-27 to 5-29 compare the four sampling strategies for different distributions of trajectory robustness measurements over 100 random initializations. First, Figure 5-27 demonstrates the performance of Algorithm 8 on the standard problem



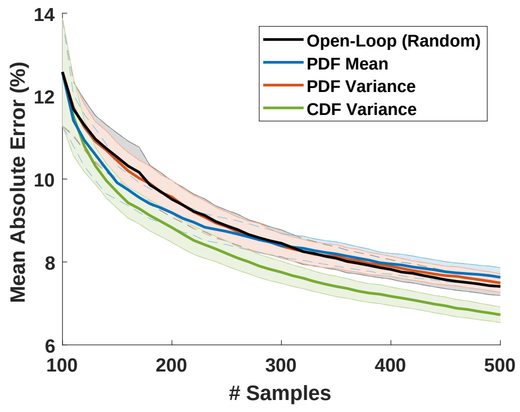
(a) MAE convergence

(b) Ratio of tests where Algorithm 8 outperforms or matches existing sampling strategies

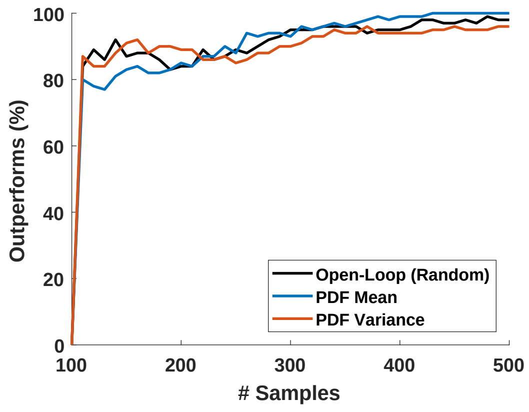
Figure 5-27: [Example 5.6.3] Comparison of mean absolute error (MAE) convergence of the four sampling strategies assuming the standard measurement distribution.

with the original performance requirement. Although Figure 5-27(b) indicates that Algorithm 8 meets or exceeds the performance of the competing active sampling algorithms at least 80% of the time, the actual percent improvement is rather low (7-9%). If the altitude-hold requirement is relaxed by 10 feet, the mean $\bar{y}(\boldsymbol{\theta})$ is shifted and Algorithm 8's improvement over the other three approaches becomes more pronounced, now up to 10-12% at the conclusion of the procedure. If this altitude requirement is relaxed even further to 20 feet, the distribution of $y(\boldsymbol{\theta})$ has changed rather drastically and Algorithm 8 demonstrates a clear 26% improvement over random and PDF variance-based sampling and 8% boost over the PDF mean-focused metric. While the merits of changing the requirement are debatable, this study is meant to highlight the changes in algorithm performance due to the underlying distribution of the data. As was seen in the last two examples, Algorithm 8 is consistently the best sampling strategy regardless of the actual distribution of the data. This is particularly important when nothing is known in advance about the performance of the stochastic nonlinear system.

Figure 5-30 further reiterates the use of CDF variance as a tool for online identification of areas of high prediction error. Since the actual MAE is unknown during an actual testing scenario, the CDF variance is the only practical method to rank

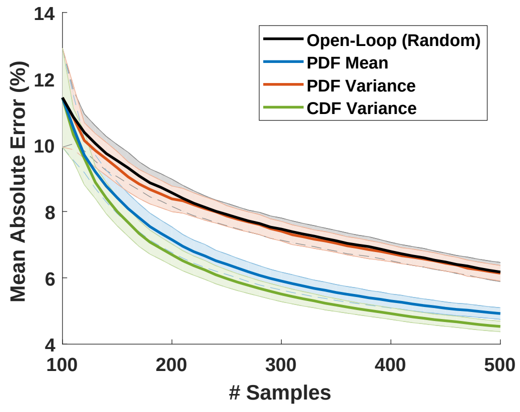


(a) MAE convergence

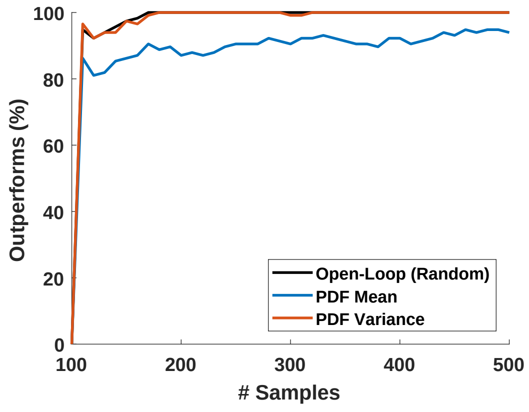


(b) Ratio of tests where Algorithm 8 outperforms or matches existing sampling strategies

Figure 5-28: [Example 5.6.3] Comparison of mean absolute error (MAE) convergence of the four sampling strategies after the requirement is loosened by 10 feet.

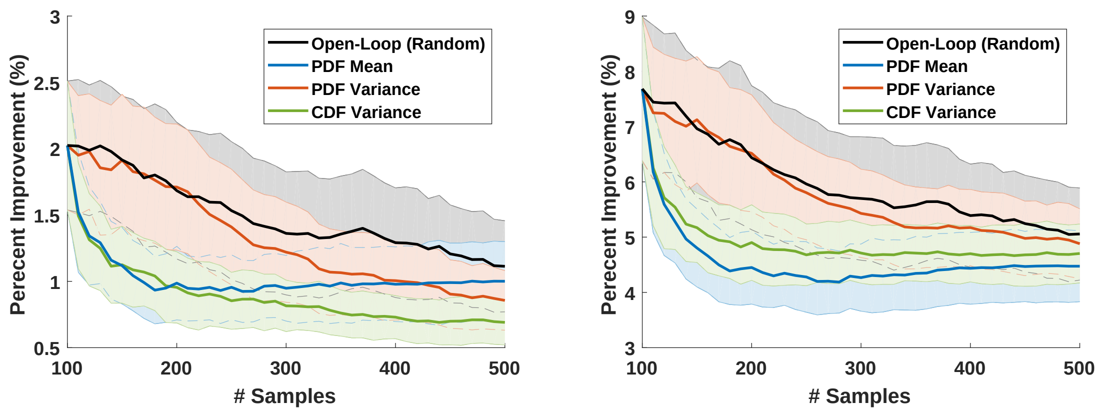


(a) MAE convergence



(b) Ratio of tests where Algorithm 8 outperforms or matches existing sampling strategies

Figure 5-29: [Example 5.6.3] Comparison of mean absolute error (MAE) convergence of the four sampling strategies after the requirement is loosened by 20 feet.



(a) Percent reduction after top 1% removed (b) Percent reduction after top 5% removed

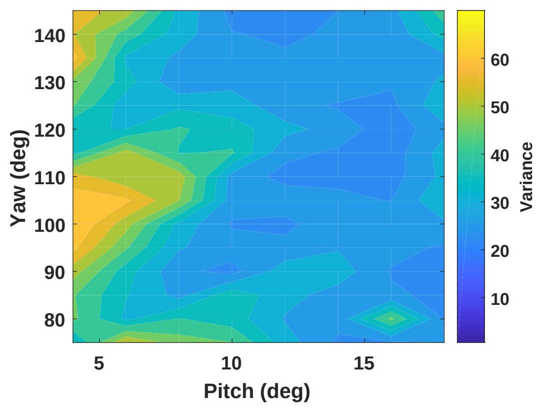
Figure 5-30: [Example 5.6.3] Concentration of prediction error in points with the top 1-5% of CDF variance. These plots illustrate the reduction in MAE after the points with the top 1-5% of CDF variance have been removed.

regions in Θ_d where confidence in the prediction accuracy is low. The two plots in this figure demonstrate the same exact observations as before; when the points with the top 1% and 5% of CDF variance are removed, the recomputed MAE experiences a noticeable improvement in prediction accuracy, meaning most of the points with high prediction error were removed.

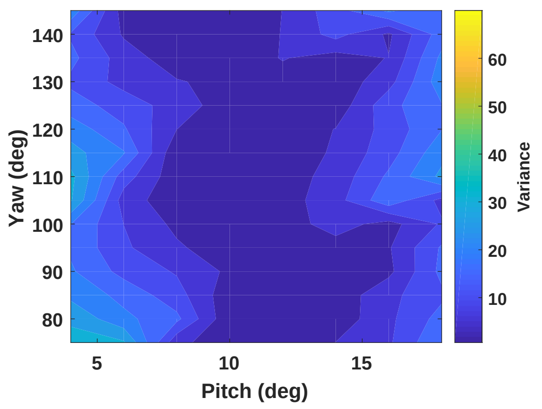
Effects of Heteroscedastic Distributions

The stochastic lateral-directional autopilot is also a good example to highlight the effects of heteroscedastic distributions with spatially-varying $\epsilon_y(\boldsymbol{\theta})$. The previous figures averaged standard deviation ϵ_y over all $\boldsymbol{\theta} \in \Theta_d$; however, the true ϵ_y will vary across Θ_d. Figure 5-31 demonstrates the large changes in variance $\epsilon_y^2(\boldsymbol{\theta})$ given different parametric uncertainties. These variance levels correspond to the original requirement with the 35 foot altitude window.

The real danger of the large changes in variance is how they will impact the predictions if ϵ_y is assumed to be constant across Θ_d. At least in this example, heteroscedastic noise will not cause the baseline Gaussian process prediction model to completely breakdown and fail to return any predictions, but it will negatively affect the predictions. Figure 5-32 illustrates the increase in mean absolute error when the



(a) 2D snapshot of variance at roll(0) = -60°, $I_{yy} = 6930 (kg \cdot m^2)$



(b) 2D snapshot of variance at roll(0) = +24°, $I_{yy} = 6930 (kg \cdot m^2)$

Figure 5-31: [Example 5.6.3] Changes in variance $\epsilon_y^2(\boldsymbol{\theta})$ across Θ_d as the result of a heteroscedastic Gaussian distribution. As initial roll angle varies, the variance $\epsilon_y^2(\boldsymbol{\theta})$ also changes. These two figures are only snapshots of the changes across Θ_d, but highlight the largest disparity in the values.

baseline homoscedastic GP from (5.8) is inadvertently applied to heteroscedastic distributions. In comparison to the homoscedastic version of the problem in Figure 5-27, the four sampling algorithms have a noticeable increase in MAE levels. For the CDF variance- and PDF mean-focused sampling procedures, the new values correspond to a 44% and 46% increase over the results in Figure 5-27. The increase in MAE is even higher for the passive and PDF variance-based procedures; they have a 93% and 113% increase over the MAE levels in the homoscedastic problem. These results highlight the need for careful consideration of whether Assumption 5.3 and the standard GP (5.8) applies or whether the heteroscedastic GP discussed in Section 5.4 should be used instead.

The use of heteroscedastic GP prediction models drastically improves some of the prediction errors. Unfortunately, the variational HGP prediction method [138, 142] presented in Section 5.4.1 is both very sensitive and expensive to train. In fact, the heteroscedastic Gaussian process model will not work at all for random sampling algorithms and the PDF mean-focused active sampling process. It will only return a numerically stable result when training data is collected with the PDF variance-based selection criteria or the new CDF variance-based criteria. These numerical instabili-

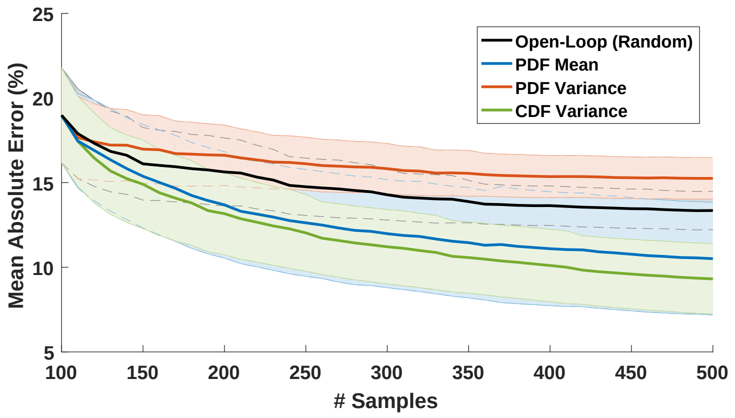


Figure 5-32: [Example 5.6.3] Degradation in mean absolute error if the baseline homoscedastic GP from (5.8) is applied to a heteroscedastic distribution.

ties highlight the need for more effective and numerically stable inference methods for spatially-varying ϵ_y, but that is beyond the scope of this thesis. Despite the trouble with numerical stability, the results that did complete do indicate the potential of heteroscedastic GPs to improve prediction errors. More specifically, closed-loop verification with heteroscedastic GPs drastically improves the MAE convergence for the PDF variance-based approach. Figure 5-33 demonstrates a 41% reduction in mean absolute error when closed-loop verification uses a heteroscedastic GP model versus a homoscedastic GP regression model. Although numerical instabilities and sensitivities prevent a full comparison of the various approaches, this discussion does indicate the challenges posed by spatially-varying $\epsilon_y(\boldsymbol{\theta})$ and modeling those distributions. Future work should more carefully examine different modeling techniques for improved inference given non-uniform Gaussian distributions.

5.7 Summary

This chapter developed stochastic extensions of the statistical verification approaches in Chapter 4 to address the open problem of verification in stochastic nonlinear systems. The chapter presented three main contributions for stochastic verification. First, Section 5.2.1 modified GP-based prediction models to handle noisy measurements and predict the satisfaction probability function $p_{sat}(\boldsymbol{\theta})$ at all points in Θ.

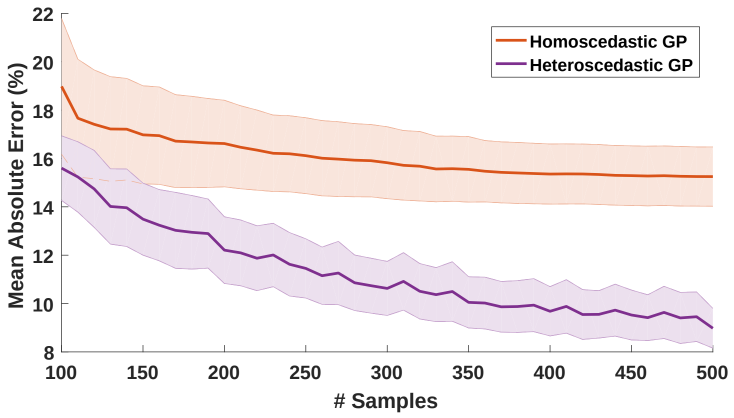


Figure 5-33: [Example 5.6.3] Comparison of mean absolute error (MAE) convergence for the PDF variance-based active sampling procedure using homoscedastic vs. heteroscedastic GP models. As the true likelihood model has heteroscedastic noise, the heteroscedastic GP should outperform the homoscedastic GP model, but the results indicate a significant reduction in MAE.

Second, the chapter introduced CDF variance in Section 5.2.2 as an efficient method for online quantification of prediction accuracy. Although the lack of an analytical solution for CDF variance prevents it from explicitly measuring the accuracy, the approximation is still extremely useful for identification of regions with low prediction confidence. Results in Section 5.6 repeatedly demonstrate this metric's ability to correctly identify regions of high prediction error without external validation methods. Lastly, the chapter developed new active sampling criteria for closed-loop verification of stochastic systems. The new CDF variance-based algorithms were able to consistently match or outperform the existing sampling strategies regardless of the underlying distribution of the robustness measurements.

Chapter 6

Extension: Stochastic Verification with Bernoulli Evaluations of Performance

Although Sections 5.4 and 5.5 discussed extensions to Algorithms 7 and 8 which enable them to address a wider class of problems, the Gaussian process-based verification approach in Chapter 5 does not address all types of stochastic verification problems. In particular, not all systems are able to provide real-valued measurements of trajectory robustness. This chapter presents the more general form of stochastic verification that encompasses a significantly wider class of possible problems. Rather than scalar robustness measurements, this approach relies upon Bernoulli distributions of binary evaluations of performance satisfaction to construct predictions models. As it relies upon binary evaluations for feedback on trajectory robustness, this work can be viewed as the stochastic equivalent of Chapter 3.

Despite the change to Bernoulli distributions, the chapter will follow roughly the same structure as Chapter 5. The first section will rewrite the problem from Section 5.1 in terms of Bernoulli distributions of binary measurements. The next section will introduce expectation propagation Gaussian process models as the new modeling and inference method for predictions of the probability of satisfaction at different operating conditions. Given this prediction model, Section 6.2.2 will rederive closed-loop statistical verification to match the new inference method. Although the implementation details have changed, Section 6.2.2 will also show that the underlying selection criteria is the same as in Section 5.3: reduce the variance of the cumulative distribution function. The last section will demonstrate the new verification procedures on two relevant systems.

6.1 Problem Description

Consider the same exact form of stochastic closed-loop system from (5.1). In these systems, the state dynamics are affected not by just deterministic parametric uncertainties $\boldsymbol{\theta}$, but also stochastic noise $\mathbf{w}(t)$. The stochastic noise may come from a variety of different sources, such as process and measurement noise, and is assumed to be the sole source of randomness in the closed-loop system. Regardless of the source, the end result of stochastic noise in the dynamics is the same: no two trajectories will be the same.

Unlike Chapter 5, this work reverts back to the original type of performance satisfaction measurements from Chapter 3.

Assumption 6.1. *A certification oracle or expert provides deterministic Boolean evaluations on whether a specific trajectory $\Phi(\mathbf{x}(t)|\mathbf{x}_0, \boldsymbol{\theta})$ satisfied the performance requirements. The Boolean evaluations are output as binary measurements where $y = +1$ corresponds to "satisfactory" performance, while $y = -1$ corresponds to "unsatisfactory" performance.*

The certification authority will return a single binary measurement for each trajectory. Although the system dynamics are stochastic, the binary measurements themselves are deterministic with respect to a given trajectory.

Remark 6.2. *As was mentioned in Chapter 5, the measurements are assumed to be deterministic with respect to the exact trajectory $\Phi(\mathbf{x}(t)|\mathbf{x}_0, \boldsymbol{\theta})$.*

Deterministic evaluations with respect to trajectory $\Phi(\mathbf{x}(t)|\mathbf{x}_0, \boldsymbol{\theta})$ means that, given the same exact sequence of states $\{\mathbf{x}(t=0), \dots, \mathbf{x}(t), \dots, \mathbf{x}(t=T_{final})\}$, the certification authority will always return the same measurement $y = \{+1, -1\}$. Stochasticity will cause any repeated simulations or experiments at the same $(\mathbf{x}_0, \boldsymbol{\theta})$ to produce completely new sequences of states, but the binary measurements themselves are not stochastic.

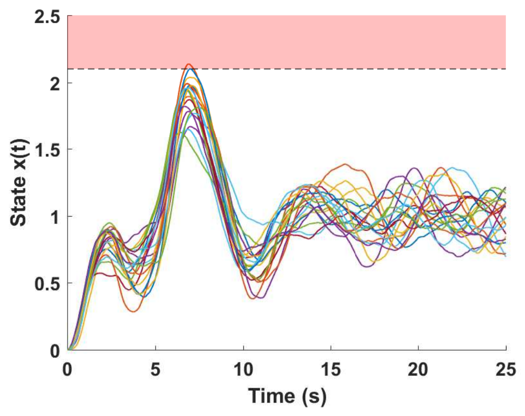
While binary evaluations $y = \{+1, -1\}$ are not as informative as non-binary measurements with $y \in \mathbb{R}$, both types of measurements still indicate the same fundamental result: whether the trajectory satisfied the performance requirement. This fact is an important observation as non-binary measurements $y \in \mathbb{R}$ are not always available, but binary evaluations are since the trajectory will either satisfy the requirement or not. In many problems, the oracle/expert is only able to provide crude Boolean evaluations of trajectory robustness rather than more informative STL robustness degrees or equivalents. For instance, if trajectories were rated by human experts, it would be difficult and time consuming to have the human experts provide non-discrete evaluations. In general, the most they would be expected to provide is a score from a small discrete set of options, such as integers between $1 - 10$. Additionally, the set of performance requirements themselves may not be in a form suitable for STL or similar metrics. For example, inverse reinforcement learning (IRL) [151–153] considers the problem of replicating the unknown, underlying reward function that motivates a (human) expert's actions. Verification problems with human certification experts/oracles can easily be viewed as an IRL problem because the logic used by the human expert is typically not known and hard to quantify. Therefore, it would extremely difficult to compute a function that replicates the latent, underyling decision making process, but rather straightforward to simply query the expert for binary evaluations and reproduce the shape. Essentially any verification problem will be able to output binary measurements for trajectory robustness.

Despite the difference in measurement types, the verification problem remains the same.

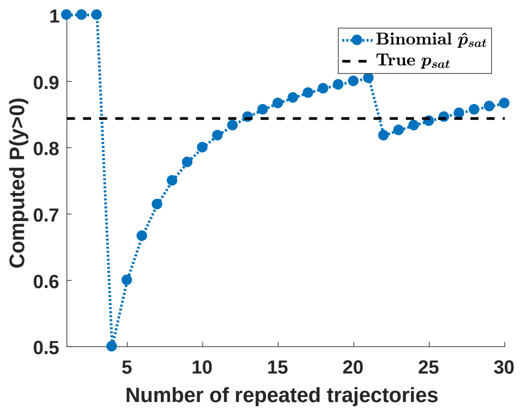
Definition 6.3. *There exists a satisfaction probability function $p_{sat}(\boldsymbol{\theta}) \in [0, 1]$ which defines the likelihood an arbitrary simulation or experimental test initialized at $\boldsymbol{\theta} \in \Theta$ will satisfy the performance requirement.*

The only difference between Definitions 5.4 and 6.3 is that $\mathbb{P}(y(\boldsymbol{\theta}) > 0) = \mathbb{P}(y(\boldsymbol{\theta}) = +1)$ for stochastic systems with binary measurements. In fact, the satisfaction probability function is actually the expected value of a Bernoulli distribution at parameter setting $\boldsymbol{\theta}$. Each trajectory initialized at $\boldsymbol{\theta}$ can be viewed as a Bernoulli trial with only two options - the corresponding binary measurement. Here, the two outcomes are $\{-1, +1\}$ rather than the customary $\{0, 1\}$ in Bernoulli trials. The objective of statistical verification is to compute an estimated satisfaction probability function $\widehat{p}_{sat}(\boldsymbol{\theta})$ to predict the likelihood of success at all operating conditions in Θ (the same objective as Prob. 5.1).

The primary challenge with binary measurements in stochastic systems is that the single binary measurement associated with each trajectory does not provide adequate information about the likelihood of satisfaction by itself. Previously in Chapter 5, one noisy measurement was suitable to estimate the entire distribution of $y(\boldsymbol{\theta})$ at that parameter vector. Now, multiple measurements will need to be taken at the same $\boldsymbol{\theta}$ location in order for anything to be inferred about the probability $\mathbb{P}(y(\boldsymbol{\theta}) = +1)$. These multiple measurements define a binomial distribution at $\boldsymbol{\theta}$, which is a finite sequence of independent Bernoulli trials that are drawn from the same distribution. This binomial distribution can be used to empirically estimate the underlying expected value $\widehat{p}_{sat}(\boldsymbol{\theta})$, but may require a substantial number of trajectories for $\widehat{p}_{sat}(\boldsymbol{\theta}) \longrightarrow p_{sat}(\boldsymbol{\theta})$. Figure 6-1 demonstrates this slow convergence on a repeat of the example from Figure 5-2. A number of simulations are performed at the same parameter setting $\boldsymbol{\theta}$, with the resulting trajectories shown in Figure 6-1(a). Given a finite number of trajectories and corresponding measurements, the empirical estimate $\widehat{p}_{sat}(\boldsymbol{\theta})$ in Figure 6-1(b) only begins to approach the true $p_{sat}(\boldsymbol{\theta}) = 0.844$ as the number of repetitions grows larger than 20. From the perspective of predicting $p_{sat}(\boldsymbol{\theta})$ at all $\boldsymbol{\theta} \in \Theta$, 20 samples is an intractably large number, as it would require 20 simulations or experiments at each $\boldsymbol{\theta} \in \Theta$. This issue will have to be addressed by the inference technique in order for statistical verification to be a viable method.



(a) 20 trajectories with the same initialization



(b) Binomial estimate for $\widehat{p}_{sat}(\boldsymbol{\theta})$

Figure 6-1: Binomial distributions for empirical computation of $\widehat{p}_{sat}(\boldsymbol{\theta})$. The left-hand plot displays 20 trajectories with the same initialization. The right-hand plot shows the change in the empirical average for $\widehat{p}_{sat}(\boldsymbol{\theta})$ as additional simulations are performed.

6.2 Probabilistic Classifiers for Stochastic Verification

The fact that each simulation or experiment can only be regarded as a single, uninformative Bernoulli trial adds further complication to the statistical prediction model. The model must still infer the change in $p_{sat}(\boldsymbol{\theta})$ across Θ, but a single measurement y can no longer define a distribution by itself as it did in Chapter 5. Instead, it is necessary to obtain multiple trajectories at each training location in order to compute an estimate for $p_{sat}(\boldsymbol{\theta})$ at each training location,

$$\widehat{p}_{sat}(\boldsymbol{\theta}) = \frac{1}{N_B}\sum_{i=1}^{N_B} y_i(\boldsymbol{\theta}) \qquad \forall \boldsymbol{\theta} \in \mathcal{D} \ . \tag{6.1}$$

This requirement for multiple measurements at each training location $\boldsymbol{\theta} \in \mathcal{D}$ introduces a second source of prediction error: the binomial confidence interval which places bounds on $|\widehat{p}_{sat}(\boldsymbol{\theta}) - p_{sat}(\boldsymbol{\theta})|$ according to $\widehat{p}_{sat}(\boldsymbol{\theta})$ and the number of samples at each training location N_B. As $N_B \longrightarrow \infty$, the confidence interval around the empirical estimate will shrink and therefore the impact of the value of N_B at each $\boldsymbol{\theta} \in \mathcal{D}$ should be directly incorporated into predictions.

Various inference techniques for modeling a spatially-varying Bernoulli satisfaction function already exist, but most use a variation of Gaussian processes [85, 154–156]. One approach of note is the beta-binomial Gaussian process model [154] that exploits the fact beta distributions are conjugate priors for binomial distributions to perform Bayesian inference on the binomial distribution at each training location. It augments the standard GP model with an additional covariance term for the binomial sample variance produced by beta-binomial inference. A similar and more common approach is expectation propagation Gaussian process (EP-GP) regression [85, 155, 156] for probabilistic classification with noisy binary outputs. Rather than operate on the binary measurements directly, EP-GPs transform the outputs from the current problem's $[0, 1]$ domain to a GP's $(-\infty, +\infty)$ domain with a nonlinear transformation. Ultimately, both approaches are very similar as they both predict $p_{sat}(\boldsymbol{\theta})$, but this chapter utilizes the EP-GP technique. The same underlying concepts can be readily applied to beta-binomial GP models with only slight modifications.

6.2.1 Expectation Propagation Gaussian Process Models

The main difference between EP-GP models and the standard GPs presented in Chapter 5 is that the new EP-GP regression models estimate $p_{sat}(\boldsymbol{\theta})$ directly rather than through the intermediary PDF for $\bar{y}(\boldsymbol{\theta})$. While operations on $p_{sat}(\boldsymbol{\theta})$ instead of intermediary $\bar{y}(\boldsymbol{\theta})$ seem more straightforward, Bernoulli-distributed trajectory measurements complicate the construction of the Gaussian process prediction model. For one, the GP output for $\widehat{p}_{sat}(\boldsymbol{\theta})$ falls within the range $[0, 1]$, but Gaussian processes traditionally operate in the $(-\infty, +\infty)$ domain. Likewise, the binary trajectory robustness observations $(y(\boldsymbol{\theta}) = \{-1, 1\})$ are no longer Gaussian distributed and therefore require a new likelihood model.

Both of these complications are addressed by the inverse probit transformation. The inverse probit transformation [85, 155] maps a real-valued latent function $h(\boldsymbol{\theta}) \in$

$\mathbb{R}$ to $p_{sat}(\boldsymbol{\theta})$ using the cumulative distribution of a standard Gaussian,

$$p_{sat}(\boldsymbol{\theta}) = \int_{+\infty}^{h(\boldsymbol{\theta})} \mathcal{N}(0,1) = \frac{1}{2} + \frac{1}{2}\,\mathrm{erf}\left(\frac{h(\boldsymbol{\theta})}{\sqrt{2}}\right), \tag{6.2}$$

or conversely

$$h(\boldsymbol{\theta}) = \sqrt{2}\,\mathrm{erf}^{-1}\Big(2\Big(p_{sat}(\boldsymbol{\theta}) - \frac{1}{2}\Big)\Big). \tag{6.3}$$

To highlight the importance of the latent function during the computation of $p_{sat}(\boldsymbol{\theta})$, the inverse probit transformation will be written as $p_{sat}(\boldsymbol{\theta}) = \Psi(h(\boldsymbol{\theta}))$. Using the probit model, the likelihood for a stochastic binary measurement becomes

$$\begin{aligned}\mathbb{P}\Big(y(\boldsymbol{\theta}) = 1|h(\boldsymbol{\theta})\Big) &= \Psi(h(\boldsymbol{\theta}))^{0.5y(\boldsymbol{\theta})+0.5} + \big(1 - \Psi(h(\boldsymbol{\theta}))\big)^{0.5y(\boldsymbol{\theta})-0.5} \\ &= \Psi\Big(h(\boldsymbol{\theta}) \cdot y(\boldsymbol{\theta})\Big)\end{aligned} \tag{6.4}$$

since $\Psi(h(\boldsymbol{\theta}))$ is symmetric. Additionally, as each measurement $y(\boldsymbol{\theta})$ is an independent Bernoulli trial, the joint likelihood conveniently factorizes to

$$\mathbb{P}(\mathbf{y}|\mathbf{h}) = \prod_{j=1}^{N_B}\prod_{i=1}^{N} \mathbb{P}(y_{i,j}|h(\boldsymbol{\theta}_i)), \tag{6.5}$$

assuming N training locations and N_B multiple measurements at each training location. As before, the posterior distribution for the latent function $\mathbf{h}$ can be computed by Bayes' rule,

$$\mathbb{P}(\mathbf{h}|\mathcal{L}, \psi) \propto \mathbb{P}(\mathbf{y}|\mathbf{h})\ \mathbb{P}(\mathbf{h}|\mathcal{D}, \psi), \tag{6.6}$$

where $\mathbb{P}(\mathbf{h}|\mathcal{D}, \psi)$ is the prior distribution. Despite the change in the likelihood model, the prior distribution is still a multivariate Gaussian $\mathcal{N}(\mathbf{h}|\mathbf{m}, \mathbf{K})$ with kernel function κ, kernel hyperparameters ψ, and mean $\mathbf{m}$. This chapter uses the squared exponential kernel. In the absence of any prior knowledge about $p_{sat}(\boldsymbol{\theta})$, the prior mean $\mathbf{m}$ is set to $\mathbb{E}[h(\boldsymbol{\theta})] = 0$, which corresponds to $p_{sat}(\boldsymbol{\theta}) = 0.5$.

Due to the probit likelihood function, the posterior in (6.6) cannot be computed analytically. Instead, the EP-GP model uses expectation propagation [85, 86] to approximate the likelihood with a local likelihood approximation in the form of an unnormalized Gaussian,

$$\mathbb{P}(y(\boldsymbol{\theta}_i) = 1 | h(\boldsymbol{\theta}_i)) \approx \widetilde{Z}_i \mathcal{N}(h(\boldsymbol{\theta}_i) | \widetilde{\mu}_i, \widetilde{\sigma}_i^2), \tag{6.7}$$

where $\widetilde{Z}_i, \widetilde{\mu}_i, \widetilde{\sigma}_i$ are the site parameters. The EP-GP training process is quite involved, but sequentially adjusts the site parameters to approximate the likelihood model best supported by the observed training dataset. The training process will repeatedly update these local approximations until the solution converges or it has run out of allowable iterations. For a significantly more detailed description of the EP-GP training process, see Chapter 3 in Rasmussen and Williams' book [85].

The output of the training process is a Gaussian distribution $\mathcal{N}\big(h(\boldsymbol{\theta}_*) | \mu_h(\boldsymbol{\theta}_*), \Sigma_h(\boldsymbol{\theta}_*)\big)$ that provides an approximate posterior predictive distribution for the latent function $h(\boldsymbol{\theta}_*)$ at arbitrary parameter vector $\boldsymbol{\theta}_*$. This predictive distribution for $h(\boldsymbol{\theta}_*)$ defines the expected probability of satisfaction,

$$\begin{aligned} \widehat{p}_{sat}(\boldsymbol{\theta}_*) &= \int \Psi\big(h(\boldsymbol{\theta}_*)\big) \mathcal{N}\big(h(\boldsymbol{\theta}_*) | \mu_h(\boldsymbol{\theta}_*), \Sigma_h(\boldsymbol{\theta}_*)\big) d(h(\boldsymbol{\theta}_*)) \\ &= \frac{1}{2} + \frac{1}{2} \mathrm{erf}\left(\frac{\mu_h(\boldsymbol{\theta}_*)}{\sqrt{1 + \Sigma_h(\boldsymbol{\theta}_*)}} \right). \end{aligned} \tag{6.8}$$

Note that this final approximation is very similar to the Gaussian CDF from (5.14). The variance of the predictions for $p_{sat}(\boldsymbol{\theta})$, labeled $V(\boldsymbol{\theta}|\mathcal{L}, \psi)$ like it was in Chapter 5, can be determined two different ways. First, it is possible to compute the variance using the predictive distribution for $h(\boldsymbol{\theta}_*)$ and the inverse probit function. The resulting approximate variance resembles (5.18) with $\epsilon_y = 1$ since the inverse probit function uses the standard Gaussian PDF. The second approach is to compute the variance of the predictions as the variance of the Bernoulli distribution defined by $\widehat{p}_{sat}(\boldsymbol{\theta})$. Both approaches are explored in the upcoming closed-loop verification procedures and they demonstrate very similar results.

Ultimately, the EP-GP model enables the verification framework to translate stochastic binary data into predicted probability of requirement satisfaction at all potential parametric uncertainties. The main downside of EP-GPs and all stochastic Bernoulli approaches is the additional computational cost associated with the more complicated training process. The new site parameters add more variables that must be optimized during the hyperparameter optimization process, which takes additional computations and lengthens the training process in comparison to Chapter 5. Likewise, multiple measurements are typically required at each training location and therefore the training sets $\mathcal{L}$ will have more datapoints, requiring more computational resources and time to first perform the simulations or experiments and then compute the predictions with the larger $\mathcal{L}$.

6.2.2 Closed-Loop Statistical Verification

Despite the change in measurements and prediction models, closed-loop statistical verification with Bernoulli trials uses the same sampling procedures from Chapter 5 with only minor modifications. Algorithms 9-11 detail these new techniques. The biggest differences between Algorithms 9-11 and the algorithms in Chapter 5 are multiple simulations/experiments performed at each training location and a slight change to the selection criteria.

Rather than select training locations based upon posterior CDF variance reduction as in Section 5.3, these new variants will only select training locations with the highest current CDF variance. There is no analytical method for computing posterior CDF variance at a particular training location like there was for Gaussian-distributed measurements. However, the analysis in Section 5.3 did identify the posterior reduction in CDF variance was generally highest at $\boldsymbol{\theta}$ locations with large CDF variance. Thus, the three active sampling algorithms for Bernoulli distributions examine current CDF variance in place of the unknown posterior CDF variance reduction.

The use of multiple measurements at each training location adds an additional for-loop. For example, in Algorithm 9 after the best training location $\overline{\boldsymbol{\theta}}$ has been selected, the algorithm performs N_B simulations or experiments at $\overline{\boldsymbol{\theta}}$. All N_B of the corresponding measurements are then added to $\mathbf{y}$ instead of a single scalar value. Algorithms 10 and 11 have a similar modification so that ultimately N_B trajectories are obtained for each of the M training locations in the batch $\mathcal{S}$. Because the CDF

Algorithm 9 Sequential closed-loop stochastic verification framework

```
1: Input: initial training set $\mathcal{L} = \{\mathcal{D}, \mathbf{y}\}$, available sample locations $\mathcal{U}$, # of iterations $T$, # of simulations at each location $N_B$
2: Initialize: train EP-GP prediction model
3: for $i = 1, 2, \ldots, T$ do
4:     Select $\overline{\theta} = \underset{\boldsymbol{\theta} \in \mathcal{U}}{\text{argmax}}\ V(\boldsymbol{\theta}|\mathcal{L}, \psi)$
5:     for $j = 1, 2, \ldots, N_B$ do
6:         Perform test at $\overline{\boldsymbol{\theta}}$, obtain measurements $y(\overline{\boldsymbol{\theta}})$
7:         Add $\{\overline{\boldsymbol{\theta}}, y(\overline{\boldsymbol{\theta}})\}$ to training set $\mathcal{L}$
8:     end for
9:     Retrain EP-GP model with updated $\mathcal{L}$
10: end for
11: Return: expected value of the satisfaction probability function $\widehat{p}_{sat}(\boldsymbol{\theta})$ and corresponding variance $V\big(\boldsymbol{\theta}|\mathcal{L}, \psi\big)$
```

variance will be more sensitive in regions where $\widehat{p}_{sat}(\boldsymbol{\theta}) \approx 0.5$, it may be advantageous to obtain more trajectory data at parameter settings with $\widehat{p}_{sat}(\boldsymbol{\theta}) \approx 0.5$ than at points where $\widehat{p}_{sat}(\boldsymbol{\theta})$ is close to 0 or 1. Therefore, the algorithms do not remove sampled locations from the list of available training locations, meaning $\mathcal{U} = \Theta_d$. This allows the algorithm to ultimately select more than N_B measurements at a particular parameter setting if the selection criterion expects these additional measurements will improve the predictions.

6.3 Simulation Results

In order to highlight the benefits and limitations of closed-loop statistical verification with Bernoulli measurements, Algorithms 10 and 11 are demonstrated on two stochastic versions of systems from earlier chapters. The first example examines the same problem from Section 5.6.1 in the last chapter, while the second example considers a stochastic version of the Van der Pol oscillator from Section 3.5.1. This last example will also identify cases where the new CDF variance-based active sampling algorithms will not demonstrate significant improvements over other sampling strategies.

Algorithm 10 Batch closed-loop stochastic verification framework using importance-weighted random sampling

```
1:  Input: initial training set $\mathcal{L} = \{\mathcal{D}, \mathbf{y}\}$, available sample locations $\mathcal{U}$, # of iterations $T$, batch size $M$, # of simulations at each location $N_B$
2:  Initialize: train EP-GP prediction model
3:  for $i = 1, 2, \ldots, T$ do
4:      Initialize: $\mathcal{S} = \emptyset$
5:      Transform $V(\boldsymbol{\theta}|\mathcal{L}, \psi)$ into probability distribution $P_V(\boldsymbol{\theta})$
6:      Generate $M$ random samples from $P_V(\boldsymbol{\theta})$, add to $\mathcal{S}$
7:      for $j = 1, 2, \ldots, N_B$ do
8:          Perform tests $\forall \boldsymbol{\theta} \in \mathcal{S}$, obtain measurements $\mathbf{y}_{\mathcal{S}}$
9:          Add $\{\mathcal{S}, \mathbf{y}_{\mathcal{S}}\}$ to training set $\mathcal{L}$
10:     end for
11:     Retrain EP-GP model with updated $\mathcal{L}$
12: end for
13: Return: expected value of the satisfaction probability function $\widehat{p}_{sat}(\boldsymbol{\theta})$ and corresponding variance $V\big(\boldsymbol{\theta}|\mathcal{L}, \psi\big)$
```

Algorithm 11 Batch closed-loop stochastic verification framework using determinantal point processes

```
1:  Input: initial training set $\mathcal{L} = \{\mathcal{D}, \mathbf{y}\}$, available sample locations $\mathcal{U}$, # of iterations $T$, batch size $M$, # of simulations at each location $N_B$
2:  Initialize: train EP-GP prediction model
3:  for $i = 1, 2, \ldots, T$ do
4:      Initialize: $\mathcal{S} = \emptyset$
5:      Transform $V(\boldsymbol{\theta}|\mathcal{L}, \psi)$ into probability distribution $P_V(\boldsymbol{\theta})$
6:      Generate $M_T$ random samples from $P_V(\boldsymbol{\theta})$, construct k-DPP
7:      Generate $M$ random samples from k-DPP, add to $\mathcal{S}$
8:      for $j = 1, 2, \ldots, N_B$ do
9:          Perform tests $\forall \boldsymbol{\theta} \in \mathcal{S}$, obtain measurements $\mathbf{y}_{\mathcal{S}}$
10:         Add $\{\mathcal{S}, \mathbf{y}_{\mathcal{S}}\}$ to training set $\mathcal{L}$
11:     end for
12:     Retrain EP-GP model with updated $\mathcal{L}$
13: end for
14: Return: expected value of the satisfaction probability function $\widehat{p}_{sat}(\boldsymbol{\theta})$ and corresponding variance $V\big(\boldsymbol{\theta}|\mathcal{L}, \psi\big)$
```

6.3.1 Concurrent Learning Model Reference Adaptive Controller

This example considers the same stochastic CL-MRAC system from Section 5.6.1. The uncertain open-loop system is subject to two parametric uncertainties $\boldsymbol{\theta} = [\theta_1, \theta_2]^T$ and also corrupted by zero-mean Gaussian process noise $\mathbf{w}(t) \sim \mathcal{N}(\mathbf{0}, \Sigma_w)$ with diagonal covariance matrix $\Sigma_w = \text{diag}([5, 5])$. The adaptive control system estimates the two uncertain parameters and attempts to track the desired reference trajectory $\mathbf{x}_m(t)$. The performance requirement states the state variable $x_1(t)$ must remain within 1 unit of the reference state $x_{m_1}(t)$ over the entire 40 second trajectory for the trajectory's performance to be considered "satisfactory", i.e. $y = +1$. As before, the set of all possible parameter settings (Θ) was approximated with a finite sampling grid Θ_d of 40,401 possible parameter vectors spanning $|\theta_1| \leq 10$ and $|\theta_2| \leq 10$.

The true probability of requirement satisfaction $p_{sat}(\boldsymbol{\theta})$ is directly taken from the same dataset used in Section 5.6.1. In order to determine $p_{sat}(\boldsymbol{\theta})$, 50 simulations were performed at each of the possible sampling locations in Θ_d and the cumulative distribution produced $p_{sat}(\boldsymbol{\theta})$. The true probability of satisfaction is displayed in Figure 6-2, which is the same exact shape shown earlier in Figure 5-9. Unlike that previous example, the satisfaction of the requirements is not measured through real-valued scalar measurements, but rather through binary evaluations $y(\boldsymbol{\theta}) = \{-1, 1\}$.

This example compares both Algorithms 10 and 11 against one another as well as a random sampling procedure. Because of the lack of non-binary measurements in these problems, there are no existing PDF mean- or variance-based active sampling procedures to compare Algorithms 10 and 11 against. Instead, an active sampling algorithm that selects $\boldsymbol{\theta}$ vectors with $\widehat{p}_{sat}(\boldsymbol{\theta})$ predictions closest to $\widehat{p}_{sat}(\boldsymbol{\theta}) = 0.5$ was developed as a rough equivalent to the PDF mean-based procedure from the preceding chapter. In this example, all of the sampling procedures start with 30 randomly-selected initial training locations with 5 simulations performed at each of these locations for a total of 150 binary evaluations in $\mathcal{L}$. Figure 6-3(a) displays one

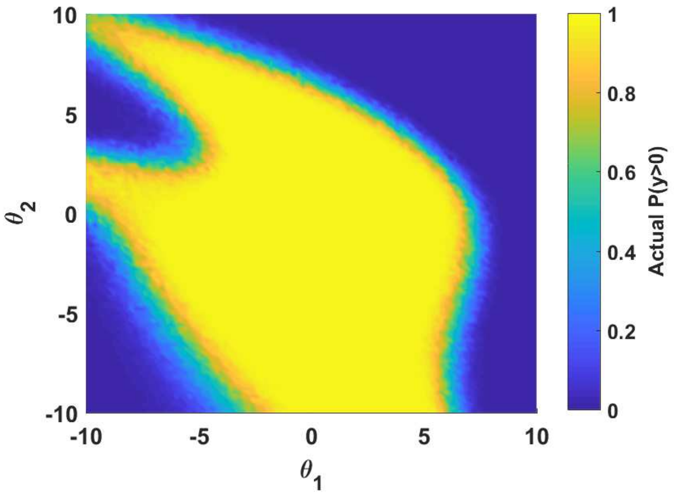
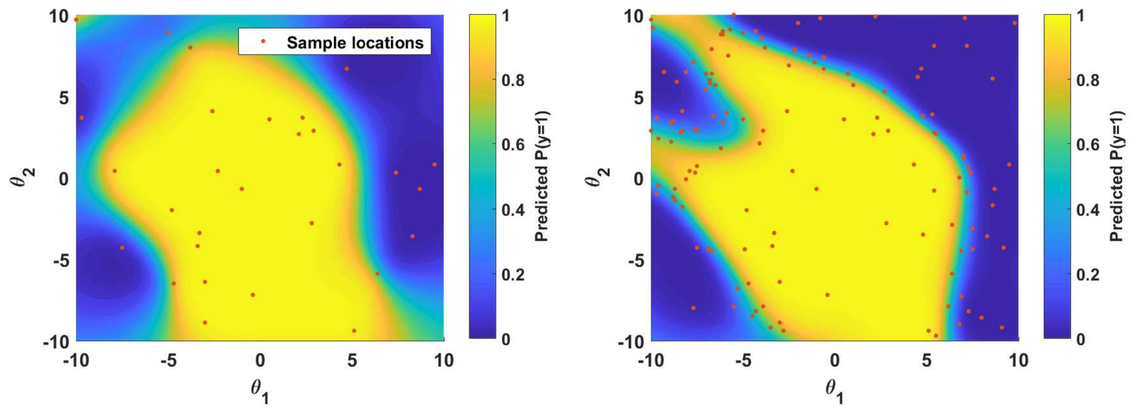


Figure 6-2: [Example 6.3.1] True probability of satisfaction $p_{sat}(\boldsymbol{\theta})$ for the stochastic CL-MRAC system. Note, this is the same $p_{sat}(\boldsymbol{\theta})$ from Figure 5-9.



(a) At the initial training step

(b) After 20 iterations of Algorithm 11

Figure 6-3: [Example 6.3.1] Predicted satisfaction probability function at the initial training step and after 20 iterations of Algorithm 11.

of these initial training datasets and the resulting $\widehat{p}_{sat}(\boldsymbol{\theta})$. In comparison to the true shape in Figure 6-2, these predictions fail to adequately replicate the true probability function $p_{sat}(\boldsymbol{\theta})$.

The four sampling approaches operate in batches of $M = 5$ training locations and $N_B = 5$ simulations at each of those training locations until a limit of $T = 20$ iterations has been reached. Figure 6-3(b) illustrates the final predictions after 20 iterations of Algorithm 11 have been performed. The resulting predictions are sig-

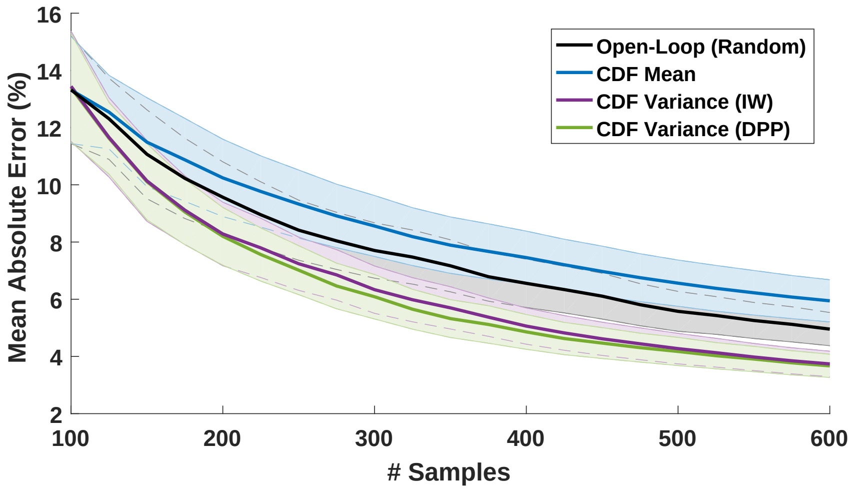
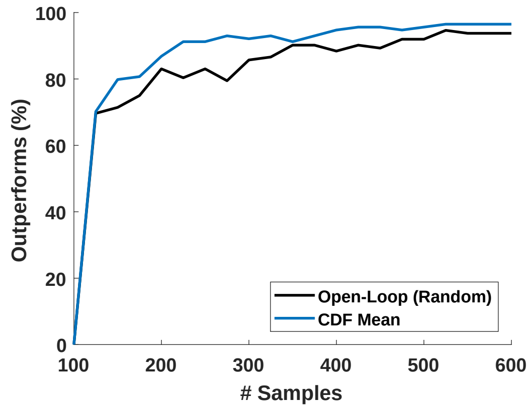
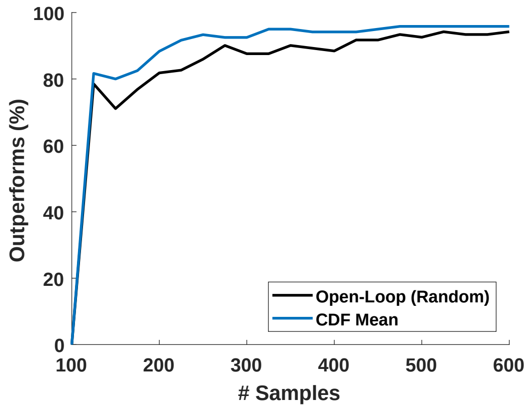


Figure 6-4: [Example 6.3.1] Comparison of mean absolute error (MAE) convergence for the different sampling strategies. The results compare both Algorithms 10 (IW) and 11 (DPP) against the CDF mean-focused and open-loop approaches.

nificantly more accurate estimates of $p_{sat}(\boldsymbol{\theta})$ from Figure 6-2 than the initial step in Figure 6-3(a). Indeed, all four sampling approaches are able to noticeably reduce the mean absolute error (MAE), as seen in Figure 6-4. This figure compares the distribution of mean absolute errors over 150 randomly-initialized test cases. Algorithms 10 and 11 demonstrate extremely similar MAE convergence, with DPP-based Algorithm 11 slightly outperforming the importance-weighted sampling approach. The improvement in MAE convergence over the competing sampling approaches is shown in Figure 6-5. Both of the new active sampling algorithms based upon CDF variance either outperform or match the MAE convergence of the open-loop random sampling procedure and the active sampling procedure that uses the CDF mean-focused selection criterion. As was seen in Figure 6-4, the DPP-based algorithm is slightly better than Algorithm 10. All of these results suggest the proposed active sampling algorithms based upon CDF variance will select informative $\boldsymbol{\theta} \in \Theta_d$ to improve the predictions.



(a) Algorithm 10: CDF variance (IW)



(b) Algorithm 11: CDF variance (DPP)

Figure 6-5: [Example 6.3.1] Ratio of runs where Algorithms 10 and 11 directly outperform or match the MAE levels of the other sampling approaches.

6.3.2 Stochastic Van der Pol Oscillator

The second example is a stochastic version of the unstable Van der Pol oscillator from Section 3.5.1. However, the Van der Pol oscillator is not naturally a stochastic system, so stochasticity was artificially added to the problem. The dynamics (3.19) are kept deterministic, but the initial conditions $\mathbf{x}(0) = [x_1(0), x_2(0)]^T$ are corrupted by stochastic noise. As was the case in Section 3.5.1, the initial conditions are also functions of parametric uncertainties $\boldsymbol{\theta}$. In short, the initial conditions $\mathbf{x}(0)$ are given by

$$\mathbf{x}(0) = \mathbf{x}_0 + \boldsymbol{\theta} + \mathbf{w} \tag{6.9}$$

where $\mathbf{x}_0 = [0, 0]^T$, $\boldsymbol{\theta} = [\theta_1, \theta_2]^T$, and $\mathbf{w} = \text{diag}([0.01, 0.01])$. These initial conditions ultimately determine the stability of the nonlinear system.

Due to the stochasticity, a particular $\boldsymbol{\theta}$ vector will no longer deterministically decide whether the system is stable or not, as was seen in Figure 3-1. Instead, there will be a probability that the system's subsequent trajectory is stable. The underlying probability of satisfaction function $p_{sat}(\boldsymbol{\theta})$ is shown in Figure 6-6. In comparison to Figure 3-1(a), the probability of satisfaction is no longer just $\mathbb{P}(y(\boldsymbol{\theta}) = 1) \in \{0, 1\}$, but covers the whole range of probabilities between 0 and 1. Despite the change to $p_{sat}(\boldsymbol{\theta}) \in [0, 1]$, it is still possible to see the rough outline of the deterministic

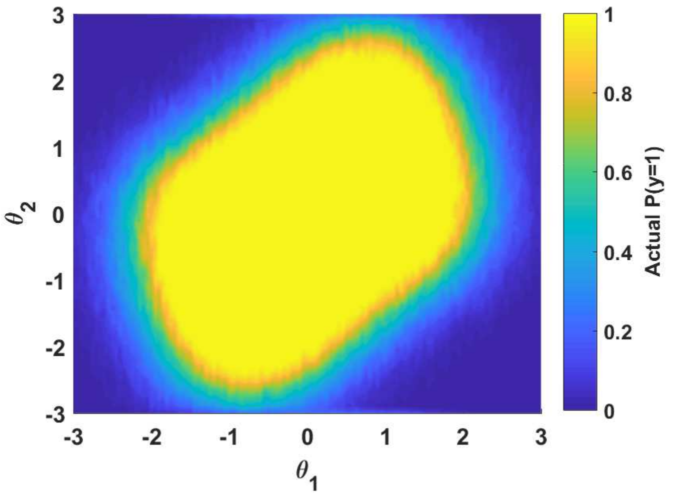
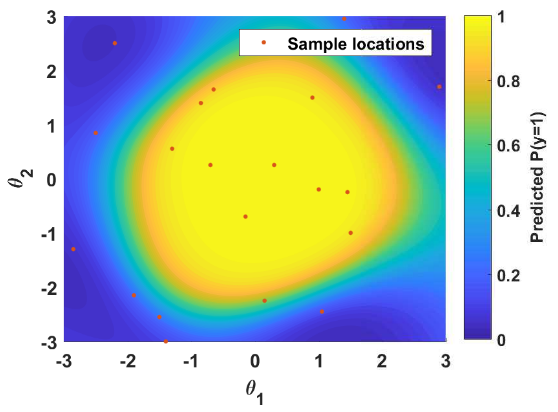


Figure 6-6: [Example 6.3.2] True probability of satisfaction $p_{sat}(\boldsymbol{\theta})$ for the stochastic Van der Pol oscillator.

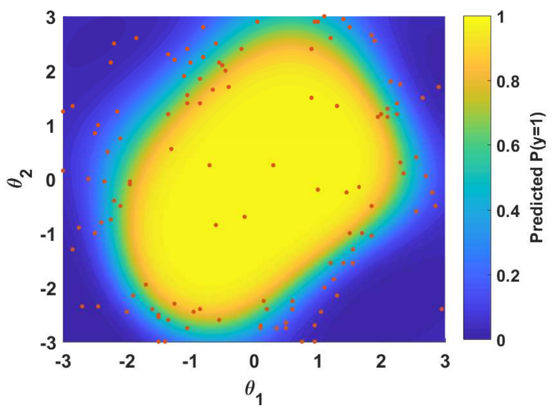
region-of-attraction from Figure 3-1(a).

The comparison of the active sampling procedures is similar to the preceding CL-MRAC example. In this example, the four sampling strategies all start from an initial training dataset of 20 randomly-selected training locations with 5 simulations at each location for a total of 100 initial training datapoints. These training points are taken from grid Θ_d of 14,461 possible sample locations covering $\theta_1 : [-3, 3]$ and $\theta_2 : [-3, 3]$. An initial training dataset and the resulting prediction model for $\widehat{p}_{sat}(\boldsymbol{\theta})$ is shown in Figure 6-7(a). The initial predictions do not adequately estimate the true $p_{sat}(\boldsymbol{\theta})$ and more samples are needed. The four sampling strategies operate in batches of $M = 5$ training locations with $N_B = 5$ simulations at each of those locations. The procedures will run until a limit of $T = 20$ iterations has been reached. The same example from Figure 6-7(a) after the completion of the 20 iterations of Algorithm 11 are shown in Figure 6-7(b). While it does not perfectly replicate the true $p_{sat}(\boldsymbol{\theta})$, the model's output for $\widehat{p}_{sat}(\boldsymbol{\theta})$ is a closer approximation of $p_{sat}(\boldsymbol{\theta})$.

Figures 6-8 and 6-9 compare the prediction performance of the various sampling approaches over 150 randomly-initialized test runs. Just as in Section 6.3.1, Algorithms 10 and 11 demonstrate lower average MAE than the other sampling approaches. When the results for the CDF mean-based and open-loop, random sam-



(a) At the initial training step



(b) After 20 iterations of Algorithm 11

Figure 6-7: [Example 6.3.2] Predicted satisfaction probability function at the initial training step and after 20 iterations of Algorithm 11.

pling procedures are directly compared against the results from Algorithms 10 and 11 in each of the 150 test runs, the CDF variance-based approaches outperform or match the mean absolute error levels of the other strategies in nearly 100% of the test run. This favorable MAE performance indicates Algorithms 10 and 11 are the best active sampling strategies to employ in order to minimize prediction error for this particular example.

While the new CDF variance-based sampling algorithms clearly outperformed the other strategies in the VDP example with stochastic noise set to $\mathbf{w} = \text{diag}\left([0.01, 0.01]\right)$, their improvement over the competing strategies will diminish as the noise increases. When the noise is increased to $\mathbf{w} = \mathbf{I}$, the true probability of satisfaction $p_{sat}(\boldsymbol{\theta})$ has a much lower gradient. This new shape is shown in Figure 6-10. Due to the more gradual slope, a significantly higher ratio of the datapoints fall between $p_{sat}(\boldsymbol{\theta}) = 0$ and 1 and a larger portion of those $\boldsymbol{\theta}$ locations have $p_{sat}(\boldsymbol{\theta}) \approx 0.5$. In fact, 63% of Θ_d has a probability of satisfaction between 0.2 and 0.8 and 77% is between 0.1 and 0.9. For the original noise setting with $\mathbf{w} = \text{diag}\left([0.01, 0.01]\right)$, these percentages were only 25% and 36%. Since the CDF variance is generally much higher at points with a probability of satisfaction near 0.5, all those points will have a similar level of CDF variance and the probability $\mathbb{P}_V(\boldsymbol{\theta})$ used by both Algorithms 10 and 11 will be closer to the uniform distribution used by the open-loop, random sampling approach. Like-

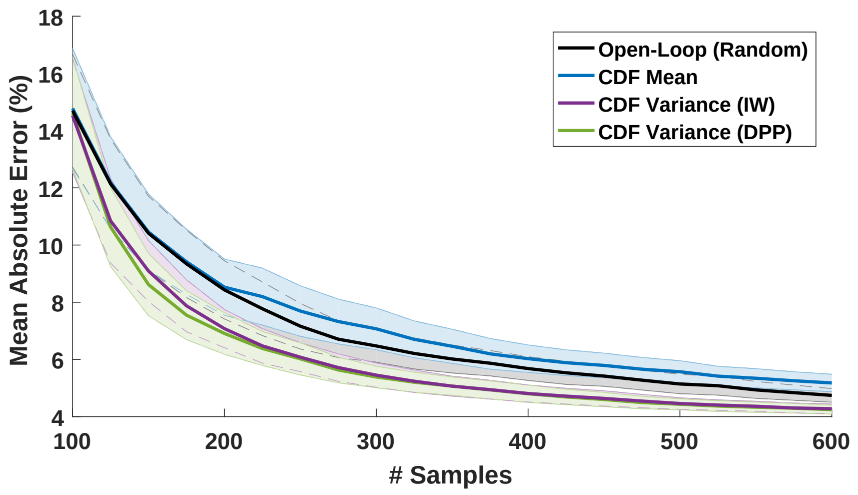
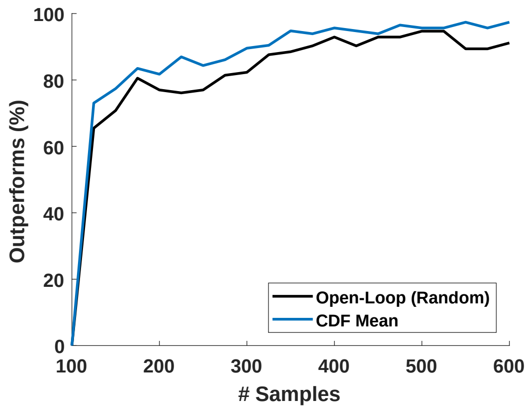
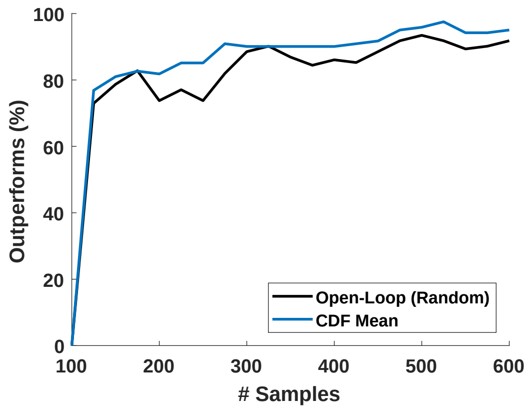


Figure 6-8: [Example 6.3.2] Comparison of mean absolute error (MAE) convergence for the different sampling strategies. The results compare both Algorithms 10 (IW) and 11 (DPP) against the CDF mean-focused and open-loop approaches.



(a) Algorithm 10: CDF variance (IW)



(b) Algorithm 11: CDF variance (DPP)

Figure 6-9: [Example 6.3.2] Ratio of runs where Algorithms 10 and 11 directly outperform or match the MAE level of the other sampling approaches.

wise, since a larger majority of the points in Θ_d have a probability of satisfaction that is close to $p_{sat}(\boldsymbol{\theta}) = 0.5$, the CDF mean-based sampling criterion that selects points with $\widehat{p}_{sat}(\boldsymbol{\theta})$ close to 0.5 will more evenly distribute training locations than it had before with the higher magnitude gradient in Figure 6-6. Therefore, the different sample selection criteria weight the different sampling locations similarly and will none will have a clear advantage over the others. This exact result is seen in Figure 6-11 where all four of the sampling strategies have roughly the same MAE convergence. For

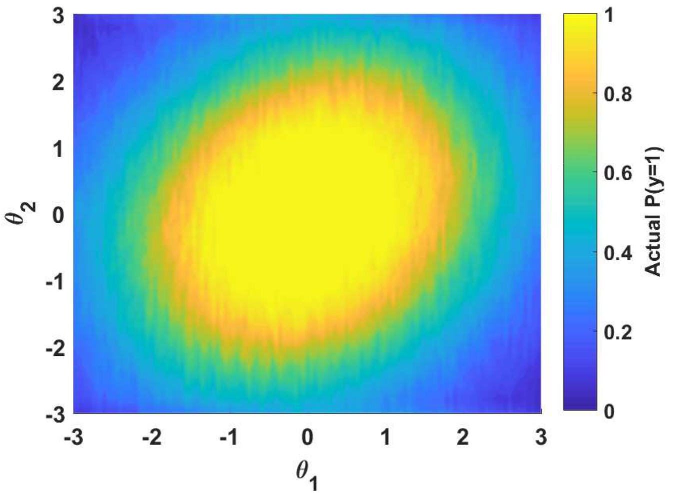


Figure 6-10: [Example 6.3.2] True probability of satisfaction $p_{sat}(\boldsymbol{\theta})$ for the high variance case.

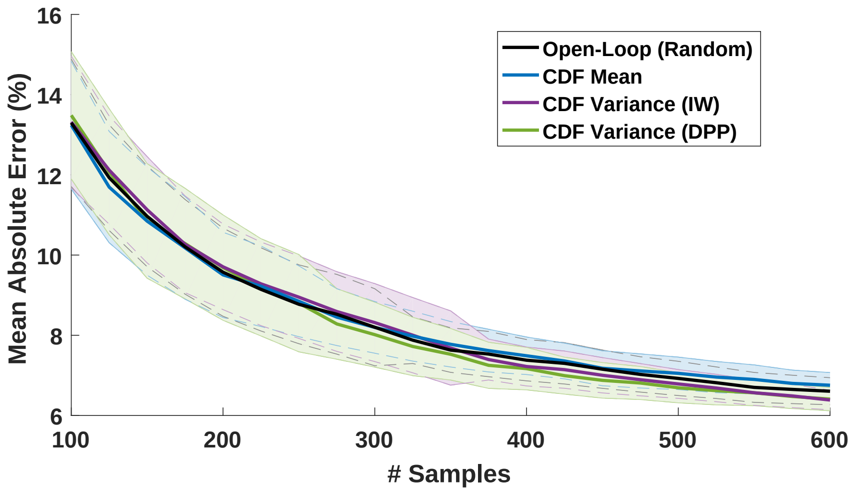


Figure 6-11: [Example 6.3.2] Comparison of mean absolute error (MAE) convergence for the different sampling strategies for the high variance case. The results compare both Algorithms 10 (IW) and 11 (DPP) against the CDF mean-focused and open-loop approaches.

comparison, this same trend was observed in Section 5.6 as the stochastic noise was increased in the CL-MRAC and autopilot examples. These results don't necessarily identify a problem with the CDF variance selection criteria since the MAE performance is the same as the other procedures, but definitely highlight the limitations of Algorithms 10 and 11 when applied to systems with large stochastic noise.

6.4 Summary

This chapter redeveloped the stochastic statistical verification frameworks from Chapter 5 for problems that are only able to produce binary measurements of performance requirement satisfaction. These binary measurements are Bernoulli trials drawn from the underlying probability function $p_{sat}(\boldsymbol{\theta})$ that describes the probability of requirement satisfaction at every possible parameter setting. Section 6.2 introduced expectation propagation Gaussian processes as the modeling and inference technique for predicting $p_{sat}(\boldsymbol{\theta})$ and Section 6.2.2 detailed the necessary changes to the closed-loop statistical verification methods to accept binary measurements. The two examples in Section 6.3 demonstrated the effectiveness of the modified statistical verification frameworks and highlighted their strengths and limits.

In comparison to Chapter 5, this Chapter makes no restrictive assumptions about the type or distribution of measurements. Due to the binary nature of verification, every stochastic verification problem can be rederived as a problem with Bernoulli distributions of binary measurements. As a result, every example from Chapter 5 can actually be written in terms of Bernoulli distributions. While this fact serves as bridge between the two chapters and explains why they share the same concepts and general approaches, it does not mean every stochastic verification problem should be converted into this chapter's more general representation. The increased complexity of EP-GP models increases the computational cost of training in comparison to the the standard Gaussian prediction models in Chapter 5. More importantly, these procedures also generally require more simulations or experimental runs in order to converge to a similar level of prediction errors. Therefore, the work presented in this chapter is an alternative approach to Chapter 5 that can handle a wider class of problems, but which also sacrifices efficiency and prediction accuracy for this wider applicability.

Chapter 7

Multi-Stage Verification and Experimental Testing

This chapter addresses implementation challenges that arise during multi-stage verification and develops new procedures specifically tailored to real-world, experiment-based scenarios with stringent safety considerations. As discussed in Section 1.2.1 and displayed in Figure 1-2(b), verification sometimes spans multiple stages of increasing complexity and fidelity. For instance in aircraft control system design, the closed-loop performance of the aircraft would first be verified in lower-fidelity simulation models to weed out wildly undesirable designs, then verified with more realistic simulations models, before the final design is flight tested on a flying prototype. In order to speed up the process and improve the efficiency, information from earlier stages should be passed to later stages in a principled manner. Additionally, a final stage with actual hardware testing typically imposes a new set of constraints on the verification procedure. Particularly when examining safety requirements, the closed-loop system's failure to satisfy a requirement might translate into partial or complete loss of the prototype. As opposed to simulation-based verification where successes and failures share the same computational cost, the cost of experimental failures greatly outweighs the cost of satisfactory trajectories, so the verification procedure should avoid failures. The following chapter both addresses forward transfer of information between verification stages as well as introduces new procedures to minimize the risk of experimental failures during data-driven verification.

The chapter is organized around four sections. First, Section 7.1 presents a simple, but theoretically-justified, approach to transfer information from earlier verification stages to later stages. This approach inputs the predictions from earlier stages as nonzero-mean priors in order to incorporate their effect without assuming they are completely accurate in the later stages. Section 7.2 introduces the problem of failure constraints in experiment-based statistical verification, where the greater cost of trajectories that fail to satisfy safety requirements motivates the need to avoid parametric uncertainties associated with a high probability of failure. The failure-adverse closed-loop statistical verification procedures in Section 7.3 are specifically developed to minimize the number of failures during the verification process and are demonstrated in Section 7.4.

7.1 Forward Transfer in Multi-Stage Verification

One of the main challenges in multi-stage verification is the incorporation of predictions from preceding verification stages into the training process of the prediction model in later stages. This challenge evolves out of a number of different considerations. First, earlier verification stages typically use lower-fidelity models with simplified dynamics while the later stages rely upon more realistic models with higher-order dynamics and higher-fidelity effects. At the extreme, real-world experiments are treated as the highest-fidelity "simulator" with unmodeled dynamics. However slight the differences between lower- and higher-fidelity models, they may be enough to result in drastically different predictions, which makes it inadvisable to blindly trust the accuracy of predictions from earlier verification stages in later stages.

Similarly, the various stages could address different verification problems where the objectives do not completely overlap. For instance, the closed-loop system may be verified on a deterministic simulation model before it is verified on a stochastic simulation model. However, the binary classification objective of $\Theta_{sat}/\Theta_{fail}$ in Chapters 3 and 4 is different than the prediction of the satisfaction probability function $p_{sat}(\boldsymbol{\theta})$

in Chapters 5 and 6. Potential differences in verification objectives will challenge the transfer of predictions into later stages.

While these issues seem to discourage the transfer of information between stages, it is also important to highlight the potential improvements in efficiency and accuracy forward transfer of predictions may provide. As shown in Figure 1-2(b), the initial verification stages will be used to prune out poorly-performing control system designs earlier in the process as they will typically rely upon less expensive models. Even if the information is completely discarded between different stages, the initial stages will still improve the efficiency of the total process by avoiding wasted time or resources on those extremely undesirable designs. However, the initial stage's information could still be useful to the later stages as it would indicate regions where the system is expected to have extremely poor performance or vice-versa. Ultimately, multi-stage verification needs to carefully balance the influence of prior information from earlier stages with the potential for discrepancies between the different stages' models.

7.1.1 Forward Transfer with Nonzero Priors

The solution to the challenge of balancing prior information with the potential for discrepancies between models, at least in Chapters 4-6, is informative priors. The approaches in all three of those chapters use nonparametric Gaussian processes to infer the satisfaction of the performance requirements over the full set of possible uncertainties Θ given a limited amount of training data. During the training process, the statistical verification techniques assumed a zero-mean prior $\mathcal{N}(\mathbf{0}, \mathbf{K})$ for the measurements in order to avoid incorrectly biasing the predictions without any prior knowledge. This work replaces that zero-mean prior with the predictive output of the GP in the preceding verification stage as a nonzero-mean prior in order to explicitly incorporate that GP output as prior knowledge. Related work in multi-fidelity [10, 28] and inverse [153] reinforcement learning has demonstrated significant improvements in reinforcement learning policy convergence using nonzero-mean priors. In those problems, the learned policy's cost function from the first model enters as a nonzero-mean prior into the learning of an improved policy on a second model.

This section uses the same general concept, but instead passes predictions of the trajectory robustness measurements in place of reinforcement learning policies.

The key assumption underlying forward transfer in multi-stage verification is that the system and parametric uncertainties under consideration are the same between stages.

Assumption 7.1. *The verification stages all examine the same dynamical system, albeit with different levels of model fidelity, and vary the same parameters* $\boldsymbol{\theta} \in \Theta$.

In short, Assumption 7.1 states that the verification stages must all address the same problem. It makes little sense to exchange predictions between models with drastically different dynamics and operating conditions. For instance, a simulation model of a fixed-wing airplane has extremely little overlap with a simulation model of a car. Likewise, it makes little sense to exchange predictions between stages if they aren't varying the same $\boldsymbol{\theta}$ conditions. The stages need to be relatively similar, although the exact definition of "similar enough" will likely subjectively change with the specific application.

It is possible reuse earlier work with a different requirement to aid the analysis of the system's robustness to a new requirement. Assuming the relevant state trajectory data is available and has been saved, it is straightforward to examine the satisfaction of a new requirement over the course of the past trajectories. However, this reuse of past trajectory data to analyze a new requirement will only be correct when the earlier work varied the same parameters of current interest. Again, Assumption 7.1 ensures the earlier work is actually relevant to the current analysis.

Nonzero-Mean Priors

In both the deterministic and stochastic verification frameworks from Chapters 4 and 5, the trained Gaussian process regression model will output a posterior predictive distribution for the scalar trajectory robustness measurement $y(\boldsymbol{\theta}_*)$ at an arbitrary parameter vector $\boldsymbol{\theta}_*$. With the earlier verification stage considered as "Stage 1", the

GP output from this stage is labeled

$$\mathbb{P}(y(\boldsymbol{\theta}_*)|\mathcal{L}_1, \boldsymbol{\theta}_*, \psi_1) = \mathcal{N}\big(\mu_1(\boldsymbol{\theta}_*), \Sigma_1(\boldsymbol{\theta}_*)\big), \tag{7.1}$$

where $\mathcal{L}_1$ is the training dataset in Stage 1, ψ_1 is the learned kernel hyperparameters, $\mu_1(\boldsymbol{\theta}_*)$ is the predictive mean at $\boldsymbol{\theta}_*$, and $\Sigma_1(\boldsymbol{\theta}_*)$ is the predictive covariance, assuming a deterministic verification problem from Chapter 4. If the closed-loop system is stochastic, the predictions would also include likelihood model hyperparameters ϑ_1 and the predictive covariance for $y(\boldsymbol{\theta}_*)$ would add ϵ_y^2 to $\Sigma_1(\boldsymbol{\theta}_*)$ like in (5.9).

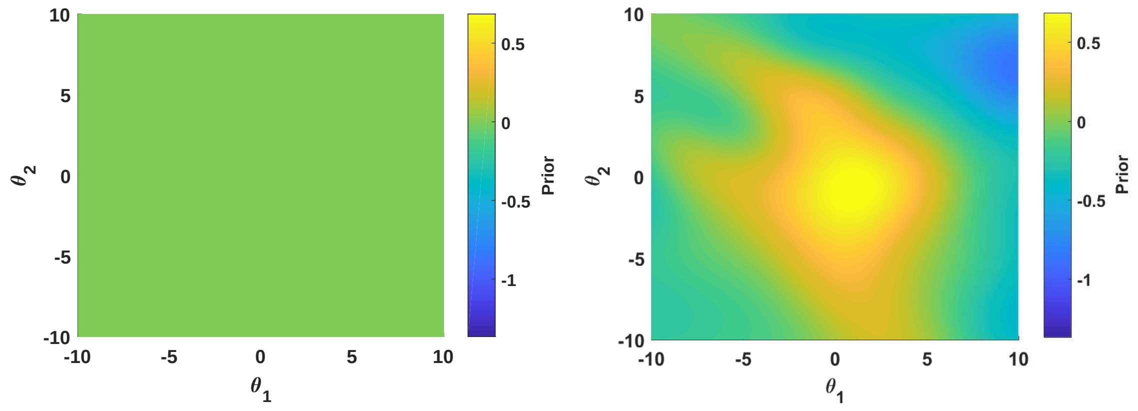
Regardless of whether the problem is deterministic or stochastic, the Gaussian process regression model outputs a posterior predictive mean $\mu_1(\boldsymbol{\theta}_*)$ for the expected robustness measurement $y(\boldsymbol{\theta}_*)$. This predictive mean is directly incorporated into the prediction of $y(\boldsymbol{\theta}_*)$ in the subsequent verification stage, "Stage 2", as a nonzero mean in the prior distribution $\mathcal{N}(\boldsymbol{\mu_1}, \mathbf{K})$. This nonzero mean on the prior will carry through the GP training process and contribute to the posterior predictive distribution for $y(\boldsymbol{\theta}_*)$ in Stage 2. When Stage 2 is a stochastic verification problem, this posterior predictive distribution is written as

$$\mathbb{P}(y(\boldsymbol{\theta}_*)|\mathcal{L}_2, \boldsymbol{\theta}_*, \psi_2, \vartheta_2) = \mathcal{N}\big(\mu_2(\boldsymbol{\theta}_*), \Sigma_2(\boldsymbol{\theta}_*) + \epsilon_y^2\big), \tag{7.2}$$

where the posterior mean and covariance are given by

$$\begin{aligned} \mu_2(\boldsymbol{\theta}_*) &= \mu_1(\boldsymbol{\theta}_*) + \mathbf{K}_*^T(\mathbf{K} + \epsilon_y^2\mathbf{I})^{-1}(\mathbf{y}_2 - \boldsymbol{\mu_1}) \\ \Sigma_2(\boldsymbol{\theta}_*) &= \mathbf{K}_{**} - \mathbf{K}_*^T(\mathbf{K} + \epsilon_y^2\mathbf{I})^{-1}\mathbf{K}_* \end{aligned} \tag{7.3}$$

and $\boldsymbol{\mu_1}$ is $\mu_1(\boldsymbol{\theta})$ evaluated at all the training locations in $\mathcal{L}_2$ corresponding to measurements $\mathbf{y}_2$. The covariance $\Sigma_2(\boldsymbol{\theta}_*)$ is not affected by the earlier predictions. Whenever Stage 1's predictions ($\boldsymbol{\mu_1}$) disagree with Stage 2's observations ($\mathbf{y}_2$), the GP prediction model will incorporate the difference into the predictions for $\mu_2(\boldsymbol{\theta})$ and correct for disagreements between the stages. As desired, this balances the influence of Stage 1's predictions upon the results in Stage 2 with the possibility Stage 1's predictions



(a) Prior with zero mean $y(\boldsymbol{\theta}) \sim \mathcal{N}(\mathbf{0}, \mathbf{K})$ (b) Prior with nonzero mean $y(\boldsymbol{\theta}) \sim \mathcal{N}(\boldsymbol{\mu}_1, \mathbf{K})$

Figure 7-1: Illustration of priors on $y(\boldsymbol{\theta})$ with zero and nonzero means. The nonzero-mean prior in the right-hand plot is taken from the GP output for $\mu(\boldsymbol{\theta})$ in Figure 5-13(a).

may be incorrect.

Note that Stage 1 was written as a deterministic verification problem while Stage 2 has stochastic measurements. This highlights the fact the predictive mean can be passed between both types of problems without modification. Additionally, the problem also labeled the kernel hyperparameters ψ_1 and ψ_2 because they do not necessarily have to agree and the same holds for ϑ_1 and ϑ_2. Given their respective training sets $\mathcal{L}_1$ and $\mathcal{L}_2$, the hyperparameter optimization process may choose a different set of hyperparameters for ψ_2. However, it may be advantageous to also use ψ_1 as a prior to ψ_2 during the hyperparameter optimization process to improve convergence if that is required. Differences between (ψ_1, ψ_2) and $(\vartheta_1, \vartheta_2)$ will have no effect on $\mu_1(\boldsymbol{\theta})$ as the predictive mean $\mu_1(\boldsymbol{\theta})$ is not changed and will be kept constant regardless of (ψ_2, ϑ_2).

Figure 7-1 illustrates the difference between zero- and nonzero-mean priors on the CL-MRAC example from Section 5.6.1. The zero-mean prior in Figure 7-1(a) provides no initial information about the values of $y(\boldsymbol{\theta})$ over the compact set Θ. All the points in Θ have the same value. In comparison, the posterior predictive output from one of the trained Gaussian processes can be used as a nonzero-mean prior, seen in Figure 7-1(b). This nonzero mean transfers the first GP's prediction as the prior predictive distribution for $y(\boldsymbol{\theta})$, which results in the visible difference between the two plots.

Limitations of Informative Priors

Up to this point, the discussion in this section has only considered the two problems in Chapters 4 and 5, but the same process can be applied to the problems addressed in Chapter 6. In those problems, the predictions for the latent function $h(\boldsymbol{\theta}_*)$ would be passed from Stage 1 to Stage 2. The use of the latent function and binary measurements precludes mixing problems from Chapter 6 with the other two problems, at least with the current modeling approaches. In general, this does not pose a significant restriction since Chapter 6 bases the predictions upon binary measurements while Chapters 4 and 5 rely upon continuous measurements. Most multi-stage problems will use a consistent measurement scheme if predictions will be passed between stages.

Similarly, it is also important to point out that the approach in (7.3) does not include any contribution from Stage 1's predictive covariance $\Sigma_1(\boldsymbol{\theta})$, only the predictive mean $\mu_1(\boldsymbol{\theta})$. The absence of $\Sigma_1(\boldsymbol{\theta})$ in (7.3) does present a limitation to the indicated approach and a more complex formulation is necessary to incorporate $\Sigma_1(\boldsymbol{\theta})$ directly into the predictions in Stage 2. The future work section in Chapter 8 describes a number of recent developments in multi-fidelity reinforcement learning and inference techniques with the potential to address this limitation; however, they require a significantly more complex approach than the straightforward method in (7.3).

Although it is not a solution to the issue, the typical utilization of multi-stage approaches minimizes the impact of the absence of $\Sigma_1(\boldsymbol{\theta})$ in (7.3). The common motivation behind multi-stage verification is to minimize the reliance upon costly higher-fidelity models in the later stages and perform more analysis in the earlier stages with less-expensive models. This preference for lower-fidelity models means Stage 1 involves significantly more simulations than Stage 2. Although a larger number of datapoints in Stage 1 does not explicitly guarantee $\Sigma_1(\boldsymbol{\theta})$ will be small, an intelligent dispersion of these datapoints across Θ_d would generally translate to lower $\Sigma_1(\boldsymbol{\theta})$ values. If the concern over $\Sigma_1(\boldsymbol{\theta})$ is great enough, Stage 1 could utilize a different active sampling approach, such as the PDF variance-based procedures, to minimize $\Sigma_1(\boldsymbol{\theta})$. Again, these do not explicitly address the absence of $\Sigma_1(\boldsymbol{\theta})$ in the second stage's predictions, but they do discuss the overall impact of it. The future work in Chapter 8 identifies extensions to directly solve that issue.

7.2 Impact of Failures in Experimental Testing

In many multi-stage verification problems, the last stage of the process is verification using real-world prototypes and experimental testing. For example, in unmanned aerial vehicles this would involve a number of flight tests of the vehicle at different operating conditions. Due to the time and effort spent to setup and perform just one experiment, it is desirable to minimize the number of experiments needed to accurately predict whether the closed-loop system satisfies a performance requirement. This further motivates careful selection of training experiments, to an even greater extent than in simulation-based verification. However, this section discusses an entirely new set of issues encountered during many real-world experiments that are not experienced in simulation-based verification.

During verification of safety requirements for physical systems such as aircraft, cars, or robots, the cost of a trajectory that fails to satisfy the requirement may far exceed the cost of a trajectory that does. For instance, a simple safety requirement for an aircraft is Federal Aviation Regulations (FAR) Part 25.333, pictured in Figure 7-2. This requirement states the aircraft must avoid the maneuvering or never-exceed speeds in the flight envelope during various maneuvers or else the aircraft risks partial or total structural failure. In the best case where the aircraft fails to meet FAR 25.333, the aircraft will only suffer light damage and require either a complete overhaul or scrapping once the vehicle lands. In the worst case, the aircraft will break apart and be total destroyed. This unfortunate result is what happened with the NASA Helios aircraft shown in Figure 1-1 when it exceeded its operating limitations. Regardless of the specific outcome, the cost of a trajectory that fails to satisfy FAR 25.333 surpasses the cost of a trajectory that stayed within the maneuvering envelope.

Similarly, experiment-based verification will typically have access only to a limited

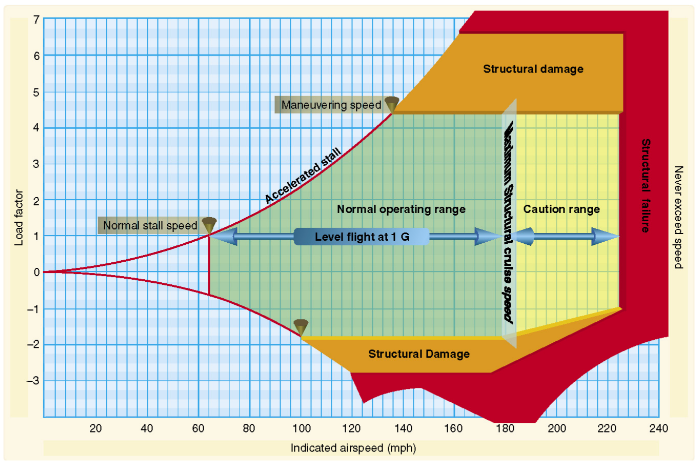


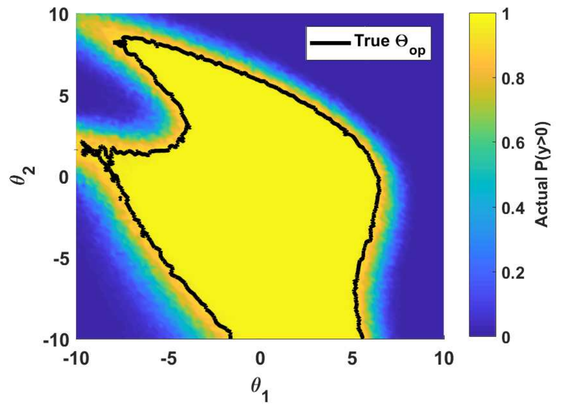
Figure 7-2: An example of a safety requirement for an aircraft with unequal costs of satisfactory and unsatisfactory trajectories. Federal Aviation Regulations (FAR) Part 25.333 requires an aircraft to avoid maneuvering and never-exceed speeds or else the aircraft will experience structural damage or failure (orange and red regions). Image source: [157]

number of prototypes or other testing objects. For instance, there may only be a handful of aircraft, cars, or robots that can be used for the experiments. This becomes a major limitation when coupled with safety requirements like those discussed in the previous paragraph. If a vehicle or testing object is damaged, destroyed, or otherwise unfit-for-use after a trajectory fails to meet the requirement, then it can no longer be used for further experiments. If multiple experiments fail to meet the requirements and result in loss of a testing object, then it is possible to completely exhaust the supply of testing objects and thus stop the experiment-based verification process altogether.

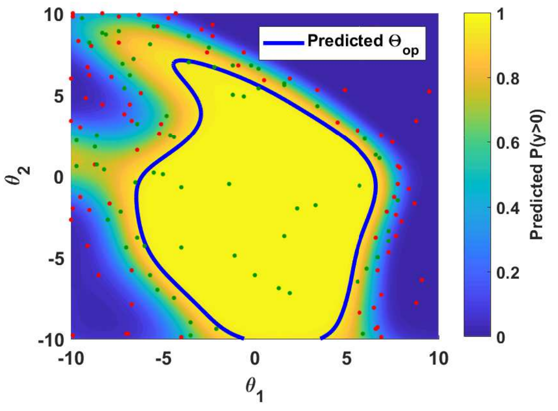
7.2.1 Region of Safe Operation

The asymmetric cost of unsafe trajectories leads to a new objective for verification. Assuming there will always be a nonzero probability of failure at some point in set Θ, the physical system will only operate within the set of parameter values with a minimum probability of satisfying the requirement. This defines a new set of possible parameters, the region of safe operation Θ_{op}.

Definition 7.2. *The* region of safe operation Θ_{op} *contains all* $\boldsymbol{\theta} \in \Theta$ *for which an arbitrary experimental test initialized at* $\boldsymbol{\theta}$ *has a minimum probability of satisfying*



(a) True $p_{sat}(\boldsymbol{\theta})$ and Θ_{op}



(b) Predictions $\widehat{p}_{sat}(\boldsymbol{\theta})$ and $\widehat{\Theta}_{op}$

Figure 7-3: Illustration of a true region of safe operation Θ_{op} and a data-driven prediction $\widehat{\Theta}_{op}$. This figure is derived from the stochastic CL-MRAC example in Section 5.6.1.

the requirement, $p_{min} \in (0, 1]$,

$$\Theta_{op} := \Big\{\boldsymbol{\theta} \in \Theta : p_{sat}(\boldsymbol{\theta}) \geq p_{min}\Big\}. \tag{7.4}$$

This region of safe operation applies to both stochastic systems and deterministic systems, although generally most real-world systems will have some form of stochasticity present in the dynamics. For example, consider the stochastic CL-MRAC system from Section 5.6.1. If the minimum probability of success is $p_{min} = 0.95$, the resulting Θ_{op} is shown in Figure 7-3(a). Notice that the region of safe operation is very similar to the region of satisfaction Θ_{sat} from Chapters 3 and 4. In fact, the region of safe operation is actually the region of satisfaction, $\Theta_{op} = \Theta_{sat}$, in deterministic systems since the true probability of satisfaction at arbitrary $\boldsymbol{\theta}$ is either 0 or 1.

Regardless of the presence of stochasticity, the physical system will attempt to remain within Θ_{op}. This assumes the $\boldsymbol{\theta}$ parameters are known during the experiments and controllable before and during execution of the trajectory. For example, an aircraft's weight and C.G. parameters can be accurately approximated through careful tracking of the current fuel and payload. Given a particular weight and C.G. setting, if the aircraft will not fall within Θ_{op} for some requirement, then the payload would be rearranged or the flight would not occur in the first place. Even wind conditions can be measured using meteorological tools.

Just like the other verification problems, the challenge with Θ_{op} is that the actual set is unknown in advance. The region of safe operation must be estimated using a statistical verification framework and a limited amount of experimental data. This estimated region of safe operation is labeled $\widehat{\Theta}_{op}$ and is determined by the predicted probability of satisfaction $\widehat{p}_{sat}(\boldsymbol{\theta})$,

$$\widehat{\Theta}_{op} := \Big\{\boldsymbol{\theta} \in \Theta : \widehat{p}_{sat}(\boldsymbol{\theta}) \geq p_{min}\Big\}, \tag{7.5}$$

shown in Figure 7-3(b). Figure 7-3 also illustrates the root of the verification problem, prediction error between $\widehat{\Theta}_{op}$ and true Θ_{op}. The ultimate objective of a closed-loop statistical verification algorithm is to minimize the prediction error and have $\widehat{\Theta}_{op} \to \Theta_{op}$ while subject to a limit on the number of experiments. However, these systems also have the restriction on the number of testing objects to consider. This restriction factors into the verification problem as a constraint on the number of failed experiments. More precisely, given the set of all parameter vectors for the experiments, set $\mathcal{D}$, subset $\mathcal{D}_{fail} \subset \mathcal{D}$ contains all parameters for which the corresponding trajectory failed to meet the safety requirement, i.e.

$$\mathcal{D}_{fail} := \Big\{\boldsymbol{\theta} \in \mathcal{D} : y(\boldsymbol{\theta}) \leq 0\Big\}. \tag{7.6}$$

The failure constraint states the size of $\mathcal{D}_{fail}$ must be strictly less than the maximum number of allowable failures N_{fail}, meaning $|\mathcal{D}_{fail}| < N_{fail}$. As the next section will discuss, this new failure constraint severely restricts the suitability of the active sampling algorithms discussed in the previous chapters.

7.2.2 Problem with Trajectory Robustness Measurements

Failures also introduce an additional challenge to statistical verification. While it is correct to assume safe trajectories which satisfy the safety requirement provide the minimum level of trajectory robustness as measurement $y(\boldsymbol{\theta})$, unsafe trajectories will not necessarily be able to provide the true minimum level of robustness, as was possible in simulations. For instance, if the requirement states "'the quadrotor must stay 1 foot away from the obstacle," safe trajectories will be able to provide the robustness value corresponding to the minimum distance between the vehicle and the obstacle. Unsafe trajectories which break this 1 foot window but still avoid the obstacle will also return the true minimum distance. However, if the quadrotor behaved erratically and actually flew into the obstacle and crashed, then the observed robustness value for this failed trajectory will be $y(\boldsymbol{\theta}) = -1$, meaning the distance between the quadrotor and obstacle was 0 ft (a collision). Assuming the quadrotor crashes with every collision, any trajectory which flies into the obstacle will return this same $y(\boldsymbol{\theta}) = -1$ since the trajectory will stop after the collision and subsequent crash. The problem is the robustness value of $y(\boldsymbol{\theta}) = -1$ does not indicate the severity of the collision; it does not delineate between a glancing blow and when the quadrotor flew full speed straight into the obstacle. In a simulation-based environment, it is possible to allow the simulator to continue the trajectory after a "collision," which allows the simulator and the resulting trajectory robustness measurements to identify the severity of the failure in unsafe trajectories. This problem with experimental failures and $y(\boldsymbol{\theta})$ will effect any statistical verification problem using non-binary measurements of trajectory robustness.

Although nothing can be done about the fact trajectory robustness measurements will stop once the vehicle collides with an obstacle, there are a number of experimental constraints and workarounds to allow data-driven statistical verification to still take place. The most obvious solution is to avoid failures altogether. Safe experimental trajectories which satisfy the requirement still provide the true robustness measurements and place no special considerations on data-driven verification. Therefore, it is best to avoid failures not just for their asymmetric cost, but also because they complicate data-driven verification. While it is impossible to know where the failures exist without any prior knowledge, modifications to the problem like the 1 foot buffer around obstacles will help avoid total failures, i.e. crashes, since not all unsafe trajectories correspond to a collision with the obstacle and premature termination of the robustness measurements.

In reality, there will always be a chance of failure, so data-driven statistical verification requires some workarounds for when the trajectory data stops once a failure is encountered. One solution is to augment the experimental trajectory data with simulation data. As mentioned earlier, it is possible in simulation-environments to continue the trajectory after a failure is encountered in order to compute the severity of the unsafe trajectory. When an experiment encounters a failure and stops, the stream of trajectory data up to that failure could be augmented with a continuation of the trajectory in the simulated world. This hybrid trajectory is not ideal, but does allow a data-driven statistical framework to incorporate some estimate of the true severity of a failed trajectory. If this hybrid approach is not practical, then it is still possible to incorporate the raw $y(\boldsymbol{\theta})$ from the experimental data or replace it with an artificial estimate of the severity. None of these completely solve the known issue, but do offer practical workarounds if such considerations are necessary. The next section will develop failure-adverse statistical verification frameworks which may indirectly sidestep this issue altogether by avoiding failures, but these workarounds will be used when failures are encountered.

7.3 Failure-Adverse Closed-Loop Verification

The new region of safe operation Θ_{op} and particularly the constrained number of allowable failures N_{fail} will greatly affect the accuracy of the existing active sampling procedures discussed in the earlier chapters. When the number of failures is factored in, the previous algorithms will demonstrate extremely poor prediction accuracy in $\widehat{\Theta}_{op}$. This poor performance motivates the development of a related, but distinct, closed-loop statistical verification framework to compute $\widehat{\Theta}_{op}$ while subject to constraint N_{lim}. For simplicity and because real-world systems are more likely to be stochastic, this section uses the stochastic Gaussian process prediction model from Section 5.2. The deterministic GP-based framework from Section 4.2 can also be used with minimal modifications. It is technically feasible to implement this work on the stochastic framework from Chapter 6, but the need for multiple binary measurements at each parameter vector $\boldsymbol{\theta} \in \mathcal{D}$ would quickly drive up the experimental cost with the large number of trajectories.

Limitations of Previous Methods

The existing sampling procedures discussed in the previous chapters suffer from one small and one large limitation. First, since the verification objective has shifted to the maximization of the accuracy of $\widehat{\Theta}_{op}$, the previous algorithms are no longer ideally suited to the verification objective. This fact is not surprising and the results in Section 7.4 will show the effect is minimal. The real issue is the large number of failures those sampling strategies will produce. For instance, consider the training data in Figure 7-3(b). The red dots indicate trajectories which failed to satisfy the requirement and these failures constitute roughly half of the training data. If this training dataset corresponds to experimental data, then the number of failures would quickly exceed a small N_{fail}. Once the number of failures exceeds N_{fail}, then the algorithm would prematurely terminate as it ran out of available testing objects.

7.3.1 Forward Transfer of Simulation-Based Predictions

The previous active sampling algorithms in the earlier chapters require an initial training set of passively-selected training locations to generate an initial GP model. This fact will only increase the number of failures as many of the resulting trajectories will inevitably fail to satisfy the requirement. A partial solution is to forward transfer predictions from the preceding verification stage to remove the necessity of a passively-selected training dataset $\mathcal{L}$. This implicitly assumes experiment-based verification is the last stage in a multistage verification process, but that will be typical of any failure-constrained verification problem; it is safe to assume the engineers won't blindly jump into experimental testing without extensively studying the closed-loop system in simulation environments. The prior predictions provided by the simulation-based verification stage(s) identify regions of Θ where the closed-loop system will likely satisfy the requirement.

The simulation predictions will guide the selection of a small initial training set before any experiments are actually performed. The posterior predictive output from the preceding simulation-based verification stage is labeled as mean $\mu_{sim}(\boldsymbol{\theta})$, covariance $\Sigma_{sim}(\boldsymbol{\theta})$, and predicted satisfaction probability function $\widehat{p}_{sim}(\boldsymbol{\theta})$. Note the dropping of subscript "sat" in the predicted satisfaction probability function due to sizing and to avoid potential confusion with the experimental function. Similarly, the subscript "rw" for "real world" is used to differentiate measurements and predictions in the experimental verification stage from the simulations. Assuming zero experiments have been performed, it is unclear at which locations $\boldsymbol{\theta} \in \Theta$ the first experiments should be performed. Active sampling cannot be performed since there are no actual measurements with which to construct a posterior predictive distribution. Therefore, the only available information is provided by the prior $\mu_{sim}(\boldsymbol{\theta})$, $\Sigma_{sim}(\boldsymbol{\theta})$, and $\widehat{p}_{sim}(\boldsymbol{\theta})$.

The obvious choice for the initial training measurements are those points in Θ with $\widehat{p}_{sim}(\boldsymbol{\theta})$ closest to 1. However, since the simulation model might fail to perfectly capture the real-world dynamics, $\widehat{p}_{sim}(\boldsymbol{\theta})$ might not match the true, real-world satisfaction probability function $p_{rw}(\boldsymbol{\theta})$. In order to minimize the likelihood of failure, the first experiment should be performed at the *most robust* parameter vector according to the simulations. The most robust parameter vector is given by

$$\overline{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}\in\Theta}{\operatorname{argmax}} \left(\mu_{sim}(\boldsymbol{\theta}) - \beta_z\sqrt{\Sigma_{sim}(\boldsymbol{\theta}) + \epsilon_y^2}\right), \tag{7.7}$$

where β_z is the z-score associated with the minimum probability of success p_{min} that defines Θ_{op}. The motivation for selection criterion (7.7) in place of $\widehat{p}_{sim}(\boldsymbol{\theta})$ directly is illustrated in Figure 7-4. The cumulative distributions at points $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ are roughly equivalent, at least to numerical precision, with $\widehat{p}_{sim}(\boldsymbol{\theta}) = 1$ for both indicated probability density functions. However, the cumulative distribution at $\boldsymbol{\theta}_2$ is much more robust to changes in $\bar{y}(\boldsymbol{\theta})$ than the distribution at $\boldsymbol{\theta}_1$. Unmodeled dynamics and effects present in the real-world system but not the simulation model will manifest as changes in the distributions, such as an increase or decrease in $\bar{y}(\boldsymbol{\theta})$. Without any

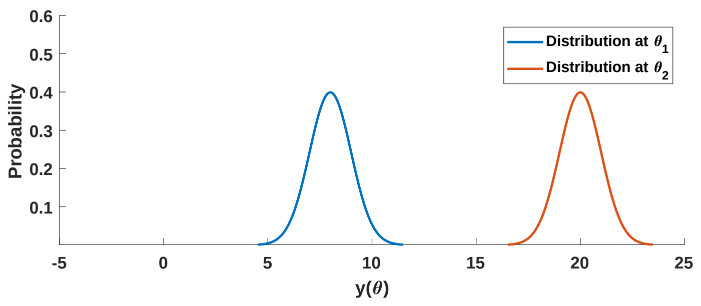


Figure 7-4: Motivation for the selection of the most robust point according to the simulation predictions. Both $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ have effectively the same $\widehat{p}_{sim}(\boldsymbol{\theta}) = 1$ given the indicated probability density functions. However, the cumulative distribution at $\boldsymbol{\theta}_2$ is much less sensitive to shifts in $\bar{y}(\boldsymbol{\theta})$ than the distribution at $\boldsymbol{\theta}_1$.

prior knowledge, it is impossible to predict the exact difference between the simulation model and real-world experiments, but (7.7) uses all available information to avoid a failure during the first experiment. If it is necessary to perform multiple experiments, then (7.7) can be used as the basis for some batch selection process like importance-weighted sampling or k-DPPs.

Once the first experiment has been performed, the resulting robustness measurement $y_{rw}(\overline{\boldsymbol{\theta}})$ can be combined with the simulation priors to compute the posterior predictive distribution for all $\boldsymbol{\theta} \in \Theta$. As in Section 7.1, the simulation-based predictive mean $\mu_{sim}(\boldsymbol{\theta})$ will act as an informative, nonzero prior to the experimental-based predictions. The resulting posterior predictive distribution for experimental measurement $y_{rw}(\boldsymbol{\theta}_*)$ is then written just like (7.2) and (7.3) with

$$\mathbb{P}(y_{rw}(\boldsymbol{\theta}_*)|\mathcal{L}_{rw}, \boldsymbol{\theta}_*, \psi, \vartheta) = \mathcal{N}\Big(\mu_{rw}(\boldsymbol{\theta}_*), \Sigma_{rw}(\boldsymbol{\theta}_*) + \epsilon_y^2\Big) \tag{7.8}$$

and

$$\begin{aligned} \mu_{rw}(\boldsymbol{\theta}_*) &= \mu_{sim}(\boldsymbol{\theta}_*) + \mathbf{K}_*^T(\mathbf{K} + \epsilon_y^2\mathbf{I})^{-1}(\mathbf{y}_{rw} - \boldsymbol{\mu}_{sim}) \\ \Sigma_{rw}(\boldsymbol{\theta}_*) &= \mathbf{K}_{**} - \mathbf{K}_*^T(\mathbf{K} + \epsilon_y^2\mathbf{I})^{-1}\mathbf{K}_* \ . \end{aligned} \tag{7.9}$$

The training dataset $\mathcal{L}_{rw}$ contains the initial measurement and the measurements from any subsequent experiments. At least during the first few experiments, there will not be enough training data to adequately optimize the hyperparameters, so kernel and likelihood hyperparameters ψ and ϑ should be copied from the simulation-based

GP regression model. Once the first experiment has been performed and posterior predictive distributions are available for all $\boldsymbol{\theta} \in \Theta$, active sampling can be used to simultaneously minimize the number of failures and improve the accuracy of $\widehat{\Theta}_{op}$, all while still limited to a total number of experiments N_{total}.

7.3.2 Selection Criteria

Given the posterior prediction distribution for $y_{rw}(\boldsymbol{\theta})$, active sampling will help minimize the number of failures and maximize the accuracy of the predicted safe operating region $\widehat{\Theta}_{op}$. The following subsection will first introduce an additional constraint to minimize the likelihood of failures. This constraint can be applied to any of the previously discussed active sampling strategies and is not restricted to certain selection criteria. While this helps avoid failures in the experiments, it does not explicitly help maximize the accuracy of $\widehat{\Theta}_{op}$ and neither do the previous selection metrics. The second portion of this subsection will develop new sample selection criteria for active sampling algorithms that do explicitly attempt to improve $\widehat{\Theta}_{op}$.

Restricted Search Area

The first modification to closed-loop statistical verification is the addition of a constraint on the set of possible parameter settings for future experiments. Rather than select $\boldsymbol{\theta}$ from the set of all unseen parameter values $\mathcal{U}$, future training locations are chosen from a subset of $\mathcal{U}$ whose resulting trajectories are expected have a minimum level of robustness to the requirement, similar to (7.7). This restriction on $\mathcal{U}$ says nothing about actual selection criterion itself and can be applied to any of the existing sample selection criteria. For instance, the search area restriction applied to the CDF variance reduction selection criterion from (5.24) will appear as a constraint,

$$\overline{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}\in\mathcal{U}}{\operatorname{argmax}}\ \widetilde{V}(\boldsymbol{\theta}|\mathcal{L}_{rw}, \psi, \vartheta) \tag{7.10}$$

$$\text{s.t.} \quad \left(\mu_{rw}(\boldsymbol{\theta}) - \widehat{\beta}_z\sqrt{\Sigma_{rw}(\boldsymbol{\theta}) + \epsilon_y^2}\right) > y_{min}, \tag{7.11}$$

where (7.10) is the original CDF variance reduction selection metric and (7.11) is the new constraint. Only those $\boldsymbol{\theta} \in \mathcal{U}$ which satisfy (7.11) are eligible for selection for the next experiment. In order to write (7.10) and (7.11) more concisely, the restricted search area can be written as set $\mathcal{U}_{sa}$, where

$$\mathcal{U}_{sa} := \Big\{\boldsymbol{\theta} \in \mathcal{U} : \Big(\mu_{rw}(\boldsymbol{\theta}) - \widehat{\beta}_z \sqrt{\Sigma_{rw}(\boldsymbol{\theta}) + \epsilon_y^2}\Big) > y_{min}\Big\}. \tag{7.12}$$

Instead of writing the constraint in terms of a minimum probability of satisfaction like Definition 7.2, the search constraint (7.11) is defined by $\mu_{rw}(\boldsymbol{\theta})$ and $\Sigma_{rw}(\boldsymbol{\theta})$ and two new terms y_{min} and $\widehat{\beta}_z$ to provide more flexibility. Term $\widehat{\beta}_z$ is the z-score for a desired confidence level while y_{min} is a minimum acceptable robustness value in the experiments. These two terms allow the control engineers to set different probabilities and robustness levels for experiment-based verification than the region of safe operation Θ_{op} used by a production-ready system. For instance, experiment-based verification may accept a greater probability of failure than would be used for Θ_{op} and $\widehat{\beta}_z$ would be smaller than β_z corresponding to p_{min}. Since the restricted search area does limit the set of possible sample locations, a looser $\widehat{\beta}_z$ would allow the active sampling algorithm to explore riskier, but potentially more informative, regions. Similarly, the engineers may allow a different level of robustness in experiments than the final product. For example, a safety requirement may again state "the quadrotor must stay 1 foot away from every obstacle," but this distance may be relaxed to 6 inches during the experiments for efficiency or prediction accuracy and hence $y_{min} = -0.5$, assuming $y(\boldsymbol{\theta})$ is given in feet. Section 7.3.3 will explore these concepts further. If this flexibility is not needed, then $\widehat{\beta}_z = \beta_z$ and $y_{min} = 0$ will reproduce $\widehat{\Theta}_{op}$ as the restricted search area.

Expected Model Increase

Given the initial experiment selected by (7.7) and the restricted search area (7.11), the active sampling process incrementally expands $\widehat{\Theta}_{op}$. As was seen before, existing selection criteria which are not ideally-suited to the current verification objective may choose uninformative $\boldsymbol{\theta}$ vectors which do little to improve $\widehat{\Theta}_{op}$. In the worst-case scenario, the selection criteria may inadvertently waste experiments and as a result $\widehat{\Theta}_{op}$ will never expand outwards. This motivates the development of a new selection criterion specifically aimed at maximizing the incremental expansion of $\widehat{\Theta}_{op}$ with each additional experiment.

The main idea is to maximize the difference in posterior safe operating region with the new training sample versus the current prediction. More specifically, this selection metric would select the best location $\overline{\boldsymbol{\theta}}$ as

$$\overline{\boldsymbol{\theta}} = \operatorname*{argmax}_{\boldsymbol{\theta} \in \mathcal{U}_{sa}} \Big(|\widehat{\Theta}_{op}^{+}| - |\widehat{\Theta}_{op}| \Big), \tag{7.13}$$

where posterior $\widehat{\Theta}_{op}^{+}$ is the predicted set after an experiment is performed at location $\boldsymbol{\theta}$ and $\mathcal{L}_{rw}^{+} = \mathcal{L}_{rw} \cup \{\boldsymbol{\theta}, y_{rw}(\boldsymbol{\theta})\}$. However, the true robustness of the trajectory is unknown in advance and thus (7.13) is not feasible. Instead, the *expected* posterior set $\mathbb{E}[\widehat{\Theta}_{op}^{+}]$ replaces the infeasible $\widehat{\Theta}_{op}^{+}$. The expected posterior set $\mathbb{E}[\widehat{\Theta}_{op}^{+}]$ is defined much like (7.5),

$$\mathbb{E}[\widehat{\Theta}_{op}^{+}] := \Big\{ \boldsymbol{\theta} \in \Theta : \mathbb{E}[\widehat{p}_{sat}^{+}(\boldsymbol{\theta})] \geq p_{min} \Big\}, \tag{7.14}$$

where $\mathbb{E}[\widehat{p}_{sat}^{+}(\boldsymbol{\theta})]$ is the expected posterior for the predicted satisfaction probability function if a measurement is taken at location $\boldsymbol{\theta}$. This expected satisfaction probability function is computed by

$$\mathbb{E}[\widehat{p}_{sat}^{+}(\boldsymbol{\theta})] = \frac{1}{2} + \frac{1}{2}\,\mathrm{erf}\bigg(\frac{\mathbb{E}[\mu(\boldsymbol{\theta})^{+}]}{\sqrt{2(\Sigma(\boldsymbol{\theta})^{+} + \epsilon_y^2)}} \bigg). \tag{7.15}$$

Since the measurement at $y_{rw}(\boldsymbol{\theta})$ is unknown in advance, the expected posterior mean $\mathbb{E}[\mu(\boldsymbol{\theta})^{+}]$ is found by replacing $y(\boldsymbol{\theta})$ with its current predictive mean $\mu(\boldsymbol{\theta})$ and recomputing the GP. Fortunately, the covariance $\Sigma(\boldsymbol{\theta})^{+}$ is independent of the actual measurement and can be determined in advance. The resulting selection metric is given by

$$\overline{\boldsymbol{\theta}} = \operatorname*{argmax}_{\boldsymbol{\theta} \in \mathcal{U}_{sa}} \Big(\big|\mathbb{E}[\widehat{\Theta}_{op}^{+}]\big| - |\widehat{\Theta}_{op}| \Big) \tag{7.16}$$

and is referred to as the *expected model increase* (EMI) criterion since it aims to maximize the outward expansion of $\widehat{\Theta}_{op}$.

Although (7.16) is tailor-made for the experiment-based verification problem with predicted $\widehat{\Theta}_{op}$, it does have a number of limitations. Mainly, the selection metric requires the current GP to be recomputed for every prospective $\boldsymbol{\theta}$ vector in order to obtain its expected posterior $\mathbb{E}[\widehat{\Theta}_{op}^{+}]$. The previous GP-based selection criteria from Chapters 4 and 5 avoided those types of metrics due to the high computational cost of retraining the GP at every possible point. However, there are two considerations that help lessen the impact of the cost of retraining the GP at every prospective sample location during experimental verification. First, the restricted search area removes a number of the available sample locations $\mathcal{U}$. Unlike the earlier problems where the selection criteria would select $\boldsymbol{\theta}$ vectors from the entire set $\mathcal{U}$, the restricted search area limits $\boldsymbol{\theta}$ to a subset of $\mathcal{U}$. This means the retraining occurs at fewer points and therefore the total cost is lower than it was in the earlier algorithms. Additionally, most applications in experiment-based verification will tolerate higher computational costs in order to maximize the return-on-investment for each trajectory. In general, the maximum number of experiments will be significantly lower than the number of simulations in the preceding stages. This means the computational cost of retraining the GP will be lower since the training dataset $\mathcal{L}_{rw}$ is smaller and also forces the implicit value of each experiment to increase. Since the monetary and temporal cost of an experiment is higher than a simulation, it will be imperative to maximize the informativeness of every experiment. If computational cost is still relatively important, then new versions of the earlier sampling algorithms modified with the restricted search area $\mathcal{U}_{sa}$ can be used in place of (7.16).

7.3.3 Sampling Algorithms

The restricted search area and the EMI selection metric from (7.16) lead to new active sampling algorithms for failure-adverse closed-loop statistical verification. This subsection will discuss two approaches to active sampling, one with a static parameters and one that adapts the search area parameters based upon the number of failures encountered.

Static Search Area Parameters

The simplest approach assumes the parameters $\widehat{\beta}_z$ and y_{min} are fixed and given at the start of the process. While the predicted $\widehat{\Theta}_{op}$ will change with new measurements, the search area criteria will not change according to the progress of the experiments. Algorithm 12 lists the steps in the sequential failure-adverse sampling procedure. The algorithm assumes the required inputs have been provided by the controls engineer and simulation priors $\mu_{sim}(\boldsymbol{\theta})$ passed from the preceding simulation stage (Step 1). The algorithm also assumes the initial experiment has been obtained by (7.7) and uses the single experimental measurement $y_{rw}(\boldsymbol{\theta})$ to construct the initial GP model for the real-world stage with predictions $\mu_{rw}(\boldsymbol{\theta})$, $\Sigma_{rw}(\boldsymbol{\theta})$, and $\widehat{\Theta}_{op}$ (Step 2). The algorithm will then select additional training locations and perform experiments at those locations until either the sampling budget $T = N_{lim} - 1$ has been met or the procedure runs out of available testing objects, where N_{fail} is the number of testing objects and thus the maximum number of allowable failures.

Steps 3-15 contain the active sampling process. At the start of each iteration, the procedure constructs the current search area $\mathcal{U}_{sa}$ based upon the GP model and provided search parameters (Step 4). In Step 5, the algorithm selects the best prospective sample location $\overline{\boldsymbol{\theta}}$ from the restricted search area according to the EMI criterion (7.16) and then performs an experiment there during Step 6. Steps 7-9 have no bearing on the actual selection process and merely keep track of the number of failures encountered. After the experimental test has concluded, the procedure adds the resulting measurement $y_{rw}(\overline{\boldsymbol{\theta}})$ to the training set $\mathcal{L}_{rw}$ (Step 10) and retrains the GP regression model (Step 11). If the process has exhausted its supply of testing objects, then it immediately terminates (Steps 12-14). Otherwise, the sequential procedure terminates once all T experiments have been performed. Regardless of the exact reason for stopping, the algorithm returns the predicted region of safe operation $\widehat{\Theta}_{op}$ at the conclusion of the iterative process (Step 16).

Algorithm 12 Sequential failure-adverse closed-loop verification framework with static search area parameters

```
1:  **Input:** simulation prior $\mu_{sim}(\boldsymbol{\theta})$, max # of experiments $T$, safe operating region parameters $p_{min}$ and $\beta_z$, search area parameters $\widehat{\beta}_z$ and $y_{min}$, max # of allowable failures $N_{fail}$, training dataset $\mathcal{L}_{rw}$, available sample locations $\mathcal{U}$
2:  **Initialize:** train GP model with $\mu_{sim}(\boldsymbol{\theta})$ and $\mathcal{L}_{rw}$, failure count $k_{fail} = 0$
3:  **for** $i = 1, 2, \dots, T$ **do**
4:      Construct search area $\mathcal{U}_{sa} := \left\{\boldsymbol{\theta} \in \mathcal{U} : \left(\mu_{rw}(\boldsymbol{\theta}) - \widehat{\beta}_z\sqrt{\Sigma_{rw}(\boldsymbol{\theta}) + \epsilon_y^2}\right) > y_{min}\right\}$
5:      Select $\overline{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}\in\mathcal{U}_{sa}}{\operatorname{argmax}} \left(\left|\mathbb{E}[\widehat{\Theta}_{op}^{+}]\right| - |\widehat{\Theta}_{op}|\right)$
6:      Run experiment at $\overline{\boldsymbol{\theta}}$, obtain measurement $y_{rw}(\overline{\boldsymbol{\theta}})$
7:      **if** destroyed testing object **then**
8:          Increment $k_{fail}$ by 1
9:      **end if**
10:     Add $\{\overline{\boldsymbol{\theta}}, y_{rw}(\overline{\boldsymbol{\theta}})\}$ to training set $\mathcal{L}_{rw}$, remove $\overline{\boldsymbol{\theta}}$ from $\mathcal{U}$
11:     Retrain GP model with $\mu_{sim}(\boldsymbol{\theta})$ and updated $\mathcal{L}_{rw}$
12:     **if** $k_{fail} \geq N_{fail}$ **then**
13:         **break** for loop
14:     **end if**
15: **end for**
16: **Return:** predicted region of safe operation $\widehat{\Theta}_{op}$
```

Adaptive Search Area Parameters

While Algorithm 12 will run until either the maximum number of tests has been reached or all testing objects have been expended, it does not update the restricted search area to reflect the number of experiments and testing objects remaining. Assuming the underlying goal is to complete all T experiments allocated to the verification procedure, it may be advantageous to adjust the restricted search area based upon the observed results. For instance, if a number of testing objects were lost early on in the process, it will generally be advisable for the procedure to tighten the search area in order to avoid additional failures. On the opposite side of the spectrum, if there are still a large number of testing objects and only a few experiments remaining, it is possible to loosen the search area and consider riskier parameter settings. A search area that varies with the number of experiments and testing objects remaining is called an *adaptive search area.*

The main idea with an adaptive search area is to update the search area parameter

$\widehat{\beta}_z$ to reflect the acceptable probability of failure. For instance, given 6 testing objects ($N_{fail} = 6$) and $T = 100$ experiments to perform, an acceptable probability of failure is 5%. While this expects there to be roughly 5 failures per 100 experiments, it also expects to complete all 100 experiments since there should be 1 object remaining. Obviously, such a high failure rate is unacceptable in many applications, including any tests involving human drivers, pilots, etc. However, higher failure rates would be acceptable in experiments with low-cost unmanned aerial vehicles where a failure might mean the UAV is damaged or destroyed, but can be easily rebuilt for another day's testing.

Algorithm 13 lists the verification procedure with the adaptive search parameter $\widehat{\beta}_z$. The only change with respect to Algorithm 12 is the addition of the computation of $\widehat{\beta}_z$ in Steps 4-6. These steps use the number of experiments remaining (T_r) and testing objects remaining (N_{obj}) to compute the acceptable probability of failure p_{accept}. In turn, this acceptable probability of failure defines a new z-score parameter $\widehat{\beta}_z$ that changes the restricted search area $\mathcal{U}_{sa}$ in Step 7. More complex functions of T_r, N_{obj}, and other relevant variables are possible, but are not considered in this work. Additionally, during implementation of Algorithm 13, it is advisable to bound p_{accept} by minimum and maximum acceptable probabilities to ensure p_{accept} and $\widehat{\beta}_z$ fall within reasonable values as $T_r \rightarrow 1$ or $N_{obj} \rightarrow 1$.

Both Algorithms 12 and 13 are sequential procedures. The majority of the two algorithms share the same computational complexity as the equivalent steps shown previously in Algorithm 3 and 7. However, the EMI search criterion in Step 5 of Algorithm 12 and Step 8 of Algorithm 13 requires substantially more operations than the previous metrics. In order to compute $\mathbb{E}[\widehat{\Theta}_{op}^{+}]$ for each $\boldsymbol{\theta} \in \mathcal{U}_{sa}$, the failure-adverse algorithms must first retrain an approximate GP using artificial measurements $\widehat{y}(\boldsymbol{\theta}) = \mu(\boldsymbol{\theta})$ and compute $\mathbb{E}[\mu(\boldsymbol{\theta})^{+}]$ and $\Sigma(\boldsymbol{\theta})^{+}$. Those computations alone require on the order of $\mathcal{O}((N+1)^2|\mathcal{U}_{sa}|) + \mathcal{O}((N+1)|\mathcal{U}_{sa}|) + \mathcal{O}((N+1)^2) + \mathcal{O}(N+1)$ operations *for every* $\boldsymbol{\theta} \in \mathcal{U}_{sa}$, thus the computational complexity of that single step is actually a quadratic function of the size of $\mathcal{U}_{sa}$. For applications with a large Θ_d, the complexity will quickly rise. Batch versions of the failure-adverse algorithms using importance-

Algorithm 13 Sequential failure-adverse closed-loop verification framework with adaptive search area parameter $\widehat{\beta}_z$

```
1: Input: simulation prior $\mu_{sim}(\boldsymbol{\theta})$, max # of experiments $T$, safe operating region parameters $p_{min}$ and $\beta_z$, search area parameter $y_{min}$, max # of allowable failures $N_{fail}$, training dataset $\mathcal{L}_{rw}$, available sample locations $\mathcal{U}$
2: Initialize: train GP model with $\mu_{sim}(\boldsymbol{\theta})$ and $\mathcal{L}_{rw}$, failure count $k_{fail} = 0$
3: for $i = 1, 2, \ldots, T$ do
4:     Compute number of experiments remaining: $T_r = T - (i-1)$, compute number of testing objects remaining $N_{obj} = N_{fail} - k_{fail}$
5:     Determine acceptable probability of failure $p_{accept} = (N_{obj} - 1)/T_r$
6:     Compute z-score $\widehat{\beta}_z$ corresponding to $1 - p_{accept}$
7:     Construct search area $\mathcal{U}_{sa} := \left\{\boldsymbol{\theta} \in \mathcal{U} : \left(\mu_{rw}(\boldsymbol{\theta}) - \widehat{\beta}_z\sqrt{\Sigma_{rw}(\boldsymbol{\theta}) + \epsilon_y^2}\right) > y_{min}\right\}$
8:     Select $\overline{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \mathcal{U}_{sa}}{\text{argmax}} \left(\left|\mathbb{E}[\widehat{\Theta}_{op}^{+}]\right| - |\widehat{\Theta}_{op}|\right)$
9:     Run experiment at $\overline{\boldsymbol{\theta}}$, obtain measurement $y_{rw}(\overline{\boldsymbol{\theta}})$
10:    if Destroyed testing object then
11:        Increment $k_{fail}$ by 1
12:    end if
13:    Add $\{\overline{\boldsymbol{\theta}}, y_{rw}(\overline{\boldsymbol{\theta}})\}$ to training set $\mathcal{L}_{rw}$, remove $\overline{\boldsymbol{\theta}}$ from $\mathcal{U}$
14:    Retrain GP model with $\mu_{sim}(\boldsymbol{\theta})$ and updated $\mathcal{L}_{rw}$
15:    if $k_{fail} \geq N_{fail}$ then
16:        break for loop
17:    end if
18: end for
19: Return: predicted region of safe operation $\widehat{\Theta}_{op}$
```

weighting and k-DPPs can be formed by constructing a probability distribution from the selection metric (7.16). Unlike the batch algorithms in the earlier chapters, the samples will only be chosen from within the restricted search area $\mathcal{U}_{sa}$.

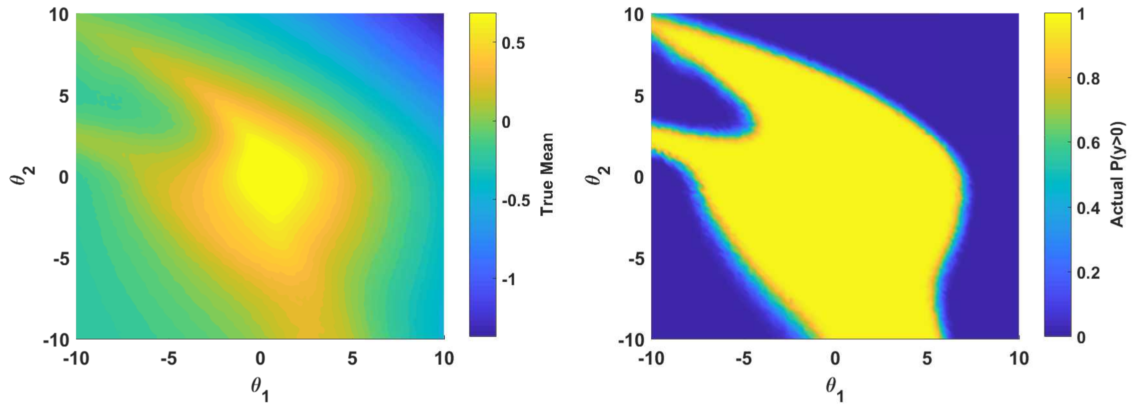
7.4 Demonstration of Failure-Constrained Verification

A stochastic variant of the 2D CL-MRAC example demonstrates the effectiveness of Algorithms 12 and 13 for failure-constrained verification problems. The example uses the stochastic CL-MRAC dynamics from Example 5.6.1 as the simulation model. Figure 7-5 depicts the underlying true $\bar{y}_{sim}(\boldsymbol{\theta})$ and $p_{sim}(\boldsymbol{\theta})$ in the simulation stage. The "real-world" dynamics are given by the high-fidelity stochastic model with higher-order nonlinear dynamics and actuator saturation from Section 7.1. In order to produce an even more challenging shape, the true $\bar{y}_{rw}(\boldsymbol{\theta})$ function is artificially modified, as seen in Figure 7-6(a). The resulting true $p_{sat}(\boldsymbol{\theta})$ in Figure 7-6(b) is substantially different than the true values in the simulation dynamics. Even if $\bar{y}_{sim}(\boldsymbol{\theta})$ and $p_{sim}(\boldsymbol{\theta})$ were perfectly known and used for the simulation prior, they would be of limited use to experiment-based verification due to this discrepancy. Lastly, unsafe trajectories that failed to satisfy the requirements are allowed to continue past the point of failure. Although this is not a perfect representation of all failure-constrained problems where the trajectory may stop once a failure is reached, the workarounds discussed in Section 7.2.2 could be applied. The end goal of this example is not to demonstrate the impact of failure on the measurements, but rather demonstrate the effectiveness of the closed-loop algorithms to avoid failures altogether.

The verification objective is to estimate the safe operating region Θ_{op} for all $\boldsymbol{\theta} \in \Theta$ with a minimum probability of satisfaction of 95% ($p_{min} = 0.95$). The safe operating region is shown in Figure 7-6(b). The rest of the problem is mostly unchanged from the other 2D CL-MRAC examples. The set of all possible parameters Θ is approximated with a finite grid Θ_d of 40,401 locations between $\theta_1 : [-10, 10]$ and $\theta_2 : [-10, 10]$.

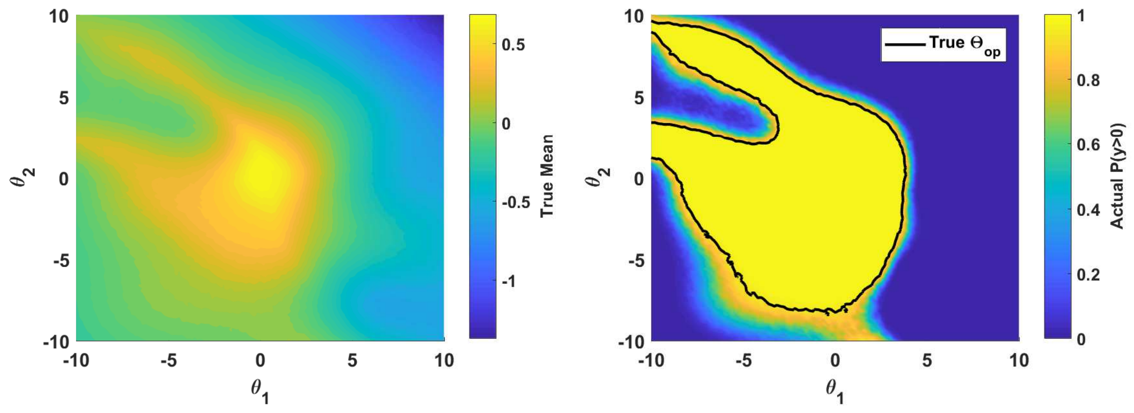
Limitations of Earlier Methods

The limitations of the previous active sampling algorithms are demonstrated in Figures 7-7 and 7-8. These figures examine the prediction accuracy of $\widehat{\Theta}_{op}$ for batch versions of the CDF variance-based algorithm from Chapter 5 as well as the open-loop, random sampling procedure and the PDF variance-based approach. Additionally, due to the similarity of computing $\widehat{\Theta}_{op}$ with the the binary classification problem from Chapters 3 and 4, the results also display the prediction accuracy of the binary classification entropy approach from Chapter 4 applied to this example. As all four of these procedures assume zero prior information, they cannot use (7.7) to select a starting point and must instead begin with an initial training dataset of 10 randomly-selected experiments. The procedures will then select samples in batch sizes of $M = 5$ and



(a) True $\bar{y}_{sim}(\boldsymbol{\theta})$ (b) True $p_{sim}(\boldsymbol{\theta})$

Figure 7-5: [Example 7.4] True probability of satisfaction function $p_{sat}(\boldsymbol{\theta})$ for the simulation stage. This information is passed as the informative nonzero prior to the experiment-based verification stage



(a) True $\bar{y}_{rw}(\boldsymbol{\theta})$ (b) True $p_{sat}(\boldsymbol{\theta})$ and Θ_{op}

Figure 7-6: [Example 7.4] True probability of satisfaction function $p_{sat}(\boldsymbol{\theta})$ and safe operating region Θ_{op} for the real-world dynamics.

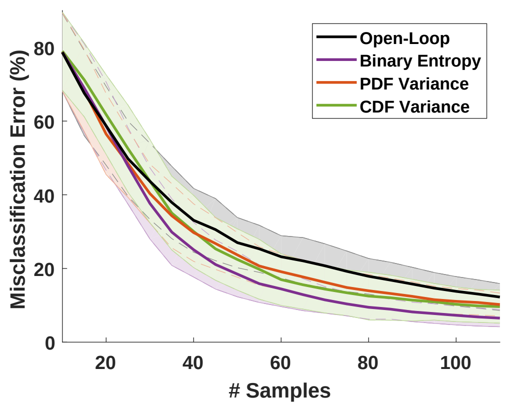
perform 20 iterations for a total training dataset of 110 experiments at the completion of the process.

Figure 7-7(a) displays the prediction error convergence for the four sampling procedures. All of them demonstrate similar prediction accuracy for $\widehat{\Theta}_{op}$, with the binary entropy approach from Chapter 4 slightly outperforming the other algorithms even though it was not originally intended for use with stochastic systems. While the prediction error reduces with each additional sample, the number of failures in Figure 7-7(b) grows steadily. If the maximum number of allowable failures (N_{fail}) is high, then the roughly 50% failure rate is not an issue. However, when N_{fail} is small in comparison to N_{lim}, the procedures will be forced to prematurely terminate. Figure 7-8 illustrates the effect of a small number of allowable failures $N_{fail} = 10$ on the prediction accuracy. As the procedures must terminate once they reach N_{fail} and run out of testing objects, there will be no further improvement of $\widehat{\Theta}_{op}$ and the predictions will fail to converge to Θ_{op}. These plots highlight the limitations of the previous active sampling methods when they are directly applied to a failure-constrained verification problem.

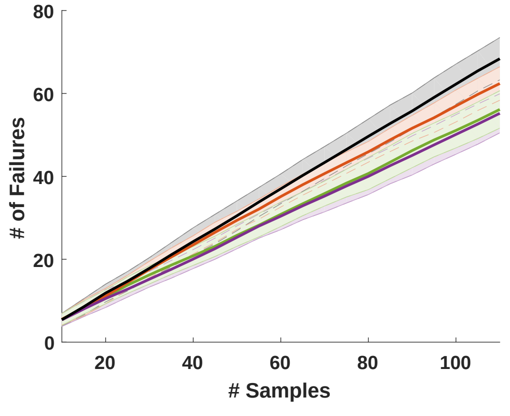
Failure-Adverse Closed-Loop Verification

The failure-adverse closed-loop verification algorithms in Section 7.3 were specifically developed to avoid those types of issues. Figure 7-9 illustrates the evolution of $\widehat{\Theta}_{op}$ produced by Algorithm 12. The process starts with one experiment selected according to (7.7) and computes the predicted probability of satisfaction $\widehat{p}_{sat}(\boldsymbol{\theta})$ and safe operating region $\widehat{\Theta}_{op}$. Figure 7-9(a) also clearly shows the effect of the simulation prior upon the predictions since $\widehat{p}_{sat}(\boldsymbol{\theta})$ has nonuniform predictions in the regions outside the immediate vicinity of the lone training point. For a zero-mean prior, the prediction would be $\widehat{p}_{sat}(\boldsymbol{\theta}) = 0.5$ over most of Θ. Given the initial GP regression model and predictions, Algorithm 12 will select additional experiments to perform. Figures 7-9(b) and 7-9(c) show the outward expansion of $\widehat{\Theta}_{op}$ for the first few iterations. In this problem, the search area is set to the same 95% confidence interval at Θ_{op} with $y_{min} = 0$ which causes the search area $\mathcal{U}_{sa}$ to match $\widehat{\Theta}_{op}$. As the number of experiments increases, the prediction model begins to learn the approximate boundaries of Θ_{op}, as seen in Figure 7-9(d). The training process experiences only 3 failures, all within close proximity to Θ_{op}, over the course of the 51 experiments.

Figure 7-10 compares the performance of a batch ($M = 5$) version of Algorithm 12 against the previous sampling approaches examined in Figures 7-7 and 7-8 over 65 different training datasets and random seeds. If there are an unlimited number of failures allowed, the prediction error convergence of the EMI approach in Figure 7-

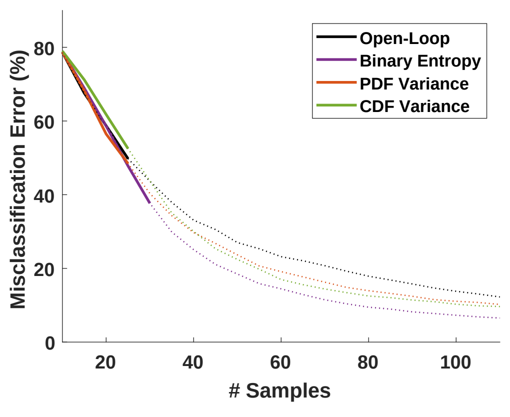


(a) Misclassification error

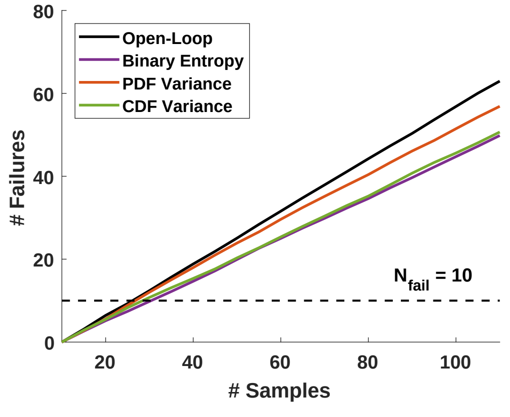


(b) # of failures encountered

Figure 7-7: [Example 7.4] Comparison of prediction error convergence and the number of failures using the previous active sampling approaches. The standard deviations correspond to 1σ bounds.



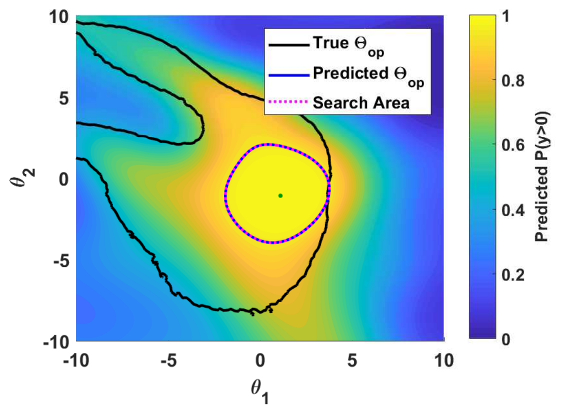
(a) Mean misclassification error



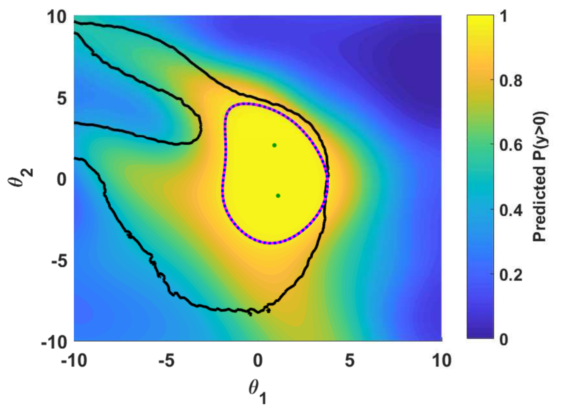
(b) Mean # of failures encountered

Figure 7-8: [Example 7.4] Illustration of the effects of small number of failures ($N_{fail} = 10$) upon the prediction accuracy when using the previous active sampling approaches.

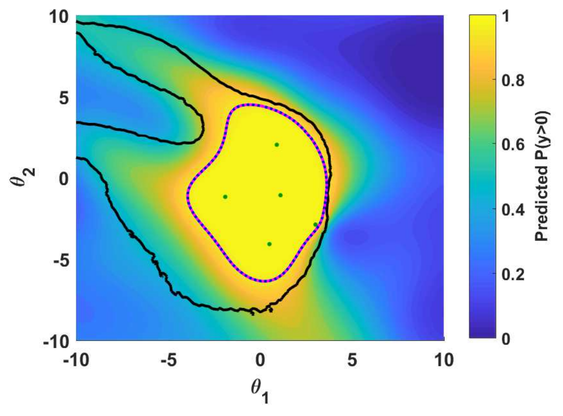
10(a) closely matches the results for the other sampling strategies. However, a large N_{fail} is unlikely and those previous algorithms from Chapters 4 and 5 will terminate prematurely for low N_{fail}, as was shown in Figure 7-8(a). In contrast, Figure 7-10(b) illustrates the EMI procedure achieves the same prediction accuracy without a large number of failures. At the conclusion of 20 iterations of the batch EMI algorithm (for $|\mathcal{L}_{rw}| = 101$), the number of failures is between 0 and 6, with an average of 2. The substantial decrease in the failure rate without a sacrifice in prediction accuracy



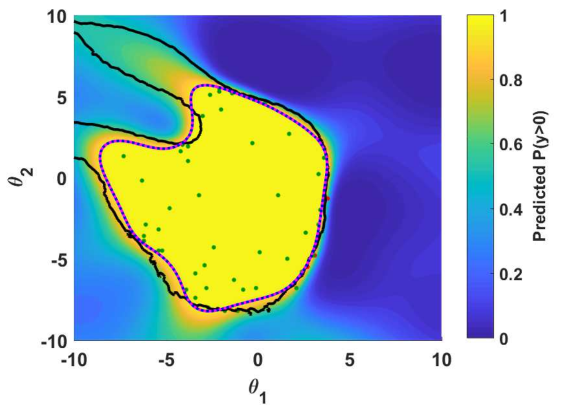
(a) After the initial experiment



(b) After 2 experiments



(c) After 5 experiments

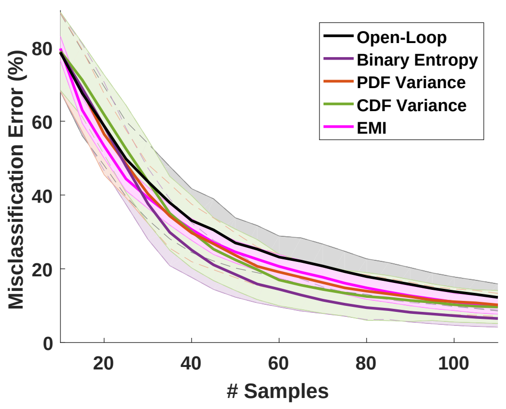


(d) After 51 experiments

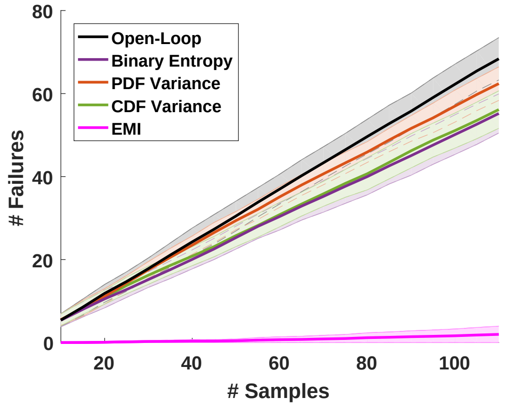
Figure 7-9: [Example 7.4] Evolution of $\widehat{\Theta}_{op}$ and the search area $\mathcal{U}_{sa}$ with each additional experiment. Note that this example uses a static search area with the same parameters as $\widehat{\Theta}_{op}$ and hence $\mathcal{U}_{sa} = \widehat{\Theta}_{op}$.

clearly highlights the advantages of the expected model increase algorithms over the previous active sampling methods.

Although Figure 7-10 demonstrated the advantage of the EMI algorithms over the previous approaches, the higher computational cost of the EMI algorithms may limit their suitability in certain applications. Instead, the non-EMI sampling approaches can be modified to include simulation priors and the restricted search area. Figure 7-11 compares the prediction error convergence of these modified algorithms against the convergence of the EMI algorithm. While the modified algorithms' convergence rates are lower than for the original algorithms in Figure 7-10(a), the failure rates



(a) Misclassification error



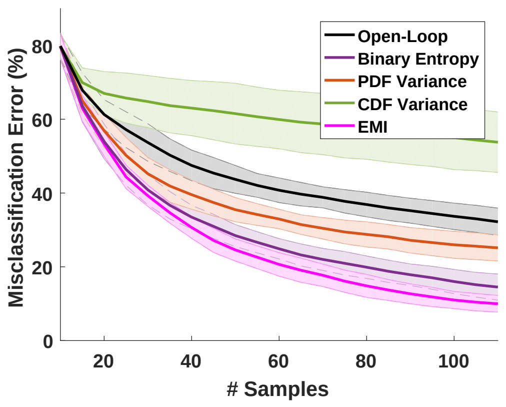
(b) # of failures encountered

Figure 7-10: [Example 7.4] Comparison of a batch version of Algorithm 12 against the previous active sampling approaches. The standard deviation intervals around the means (solid lines) correspond to 1σ bounds.

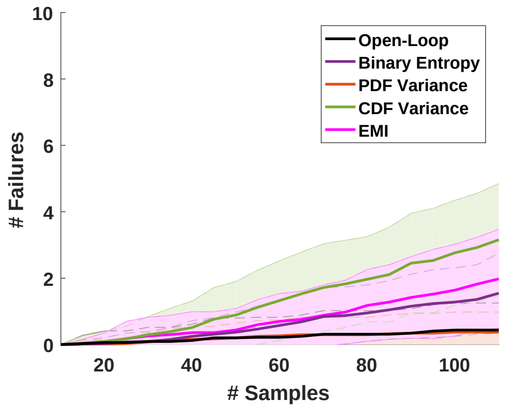
are substantially lower. The performance of the CDF variance-based algorithm from Chapter 5 suffers considerably, but the prediction errors of binary entropy and PDF variance approaches are only marginally worse. When the modifications are applied to sequential versions of the sampling algorithms, the same general trends occur. In fact, the convergence of the sequential PDF variance-based algorithm matches the EMI algorithm in the results from Figure 7-12(a) with the binary entropy-based procedure closely behind. The modified versions of those two approaches offer good alternatives to the EMI algorithms in both sequential and batch scenarios. Regardless of the exact selection metric, failure-adverse closed-loop verification reduced the number of failures by 94-99%.

7.5 Summary

This chapter presented a method for transferring predictions between verification stages and developed a framework for failure-adverse verification during experimental testing. Simulation-based predictions from a preceding verification stage provide informative priors to experiment-based verification. These simulation priors help the closed-loop verification procedures avoid encountering failures during the initial set

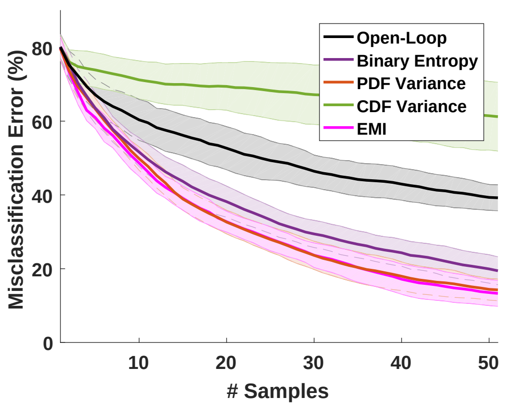


(a) Misclassification error

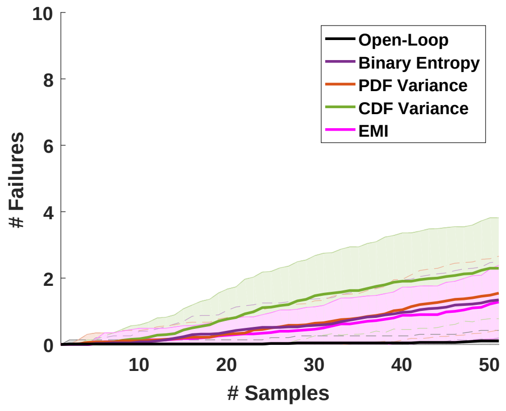


(b) # of failures encountered

Figure 7-11: [Example 7.4] Comparison of a batch version of Algorithm 12 against the previous active sampling approaches modified with a restricted search area. The standard deviation intervals around the means (solid lines) correspond to 1σ bounds.



(a) Misclassification error



(b) # of failures encountered

Figure 7-12: [Example 7.4] Comparison of sequential Algorithm 12 against the previous active sampling approaches modified with a restricted search area. The standard deviation intervals around the mean (solid lines) correspond to 1σ bounds.

of experiments. Additionally, this framework introduced constraints for the set of available sampling to restrict the active sampling process to a set of parameters with a minimum probability of satisfying the requirement. These two modifications form the basis for new active sampling procedures, but can also be applied to any existing active sampling procedure. The new failure-adverse closed-loop verification framework drastically reduced the number of failures by 94-99% with minimal influence on prediction accuracy.

Chapter 8

Conclusions and Future Work

This thesis developed strategies for efficient statistical verification of complex nonlinear systems subject to parametric uncertainties. Statistical verification of uncertain nonlinear systems traditionally relies upon exhaustive simulation-based testing of the closed-loop system under consideration, which limits the speed with which the system can be verified and reduces suitability in resource-constrained applications. The work in this thesis combines data-driven statistical learning techniques with control system verification to maximize the accuracy of predictions while restricted to a computational budget.

Chapter 3 introduced data-driven verification methods and closed-loop statistical verification frameworks for deterministic nonlinear systems. Given a small set of observed trajectories, these data-driven verification methods exploit support vector machines to learn and predict the satisfaction of performance requirements over the entire set of possible parametric uncertainties. In contrast to deductive verification techniques, this statistical verification approach is not beholden to Lyapunov function-based analytical certificates or similar analytical methods that restrict the class of applicable systems. Due to the importance of the set of observed training data upon the accuracy of the predictions, closed-loop statistical verification was developed to improve the informativeness of the training dataset. This framework iteratively selects the next set of simulations or experiments to perform in order to maximize the expected improvement in the predictions. Simulation results with the closed-loop framework demonstrated up to a 50% improvement in prediction accuracy over passive statistical verification approaches.

Chapter 4 expanded upon the closed-loop verification framework to improve its accuracy and suitability within an important subset of the class of problems addressed in Chapter 3. This chapter developed a new verification framework based upon Gaussian process regression models to exploit the availability of non-binary measurements of a trajectory's robustness to the performance requirement. The GP-based prediction model introduced the ability to quantify prediction confidence online without relying upon external validation datasets that typically require valuable simulation or experimental data to be siphoned away from the training dataset. The change from SVMs to GPs and their quantifiable prediction confidence also motivated the redevelopment of the closed-loop verification procedures. New GP-based closed-loop procedures maximize the improvement in prediction confidence and their advantage over competing approaches was demonstrated on numerous simulation examples of increasing complexity.

Chapters 5 and 6 adapted the procedures from Chapters 3 and 4 to address stochastic systems. The presence of stochasticity in the dynamics produces distributions of trajectory robustness measurements that challenge the earlier verification methods which assume single, deterministic measurements. Chapters 5 and 6 introduced stochastic verification frameworks that implement different GP-based prediction models tailored to the various possible distributions. Regardless of whether the measurements are binary or not, the closed-loop verification procedures rely upon a new selection metric that reduces the variance of the cumulative distribution in order to improve the accuracy of the predictions. Results for stochastic versions of the previous simulation examples demonstrated the consistent effectiveness of the stochastic closed-loop verification frameworks for various distributions.

Finally, Chapter 7 presented data-driven statistical verification techniques for multi-stage verification processes which perform verification on different models of the system with increasing fidelity. In particular, the chapter formulated an approach to transfer predictions derived from simulators into real-world domains. This forward transfer of information is a key component of an extension of closed-loop verification called failure-adverse closed-loop verification that adds constraints to the sample selection process in order to avoid failures during experiments. Failure-adverse closed-loop statistical verification significantly reduces the number of failures over the original closed-loop approaches with minimal impact upon prediction accuracy. This was shown on a redevelopment of earlier examples. In short, the last chapter ties all the preceding deterministic and stochastic frameworks together as part of a higher-level process. This work highlighted the ability of data-driven statistical verification to improve the efficiency of control system verification all the way from low-fidelity deterministic verification during preliminary control system design to experimental testing on actual prototypes of the closed-loop system.

8.1 Future Work

The following section describes possible extensions to the work in this thesis. The first two extensions discuss changes to the implementation details of the statistical verification frameworks to improve the utility of the approaches in more challenging applications. The third extension examines the effect of sampling discretization upon prediction accuracy, while the following two extensions address issues and limitations of the GP-based regression techniques that were identified earlier in the thesis. The final extension discusses the use of closed-loop verification alongside recent data-driven optimization techniques for black-box control system design.

High-Dimensional Systems and Sparse Approximations

The work in Chapters 4 and 5 mainly considered the standard deterministic and stochastic Gaussian process regression models that are widely-used across many unrelated disciplines. However, the computational complexity associated with the construction of these models limits their tractability as 1) the size of the training dataset, 2) the dimension of $\boldsymbol{\theta}$, and/or 3) the size of Θ_d become large. The following recent developments in machine learning and statistical inference could help address those three issues in a practical manner.

First, sparse Gaussian process techniques [150] have been developed for regression and classification problems to cap the number of allowable points in the training dataset. These sparse GP methods restrict the GP's training dataset to a fixed number of points in order to maintain a certain level of computation complexity. After each additional datapoint or batch of datapoints are obtained, the training process identifies whether a new datapoint should replace one of the points in the current training dataset or should be ignored. In this manner, sparse GPs are similar in concept to support vector machines since they only actually make predictions based upon a subset of the total set of observed simulation or experimental data. Additionally, sparse GPs have also been successfully employed in Bayesian nonparametric adaptive control [88] and reinforcement learning [158]. While they would help reduce the computational overhead associated with large datasets, sparse GPs do introduce new challenges with hyperparameter optimization and numerical stability, which is why they were not used in this thesis.

Second, recent developments in high-dimensional Gaussian process models [123] would help improve the suitability of the statistical verification frameworks in systems where the value of p in the parametric uncertainties $\boldsymbol{\theta} \in \mathbb{R}^p$ is large. This new derivation of the standard GP models finds a sparse approximation of the full GP model during a more complex training procedure. Similarly, another set of recent developments [124] decomposes the full sampling set Θ_d into smaller subsets and trains one Gaussian process model for each of those subsets. In this manner, one "full" GP trained on the entire set Θ_d is replaced by an array of GPs trained on less computationally-demanding subsets. This method could be used to address both high-dimensional $\boldsymbol{\theta}$ and large Θ_d.

A third possible modification is an adaptive resolution for Θ_d. While this does not necessarily require any changes to the current implementation of the Gaussian process model, it would introduce an additional method to increase the discretization of Θ_d in pertinent regions. Due to the similar concept, box thresholding techniques [72] could be redeveloped as a possible solution. Ultimately, all these possible extensions still exploit the same concepts and frameworks developed in the thesis, but would introduce more complex statistical inference techniques to address difficult subsets of the class of relevant problems.

Improved Stochastic Modeling Techniques

Another related potential research direction is the implementation of different stochastic inference techniques. Section 5.5 already briefly introduced recent techniques for modeling non-Gaussian likelihoods [145–149]. The majority of these were for t-process regression methods [145–147] which replace Gaussian distributions with Student's t-distributions in order to reduce the sensitivity of the regression model to outliers. While these approaches are still being actively developed and refined, they could improve the robustness and numerical stability of the prediction model with respect to stochastic measurements. Meta-GP [149] is another new method for modeling non-Gaussian likelihoods; however, this approach can handle a wider class of distributions than t-processes, such as multi-modal distributions. Although they are capable of handling non-Gaussian likelihoods, all of these methods assume the likelihood model does not vary with $\boldsymbol{\theta}$. The single greatest improvement for the stochastic verification framework would be the development of a modeling technique that is capable of handling spatially-varying non-Gaussian distributions without prior assumptions.

Impact of Sampling Grid Resolution upon Prediction Accuracy

An interesting research direction for both the SVM- and GP-based methods is the exploration of the impact of the resolution of sample set Θ_d upon the accuracy of the predictions. Since samples are chosen from Θ_d, the resolution of this set directly controls the ability of the prediction model to reproduce arbitrary shapes for the $\Theta_{sat}/\Theta_{fail}$ boundary or surface $p_{sat}(\boldsymbol{\theta})$. If Θ_d is a rather coarse discretization of Θ, then even if measurements are taken at every single location in Θ_d, the prediction model will not be capable of accurately reproducing surfaces with high curvature. Earlier work in radial basis function (RBF) neural networks, particularly those focused on adaptive control, demonstrated that the spacing of points in a sampling lattice will restrict the ability of the RBF neural network to reproduce a smooth function [159–162]. SVMs and GPs are closely related to RBF neural networks and thus the same conclusions apply with minor modification. Closed-loop verification addresses an even more complex problem than this previous work because the active selection of training points typically results in an irregular distribution of points across Θ_d.

The earlier work in RBF neural networks could be extended to closed-loop verification problems to compute guarantees for the minimum accuracy of the predictions. Those earlier methods [159, 160, 162] use Fourier transforms and bandlimited functions to determine the approximation ability of a RBF network with a regularly-spaced sampling grid, similar to the Nyquist-Shannon sampling theorem for digital signal processing. In the simplest case where simulations or experiments are performed at every location in regularly-spaced sample set Θ_d, these same approaches could be used to determine the ability of a SVM or GP to reproduce a surface with a certain curvature. These techniques could also be used in reverse order to determine an appropriate discretization for Θ_d. Such an analysis would also enhance prediction confidence and guarantee that the current prediction model is able to reproduce a certain set of possible surfaces. As the resolution increases, the guaranteed set of surfaces that the prediction model can reproduce is expected to increase. Later extensions would explore more relevant problems with irregular grids of points, as would be expected after closed-loop verification. These extensions would be able to produce local guarantees of prediction accuracy given the neighboring points in the irregular distribution.

Systems with Multiple Requirements

As discussed in Sections 4.1 and 4.2, the Gaussian process regression techniques presented in Chapters 4 and 5 are only able to model a single requirement at a time. Each Gaussian process corresponds to a single requirement and multiple, parallel GPs are necessary to model the simultaneous satisfaction of multiple requirements. By itself, this does not present a significant obstacle to modeling the satisfaction of the requirements, and only necessitates a higher computational cost to train the multiple

GPs. However, the need for multiple GPs does challenge the closed-loop verification procedures presented in the earlier chapters.

The main issue faced by closed-loop verification applied to systems with multiple non-binary requirements is the likely disagreement over the informativeness of particular operating conditions. Especially in systems with competing requirements, it is not hard to imagine each of the parallel GPs will rank the expected informativeness of a simulation at one parameter setting differently. Each parallel GP will have its own highest-ranked parameter setting or set of conditions for the upcoming simulations. The problem is how to balance the competing suggestions and choose a single simulation/experiment or set of simulations/experiments.

One possible direction would exploit recent developments in multi-task active learning [117] to develop new closed-loop verification procedures. Multi-task active learning techniques combine the selection metrics of different tasks together in a scalar function to rank prospective sample locations with a single criterion. These functions could be as simple as the maximum of the parallel scores or a more complex nonlinear combination of them. Similar ideas from multi-objective optimization [163] could also be applied. Ideally, the new closed-loop verification procedures would combine aspects from all of these existing techniques to address the specific challenges associated with verification.

Multi-Stage Verification with Forward Transfer of Predictive Covariance

Section 7.1 identified the primary limitation of the forward transfer method used in this thesis: the absence of the prior verification stage's predictive covariance $\Sigma_1(\boldsymbol{\theta})$ in the later stage's predictions, $\mu_2(\boldsymbol{\theta})$ and $\Sigma_2(\boldsymbol{\theta})$. Without this information, the second stage has no way of knowing whether the first stage had high confidence in its predictions when it incorporates those results into its own prediction model. Recent work in transfer learning for Gaussian process models [164] has the potential to completely fix that limitation. This new forward transfer method not only includes the ability to transfer the covariance $\Sigma_1(\boldsymbol{\theta})$, but also presents a novel method to use Stage 1's predictions to improve the hyperparameter optimization process in Stage 2.

The latter aspect would offer significant assistance when the second stage is forced to rely upon an extremely small training dataset $\mathcal{L}_2$. Additionally, an improved forward transfer method would enable new directions for the research in multi-stage verification such as more complex procedures that seamlessly switch back and forth between models of different fidelity.

Black-Box Robust Controller Design and Optimization

A final possible research extension of this work would be the implementation of closed-loop verification within a black-box controller design and optimization process. Recent Bayesian optimization techniques applied to controller design and optimization [96,165] have shown potential for black-box control system design or parameter tuning, but have restricted the measurements to quantifiable metrics obtainable via a single simulation or experiment. One interesting extension would be the combination of these Bayesian optimization techniques with closed-loop statistical verification frameworks to optimize the robustness of a control system. Unlike the existing controller optimization techniques [96,165], this research direction would use closed-loop verification to predict the robustness of a set of candidate controller parameters. The Bayesian optimization technique would then utilize these robustness predictions to adjust the controller parameters and either maximize the robustness of the closed-loop system or maximize another value while meeting a minimum level of robustness.

Appendix A

Concurrent Learning Model Reference Adaptive Control

This appendix describes two concurrent learning model reference adaptive control (CL-MRAC) examples used throughout this thesis. The first example considers a two-state linear system with two sources of parametric uncertainty. The second example expands upon this system and includes two additional sources of uncertainty as well as a more complex formulation with control saturation.

Simple CL-MRAC System

For the first example, consider an uncertain, second order linear system

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -0.2+\theta_1 & -0.2+\theta_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t) \tag{A.1}$$

with two uncertain parameters $\boldsymbol{\theta} = [\theta_1, \theta_2]^T$ that are not known in advance. The system is expected to track a desired reference trajectory produced by the following linear system,

$$\begin{bmatrix} \dot{x}_{m_1} \\ \dot{x}_{m_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -2\zeta_n\omega_n \end{bmatrix} \begin{bmatrix} x_{m_1} \\ x_{m_2} \end{bmatrix} + \begin{bmatrix} 0 \\ \omega_n^2 \end{bmatrix} r_{cmd}(t), \tag{A.2}$$

with $\zeta_n = 0.5$ and $\omega_n = 1$. This reference system is excited by step commands $r_{cmd} = 1$ between $t = 0$ and $t = 2$ seconds, $r_{cmd} = 1.5$ between $t = 10$ and $t = 12$ seconds, and $r_{cmd} = -1.5$ between $t = 20$ and $t = 22$ seconds, where $r_{cmd} = 0$ at all other times. The trajectory length for the simulations is set to $T_f = 40$ seconds.

The closed-loop control policy for the system is a function of the actual state $\mathbf{x}(t)$, reference state $\mathbf{x}_m(t)$, reference command $r_{cmd}(t)$, and estimated parameters $\widehat{\boldsymbol{\theta}}(t)$. The scalar control input $u(t)$ consists of three components: reference input $u_{rm}(t)$, feedback input $u_{pd}(t)$, and adaptive input $u_{ad}(t)$,

$$u(t) = u_{rm}(t) + u_{pd}(t) - u_{ad}(t). \tag{A.3}$$

The reference and feedback inputs are constructed so that in the absence of uncertainties ($\boldsymbol{\theta} = [0, 0]^T$), these two inputs are sufficient for ensuring stable, closed-loop tracking of the reference trajectory. The reference and feedback inputs are given by

$$u_{rm}(t) = -\omega_n^2 x_{m_1} - 2\zeta_n\omega_n x_{m_2} + \omega_n^2 r_{cmd}(t) \tag{A.4}$$

$$u_{pd}(t) = K_p e_1(t) + K_d e_2(t), \tag{A.5}$$

where $K_p = 1.5$, $K_d = 1.3$, and $\mathbf{e}(t) = \mathbf{x}_m(t) - \mathbf{x}(t)$ is the tracking error between the reference and actual states. In the presence of uncertainties, the adaptive input $u_{ad}(t)$ helps ensure closed-loop tracking,

$$u_{ad}(t) = \widehat{\theta}_1(t)x_1(t) + \widehat{\theta}_2(t)x_2(t), \tag{A.6}$$

where $\widehat{\theta}_1(t)$ and $\widehat{\theta}_2(t)$ are the estimated parameters updated online according to the concurrent learning adaptive law [121].

Unlike the standard MRAC adaptive law, the CL-MRAC adaptive law updates $\widehat{\boldsymbol{\theta}}$ as a combination of instantaneous and recorded data,

$$\dot{\widehat{\boldsymbol{\theta}}}(t) = -\Gamma\mathbf{x}(t)\mathbf{e}(t)PB - \Gamma_c \sum_{k=1}^{p_{max}} \mathbf{x}_k\mathbf{x}_k^T\widetilde{\boldsymbol{\theta}}^T, \tag{A.7}$$

with parameter estimation error $\widetilde{\boldsymbol{\theta}}(t) = \widehat{\boldsymbol{\theta}}(t) - \boldsymbol{\theta}$. In this example, the adaptive gains are set to $\Gamma = 2$ and $\Gamma_c = 0.2$. Vector B is the control input matrix from (A.1). The symmetric positive-definite matrix P is determined by the Lyapunov equation $A^T P + PA = -I$, where A is the nominal open-loop plant from (A.1) (with $\boldsymbol{\theta} = [0,0]^T$).

The crux of the CL-MRAC adaptive law is the time history stack found in (A.7). The CL-MRAC law actively selects specific datapoints $\mathbf{x}_k$ from the observed portion of the trajectory in order to improve the convergence of the tracking error. These datapoints are specifically chosen to guarantee matrix $\sum_{k=1}^{p_{max}} \mathbf{x}_k \mathbf{x}_k^T$ is positive definite. Once the datapoint budget p_{max} is reached (here $p_{max} = 20$), older datapoints are only replaced with a new datapoint if the resulting matrix will have a higher minimum singular value. This process, called singular value maximization [121], guarantees that new datapoints added to the history stack will strictly improve the rate of tracking and parameter estimation error convergence.

Although this CL-MRAC approach guarantees asymptotic convergence of the tracking and parameter estimation errors, the adaptive control law converts the open-loop linear system into a nonlinear closed-loop system. In particular, the history stack greatly complicates analysis of the closed-loop response due to its periodic, but non-uniform, updates of the saved datapoints. The adaptive control inputs can vary significantly, even at the same state vector $\mathbf{x}(t)$, depending upon the current estimate of the parameters $\widehat{\boldsymbol{\theta}}(t)$.

CL-MRAC System with Control Saturation

The second example is a more complex version of the previous CL-MRAC system. This example involves the same open-loop plant from (A.1); however, the control input $u(t)$ is saturated within limits $-u_{max} \leq u(t) \leq u_{max}$, where $u_{max} > 0$. For additional complexity, two more sources of uncertainty are added to the same previous uncertain parameters (θ_1, θ_2): θ_3 captures uncertainty in the initial state $x_1(0)$ and θ_4 models uncertainty in the control saturation limit u_{max}.

Concurrent learning model reference adaptive control (CL-MRAC) is still used, but the presence of control saturation can lead to instabilities in the adaptation if left unaddressed. In order to counter these issues, pseudo-control hedging (PCH) [122] augments the baseline CL-MRAC procedure. Pseudo-control hedging creates a hedge input ν_h that measures the distance between the desired control input before saturation $u_{des}(t)$ and the control input at saturation $\pm u_{max}$. Note that the desired control input before saturation is simply the original formulation for the control input,

$$u_{des}(t) = u_{rm}(t) + u_{pd}(t) - u_{ad}(t), \tag{A.8}$$

while the new (true) control input is given by

$$u(t) = \begin{cases} u_{max} & \text{if } u_{des}(t) > u_{max} \\ u_{des}(t) & \text{otherwise} \\ -u_{max} & \text{if } u_{des}(t) < -u_{max}. \end{cases} \tag{A.9}$$

The pseudo control hedge input ν_h is simply the difference

$$\nu_h = \begin{cases} u_{max} - u_{des}(t) & \text{if } u_{des}(t) > u_{max} \\ 0 & \text{otherwise} \\ -u_{max} - u_{des}(t) & \text{if } u_{des}(t) < -u_{max}. \end{cases} \tag{A.10}$$

The main issue with control saturation is the controller can no longer perfectly track the reference trajectory $\mathbf{x}_m(t)$ when the control input $u(t)$ is saturated. Even once the parameter estimates $(\widehat{\theta}_1, \widehat{\theta}_2)$ converge to their true value, the actual state trajectory may not be able to follow the unconstrained reference trajectory. Instead, the pseudo-control hedge modifies the reference model to prevent the saturation from negatively affecting the tracking error $\mathbf{e}(t) = \mathbf{x}_m(t) - \mathbf{x}(t)$. Specifically for this example, the PCH-modified reference model becomes

$$\begin{bmatrix} \dot{x}_{m_1} \\ \dot{x}_{m_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega_n^2 & -2\zeta_n\omega_n \end{bmatrix} \begin{bmatrix} x_{m_1} \\ x_{m_2} \end{bmatrix} + \begin{bmatrix} 0 \\ \omega_n^2 \end{bmatrix} r_{cmd}(t) - \begin{bmatrix} 0 \\ 1 \end{bmatrix} \nu_h. \tag{A.11}$$

This PCH-modification also changes the Lyapunov function used to verify the closed-loop stability of the system, discussed with more detail in [122]. This modification to the Lyapunov function adds to the difficulty in obtaining analytical barrier certificates or applying other analytical verification methods - see Section 3.2.

Appendix B

Robust Multi-Agent Task Allocation for Aerial Forest Firefighting

The following appendix describes the robust multi-agent task allocation example used in the thesis. In this example, UAV agents are assigned to find, identify, and track the expansion of forest fires in rough terrain while subject to uncertain wind conditions. A more complete description of the aerial forest firefighting problem is found in earlier work [32]. Additional details for the robust task allocation strategy, robust CBBA, can be found in the seminal work [26].

Wildfires pose a severe environmental and monetary risk around the world. Many of these fires start in hard-to-reach remote areas due to dry conditions and lightning strikes or uncontrolled campfires. After the initial blaze, these fires can rapidly grow in size and eventually burn thousands or even upwards of a million acres of land [166]. One of the primary challenges with these remote wildfires, at least during the initial stages, is adequate detection and surveying of the fire for subsequent firefighting efforts [167]. This initial monitoring of the fire is made even more difficult by the existence of fire spotting, where embers from the original fire are carried aloft by winds until they ignite a second fire downwind from the first. Unmanned aerial vehicles, in particular small, backpack-transportable ones, have been proposed as a potential tool for monitoring wildfires during the initial stages [167].

This example combines robust task allocation with a forest fire simulator to develop robust and effective strategies for UAV monitoring of wildfires given uncertainties in the wind conditions. Changes in the wind conditions (wind speed and direction) will drastically affect the expansion of the fire due to changes in the terrain and vegetation. For instance, dry grasslands will typically burn faster than rocky shrubbery and the wind direction pointing towards either type will change the speed of fire growth. This example utilizes a Matlab-based derivative of commercially-available wildfire simulators [168, 169] to model the expansion of wildfires given a set of wind conditions, vegetation map, and initial fire starting location. A robust task allocation optimization problem will then attempt to assign UAVs in the most efficient manner to maximize the expected coverage of the fire from the UAVs while limited to fuel constraints and vehicle dynamics. The verification problem will then examine a candidate task assignment policy to determine at which wind conditions the control policy will fail to maintain a desired level of coverage and at which it will. Ultimately, a higher-level planner will evaluate the verified robustness of different policies and select the one with the greatest expected performance [32].

Robust Planning under Uncertainty

The task assignment policy is constructed through a robust task allocation framework. This example uses the robust consensus-based bundle algorithm (CBBA) [26,132,170] as the task allocation framework. The robust CBBA algorithm produces conflict-free distributed task assignments in polynomial time and has been demonstrated through flight tests with UAVs and other hardware scenarios [26].

The basic multi-agent task allocation problem attempts to maximize mission performance given a team of N_a agents and N_t tasks to complete. This mission performance is defined by a global objective function that captures the costs or rewards associated with the assignment of particular agents to tasks. This work makes the common assumption the tasks can only be completed by one agent at a time, allowing the objective function to be rewritten as the sum of local objective functions for each agent and their assigned task(s). Additionally, the rewards for completing tasks will vary explicitly with time. For instance, an agent may be penalized for not completing certain tasks within a set time window. The resulting global task allocation problem is given by the mixed-integer nonlinear optimization program:

$$\begin{aligned} \max_{\mathbf{x},\boldsymbol{\tau}} \ & \mathbb{E}_{\boldsymbol{\theta}}\Big\{ \sum_{i=1}^{N_a} \sum_{j=1}^{N_t} c_{ij}(\mathbf{x},\boldsymbol{\tau},\boldsymbol{\theta}) x_{ij} \Big\} \\ \text{s.t.} \quad & \mathbf{G}(\mathbf{x},\boldsymbol{\tau},\boldsymbol{\theta}) \leq \mathbf{b} \\ & \mathbf{x} \in \mathcal{X} \quad \boldsymbol{\tau} \in \mathcal{T}. \end{aligned} \tag{B.1}$$

Design vector $\mathbf{x} \in \mathcal{X}$ is the assignment of all agent-task pairings with x_{ij} denoting whether agent i is assigned to task j, i.e. $\mathcal{X} = \{0,1\}^{N_a \times N_t}$. Decision variable $\boldsymbol{\tau} \in \mathcal{T}$ is the execution sequence, where scalar term τ_{ij} denotes the time when agent i will execute the assigned task j or $\tau_{ij} = \emptyset$ if task j is not assigned to agent i. Vector $\boldsymbol{\theta}$ consists of the planning parameters that may influence the objective cost c_{ij}. In this example, the planning parameters correspond to uncertain wind conditions θ_1 (wind speed) and θ_2 (wind direction). The cost function maps the cost or reward obtained by agent i for completing task j to the set of task assignments $\mathbf{x}$, execution sequence $\boldsymbol{\tau}$, and planning parameters $\boldsymbol{\theta}$. Because the wind conditions are not perfectly known and may change before execution of the policy, the robust CBBA algorithm maximizes the *expected* performance with respect to uncertainties $\boldsymbol{\theta}$. In order to capture the effects of vehicle dynamics and other limitations, the nonlinear constraints $\mathbf{G}$ and $\mathbf{b}$ are placed on the optimization problem.

As it is a distributed algorithm, robust CBBA decomposes (B.1) amongst each agent and greedily generates a task assignment to maximize each agent's individual score. As these will neglect the other agents' scores, CBBA introduces a bidding process to eventually arrive at a conflict-free task assignment with a locally optimum solution. The total expected reward J is the cumulative expected reward of each agent. This example does not directly examine the construction of the task assignment, but instead evaluates the robustness of a task assignment policy after it has already been obtained through the bidding process.

The robust CBBA planner attempts to assign 4 UAVs to complete 30 fire detection and monitoring tasks at locations spread across an arbitrary map of varying vegetation

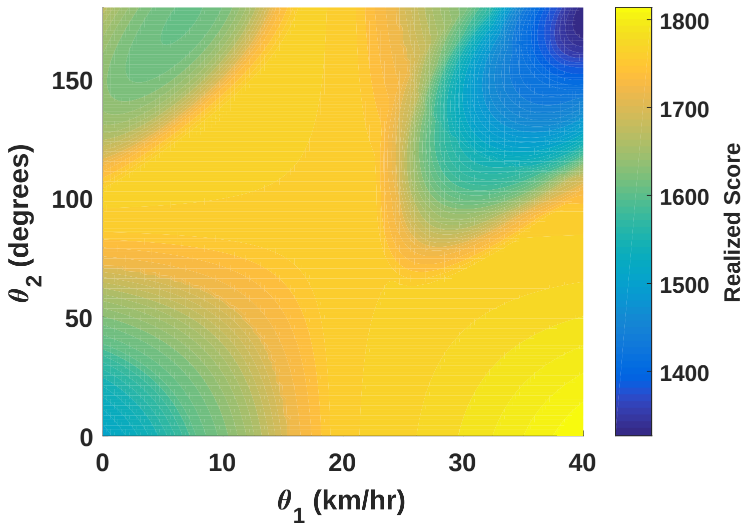


Figure B-1: Illustration of the realized mission score as a function of uncertain wind parameters θ_1 (wind speed) and θ_2 (wind direction). Note, the non-dimensional units of the mission score hold no real-world, physical meaning and simply measure whether the system successfully completed the assigned tasks.

types and terrain. The agents are broken into two types: 2 fast UAVs and another 2 UAVs that are 25% slower. The fire detection tasks correspond to potential fire locations. The UAVs must fly to those locations and check the immediate vicinity for fire hotspots. If one is detected, the UAV will spend more time at that location to map the approximate fire boundary and study its rate of burn before moving on to the next one. As mentioned earlier, different wind conditions will cause the fire to spread in different directions at different rates, meaning some tasks will take longer given one $\boldsymbol{\theta}$ condition, but significantly less for another. Each task also has a certain time window for which the UAV is expected to complete that task. Unforeseen delays may cause a UAV to arrive too late to complete a certain task within its assigned time window, and thus the UAV will miss the "reward" associated with successful completion of that task. Given a task assignment optimized for $\theta_1 = 20$ (km/hr) and $\theta_2 = 90°$ (Easterly wind), the realized score at different wind conditions is plotted in Figure B-1. Note, the example actually considers the full 360° but Figure B-1 only shows $0 - 180°$. The verification problem then attempts to identify whether a set of wind conditions $\boldsymbol{\theta}$ will achieve a realized mission score above 1700. For the stochastic version of the problem, each task duration is multiplied by its own scaling factor k_G. A rough scaling factor $\widehat{k}_G$ is taken from a standard Gaussian distribution $\widehat{k}_G \sim \mathcal{N}(1.1, 1)$ centered at 1.1. In order to avoid negative scaling factors, the actual scaling factor k_G has a minimum of 0.1, i.e. $k_G = \max([0.1, \widehat{k}_G])$.

Appendix C

Lateral-Directional Autopilot Model

The lateral-directional autopilot example considers an autopilot for a de Havilland DHC-2 Beaver airplane. This airplane is a single-engined propeller aircraft with a high-wing design and floats for water landings. The autopilot and autopilot requirements were provided during a session of the USAF's S5 conference [133, 134] while the baseline aircraft model is available in Matlab's Aerospace Blockset and can be accessed with the "asbdhc2" command in the command window.

The example's simulation model replicates the various components of a real-world flight control system, seen in Figure C-1. The airframe model includes full nonlinear 6 degrees-of-freedom aircraft dynamics, complete with actuator dynamics and saturation for each of the flight control surfaces. These dynamics also include nonlinear functions for engine spool-up and other noticeable effects. For the stochastic version of the model, the Dryden wind field model [15] was added to the airframe model to incorporate realistic effects of wind turbulence upon the aircraft's trajectory. The autopilot component of the simulation model contains different controllers for pitch, roll, and yaw which will adjust outputs and gains according to the particular autopilot mode and commands. These autopilot controllers are not modified for the thesis work and verification tests the autopilot's ability to satisfy the given performance requirements.

This example focuses on the "heading-hold" option of the lateral-directional autopilot. The autopilot is expected to turn the aircraft to the desired reference heading

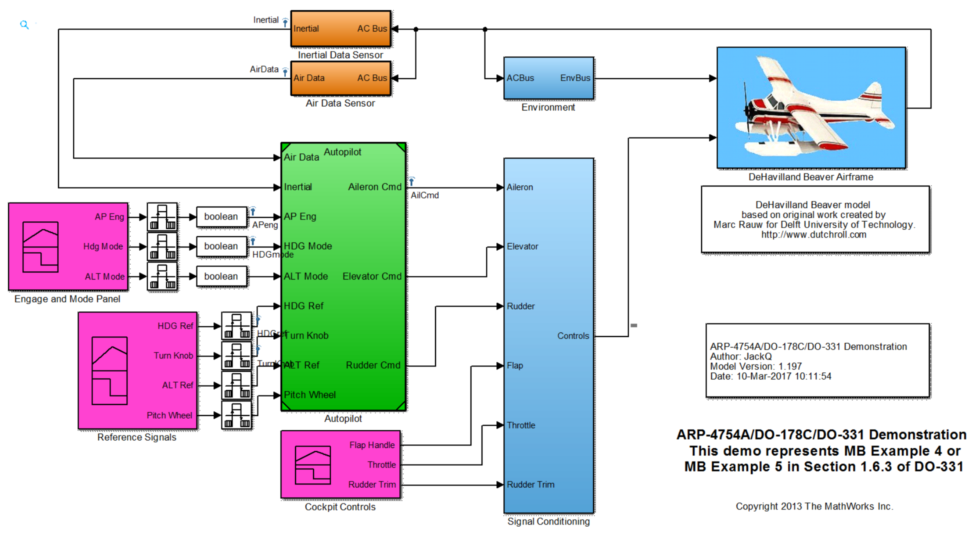
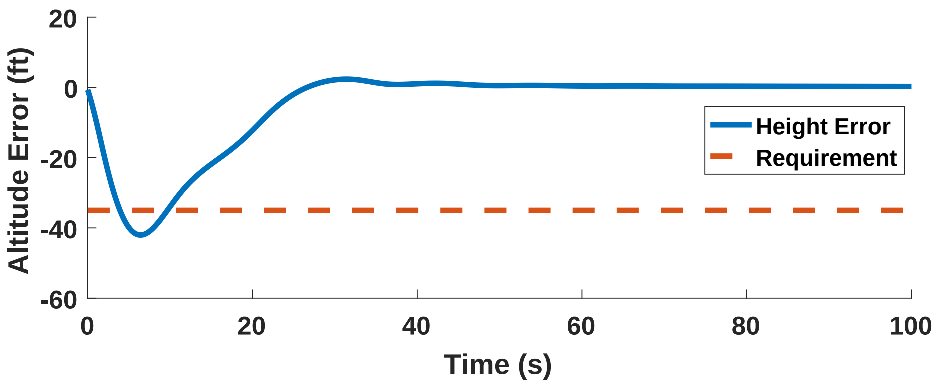

Inertial Data Sensor
Air Data Sensor
Environment
DeHavilland Beaver Airframe
DeHavilland Beaver model based on original work created by Marc Rauw for Delft University of Technology. http://www.dutchroll.com
Autopilot
Engage and Mode Panel
Reference Signals
Cockpit Controls
Signal Conditioning
ARP-4754A/DO-178C/DO-331 Demonstration Author: JackQ Model Version: 1.197 Date: 10-Mar-2017 10:11:54
ARP-4754A/DO-178C/DO-331 Demonstration This demo represents MB Example 4 or MB Example 5 in Section 1.6.3 of DO-331
Copyright 2013 The MathWorks Inc.

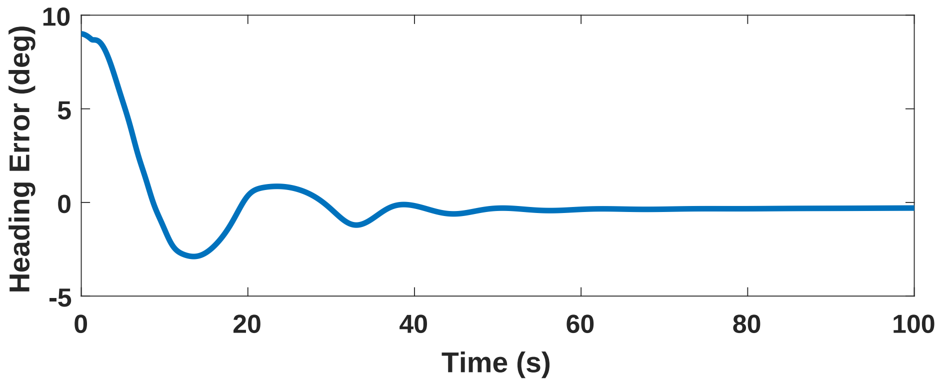
Figure C-1: Components of the lateral-directional autopilot and flight control system for the de Havilland Beaver flight simulation model.

angle and maintain that heading. The desired heading angle is fixed at 112°, which is the default value for the aircraft simulation model. The autopilot is expected to meet various performance requirements as it controls the aircraft during the turn maneuver. The first requirement states the aircraft should achieve a steady-state error of 1° in calm air (no turbulence). The second requirement limits the allowable overshoot and states the aircraft's heading angle should not exceed 10% overshoot in calm air. These two requirements constitute acceptable heading angle performance for the autopilot. During an exhaustive search of different initial conditions and possible parameter settings, the autopilot always met these two heading angle requirements.

Although the autopilot consistently satisfies the heading angle tracking and overshoot requirements, the autopilot is not always able to satisfy a third requirement that places an altitude restriction on the aircraft trajectory during the turn maneuver. This altitude requirement states the aircraft must remain within 35 feet of the initial altitude when the heading-hold command was given. As the aircraft turns from the initial heading angle towards the desired reference heading, the aircraft may gain or lose altitude based upon the effects of the control surface deflections. An example of the heading angle and altitude tracking errors during an arbitrary turn maneuver is shown in Figure C-2. In this trajectory, the autopilot successfully ensures the aircraft



(a) Altitude tracking error



(b) Heading angle tracking error

Figure C-2: Satisfaction of the heading autopilot's requirements over an example trajectory.

satisfies the two requirements for heading angle performance, but is not able to satisfy the altitude restriction. Changes in the initial conditions will affect the satisfaction of the altitude requirement, but only have minimal impact upon the satisfaction of the other heading angle requirements (tracking and overshoot). For this reason, the example problems in this thesis will only examine the satisfaction of the altitude requirement.

Lastly, the autopilot may be engaged at a wide variety of different aircraft orientations and states. Assuming a pilot will only engage the autopilot during cruise conditions, the most relevant variations in orientation and state are in the roll, pitch, and heading angles. These angles represent the aircraft's angular position with respect to the horizon and magnetic North. Additionally, the aircraft may be loaded differently between flights as passengers and cargo are added or removed. The aircraft's weight itself was found to have minimal impact upon the satisfaction of the requirements during the exhaustive search, but changes in moments of inertia exhibited stronger influence. In particular, the longitudinal moment of inertia (I_{yy}) had a non-negligible influence upon the satisfaction of the altitude requirement. These four variables (roll, pitch, heading, and I_{yy}) are the parametric uncertainties examined during verification.

Appendix D

Determinantal Point Processes for Sampling

Determinantal Point Processes (DPPs) are useful tools for selecting sets of samples where diversity in the samples is important [130]. In these approaches, a set of samples generated according to the underlying probability distribution is used to construct a DPP, which then can be used to produce a second sample set of the same size with a higher level of diversity. In many applications, only a small number of samples is desired; however, the DPP loses its utility when it is constructed from a small number of initial samples. For these problems, k-DPPs [131] were developed to produce a small set of samples (of size k) from a DPP constructed with a significantly larger initial set of samples. The overall process is deeply rooted in random matrix theory and an interested reader should examine the seminal work [130, 131] for the full discussion and details. The following algorithm describes k-DPP sampling as it relates to the data-driven verification procedures.

The k-DPP sampling approach in Algorithm 14 assumes M_T samples of $\boldsymbol{\theta}$ have been generated according to probability distributions $\mathbb{P}_E(\boldsymbol{\theta})$ or $\mathbb{P}_V(\boldsymbol{\theta})$ determined by the respective binary classification entropy or CDF variance selection metrics. In order to have a suitable number of samples to construct the DPP, $M_T = 1000$ for the examples in this thesis. These samples form a matrix L that measures correlation between samples (Step 4). An isotropic squared exponential kernel is used to measure similarity and ensure the components of L are bounded $(L(i, j) \leq 1)$ and L is positive definite. The term l is the lone hyperparameter of the RBF kernel. This term was set to $l = 5$ for the examples. Next, the eigenvalues λ_j and eigenvectors v_j of L are found (Step 7). The eigenvalues are also used to compute the corresponding elementary symmetric polynomials e_m. These elementary polynomials and the eigenvalues weight the sample locations and are used to randomly select indices from the M_T samples in Step 10, adding the selected index to set J. Once the loop has selected M indices, a sample y_i from the set J is randomly chosen and added to Y in Steps 21 and 22. The eigenvector corresponding to y_i is then removed from the set V of all remaining eigenvectors. Steps 21-23 are repeated until M samples have been chosen, completing the batch. Note that the values in the output set Y correspond to *indices* of $\boldsymbol{\theta}$ terms in the initial input set of M_T locations sampled from $\mathbb{P}_E(\boldsymbol{\theta})$ or $\mathbb{P}_V(\boldsymbol{\theta})$. The actual sample locations are taken from the original set of M_T points.

Algorithm 14 k-DPP sampling algorithm; adapted from [131].

```
1: Input: $M_T$ randomly generated samples of $\boldsymbol{\theta}$, empty set $J$, batch size $M$
2: for $i = 1, 2, \dots, M_T$ do
3:     for $j = 1, 2, \dots, M_T$ do
4:         Compute $L(i, j) = e^{-||\boldsymbol{\theta}_i - \boldsymbol{\theta}_j||^2/l^2}$
5:     end for
6: end for
7: Eigendecomposition of $L \to \{v_j, \lambda_j\}$
8: Initialize $m = M$
9: for $j = M_T, M_T - 1, \dots, 1$ do
10:    if $u \sim \text{Uniform}[0, 1] < \lambda_j \frac{e^{j-1}_{m-1}}{e^j_m}$ then
11:        $J \leftarrow J \cup \{j\}$
12:        $m \leftarrow m - 1$
13:        if $m = 0$ then
14:            break
15:        end if
16:    end if
17: end for
18: $V \leftarrow \{v_j\} j \in J$
19: $Y \leftarrow \emptyset$
20: while $|V| > 0$ do
21:    Select $y_i$ with probability $\mathbb{P}(y_i) = \frac{1}{|V|} \sum_{v \in V} (v^T e_i)^2$
22:    $Y \leftarrow Y \cup y_i$
23:    $V \leftarrow V_\perp$ (orthonormal basis for subspace of $V$ orthogonal to $e_i$)
24: end while
25: Return: sample set $Y$ of size $M$
```

Bibliography

[1] Eugene Lavretsky and Kevin A. Wise. *Robust and Adaptive Control*. Springer, 2013.

[2] Richard Sutton and Andrew Barto. *Reinforcement Learning, an Introduction*. MIT Press, 1998.

[3] J.-J. E. Slotine and W. Li. *Applied Nonlinear Control*. Prentice Hall, 1991.

[4] Damien B. Jourdan, Michael D. Piedmonte, Vlad Gavrilets, David W. Vos, and Jim McCormick. Enhancing uav survivability through damage tolerant control. In *AIAA Guidance, Navigation, and Control Conference*, 2010.

[5] Girish Chowdhary, Eric Johnson, M. Scott Kimbrell, Rajeev Chandramohan, and Anthony Calise. Flight test results of adaptive controllers in presence of severe structural damage. In *AIAA Guidance, Navigation, and Control Conference*, 2010.

[6] Girish Chowdhary and Eric Johnson. Theory and flight-test validation of a concurrent-learning adaptive controller. *Journal of Guidance, Control, and Dynamics*, 34(2):592–607, 2011.

[7] Eric N Johnson and Suresh K. Kannan. Adaptive trajectory control for autonomous helicopters. *Journal of Guidance, Control, and Dynamics*, 28(3):524–538, 2005.

[8] Pieter Abbeel, Adam Coates, and Andrew Y Ng. Autonomous helicopter aerobatics through apprenticeship learning. *The International Journal of Robotics Research*, 29(13):1608–1639, 2010.

[9] Pieter Abbeel, Adam Coates, Morgan Quigley, and Andrew Y. Ng. An application of reinforcement learning to aerobatic helicopter flight. In *Conference on Advances in Neural Information Processing Systems*, page 2007. MIT Press, 2007.

[10] Mark Cutler and Jonathan P. How. Autonomous drifting using simulation-aided reinforcement learning. In *IEEE International Conference on Robotics and Automation*, 2016.

[11] Christopher Petersen, Morgan Baldwin, and Ilya Kolmanovsky. Model predictive control guidance with extended command governor inner-loop flight control for hypersonic vehicles. In *AIAA Guidance, Navigation, and Control Conference*, Boston, MA, August 2013. American Institute of Aeronautics and Astronautics.

[12] Zheng Qu, Anuradha Annaswamy, and Eugene Lavretsky. An adaptive controller for very flexible aircraft. In *AIAA Guidance, Navigation, and Control Conference*, 2013.

[13] Sohrab Haghighat, Hugh H. T. Liu, and Joaquim R. R. A. Martins. Model-predictive gust load alleviation controller for a highly flexible aircraft. *Journal of Guidance, Control, and Dynamics*, 35(6):1751–1766, 2012.

[14] Technology horizons: A vision for air force science and technology during 2010-2030. Technical report, Office of the US Air Force Chief Scientist, 2011. AF/ST-TR-10-01-PR.

[15] Department of Defense MIL-HDBK-1797. Flying qualities of piloted aircraft.

[16] Federal Aviation Administration FAR Part 25. Airworthiness standards: Transport category airplanes.

[17] Jan Roskam. *Airplane Flight Dynamics and Automatic Flight Controls*. DARcorporation, 1995.

[18] Federal Aviation Administration CFR Part 60. Flight simulation training device initial and continuing qualification and use.

[19] A. Da Ronch, K. J. Badcock, Y. Wang, A. Wynn, and R. Palacios. Nonlinear model reduction for flexible aircraft control design. In *AIAA Atmospheric Flight Mechanics Conference*. AIAA Papaer 2012-4404, American Institute of Aeronautics and Astronautics, 2012.

[20] Xiaohong Li and Ramesh Agarwal. Application of reduced-order models to robust control of the dynamics of a flexible aircraft. In *AIAA Guidance, Navigation, and Control Conference*. AIAA Paper 2003-5504, American Institute of Aeronautics and Astronautics, 2003.

[21] A. Schirrer, M. Kozek, F. Demourant, and G. Ferreres. *Feedback Control Designs*. Springer, 2015.

[22] John Del Frate. Helios prototype vehicle mishap: Technical findings, recommendations, and lessons learned. Technical report, NASA, 2007.

[23] Thomas E. Noll, Stephen D. Ishmael, Bart Henwood, Marla E. Perez-Davis, Geary C. Tiffany, John Madura, Matthew Gaier, John M. Brown, and Ted Wierzbanowski. Technical findings, lesson learned, and recommendations resulting from the helios prototype vehicle mishap. Technical report, NASA, 2007.

[24] Luca F. Bertuccelli. *Robust Decision-Making with Model Uncertainty in Aerospace Systems.* PhD thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, Cambridge MA, September 2008.

[25] Alborz Geramifard. *Practical Reinforcement Learning Using Representation Learning and Safe Exploration for Large Scale Markov Decision Processes.* PhD thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, February 2012.

[26] Sameera S. Ponda. *Robust Distributed Planning Strategies for Autonomous Multi-Agent Teams.* PhD thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, September 2012.

[27] Eric N. Johnson and Sebastien Fontaine. Use of flight simulation to complement flight testing of low-cost uavs. In *AIAA Modeling and Simulation Technologies Conference*, 2001.

[28] Mark J. Cutler. *Reinforcement learning for robots through efficient simulator sampling.* PhD thesis, Massachusetts Institute of Technology, 2015.

[29] Anirudha Majumdar, Amir Ali Ahmadi, and Russ Tedrake. Control and verification of high-dimensional systems with dsos and sdsos programming. In *IEEE Conference on Decision and Control*, 2014.

[30] Edmund M. Clarke and Paolo Zuliani. Statistical model checking for cyber-physical systems. In *International Symposium for Automated Technology for Verification and Analysis*, 2011.

[31] Paolo Zuliani, Andre 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] John F. Quindlen and Jonathan P. How. Machine Learning for Efficient Sampling-based Algorithms in Robust Multi-Agent Planning under Uncertainty. In *AIAA SciTech Conference*, 2017.

[33] Brandon D. Luders. *Robust Sampling-based Motion Planning for Autonomous Vehicles in Uncertain Environments.* PhD thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, May 2014.

[34] Brandon Luders and Jonathan P. How. Probabilistic feasibility for nonlinear systems with non-Gaussian uncertainty using RRT. In *AIAA Infotech@Aerospace Conference*, St. Louis, MO, March 2011. (AIAA-2011-1589).

[35] Brandon Luders, Sertac Karaman, Emilio Frazzoli, and Jonathan P. How. Bounds on tracking error using closed-loop rapidly-exploring random trees. In *American Control Conference (ACC)*, pages 5406–5412, Baltimore, MD, June/July 2010.

[36] Brandon Luders, Ian Sugel, and Jonathan P. How. Robust trajectory planning for autonomous parafoils under wind uncertainty. In *AIAA Infotech@Aerospace Conference*, Boston, MA, August 2013.

[37] David A. Levin, Yuval Peres, and Elizabeth L. Wilmer. *Markov Chains and Mixing Times*. American Mathematical Society, 2008.

[38] Charles Grinstead and J. Laurie Snell. *Introduction to Probability*. American Mathematical Society, 2009.

[39] Goran Frehse, Colas Le Guernic, Alexandre Donze, Scott Cotton, Rajarshi Ray, Oliver Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. Spaceex: Scalable verification of hybrid systems. In *International Conference on Computered Aided Verification*, 2011.

[40] Sadra Sadraddini and Calin Belta. Formal methods for adaptive control of dynamical systems (preprint), 2017.

[41] Sicun Gao, Soonho Kong, and Edmund Clarke. dreal: An smt solver for nonlinear theories of reals. In *International Conference on Automated Deduction*, 2013.

[42] Soonho Kong, Sicun Gao, Wei Chen, and Edmund Clarke. dreach: Delta-reachability analysis for hybrid systems. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 2015.

[43] Fedor Shmarov and Paolo Zuliani. Probreach: Verified probabilistic delta-reachability for stochastic hybrid systems. In *Hybrid Systems Computation and Control*, 2015.

[44] Girish Chowdhary, Tansel Yucelen, Maximilian Muhlegg, and Eric Johnson. Concurrent learning adaptive control of linear systems with exponentially convergent bounds. *International Journal of Adaptive Control and Signal Processing*, 27(4):280–301, April 2013.

[45] Stephen Prajna. *Optimization-Based Methods for Nonlinear and Hybrid Systems Verification*. PhD thesis, California Institute of Technology, 2005.

[46] Stephen Prajna, Ali Jadbabaie, and George J. Pappas. A framework for worst-case and stochastic safety verification using barrier certificates. *IEEE Transactions on Automatic Control*, 52(8):1415–1428, August 2007.

[47] James Kapinski, Jyotirmoy Deshmukh, Xiaoqing Jin, Hisahiro Ito, and Ken Butts. Simulation-guided approaches for verification of automotive powertrain control systems. In *American Control Conference*, 2015.

[48] Joseph Moore and Russ Tedrake. Control synthesis and verification for a perching UAV using LQR-trees. In *IEEE Conference on Decision and Control*, 2012.

[49] Amir Ali Ahmadi, Anirudha Majumdar, and Russ Tedrake. Complexity of ten decision problems in continuous time dynamical systems. In *American Control Conference*, 2013.

[50] James Kapinski, Jyotirmoy Deshmukh, Sriram Sankaranarayanan, and Nikos Arechiga. Simulation-guided Lyapunov analysis for hybrid dynamical systems. In *Hybrid Systems: Computation and Control*, 2014.

[51] Ufuk Topcu. *Quantitative Local Analysis of Nonlinear Systems.* PhD thesis, University of California, Berkeley, 2008.

[52] Ufuk Topcu, Andrew K. Packard, Peter Seiler, and Gary J. Balas. Robust region-of-attraction estimation. *IEEE Transactions on Automatic Control*, 55(1):137–142, January 2010.

[53] Ufuk Topcu, Andrew Packard, and Peter Seiler. Local stability analysis using ssimulation and sum-of-squares programming. *Automatica*, 44(10):2669–2675, 2008.

[54] Philipp Reist, Pascal V. Preiswerk, and Russ Tedrake. Feedback-motion-planning with simulation-based LQR-trees. *International Journal of Robotics Research*, 35:1393–1416, 2016.

[55] Alexandre Donze. Breach: A toolbox for verification and parameter synthesis of hybrid systems. In *International Conference on Computered Aided Verification*, 2010.

[56] Bardh Hoxha, Hoang Bach, Houssam Abbas, Adel Dokhanchi, Yoshihiro Kobayashi, and Georgios Fainekos. Towards formal specification visualization for testing and monitoring of cyber-physical systems. In *International Workshop on Design and Implementation of Formal Tools and Systems*, 2014.

[57] Hengyi Yang, Bardh Hoxha, and Georgios Fainekos. Querying parametric temporal logic properties on embedded systems. In *International Conference on Testing Software and Systems*, 2012.

[58] Truong Nghiem, Sriram Sankaranarayanan, Georgios Fainekos, Franjo Ivancic, Aarti Gupta, and George J. Pappas. Monte-carlo techniques for falsification of temporal properties of non-linear hybrid systems. In *International Conference on Hybrid Systems: Computation and Control*, 2010.

[59] Reuven Y. Rubinstein and Dirk P. Kroese. *Simulation and the Monte Carlo Method.* Wiley, 2008.

[60] Reuven Y. Rubinstein and Dirk P. Kroese. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning.* Springer, 2004.

[61] Young Joon Kim and Mykel Kochenderfer. Improving aircraft collision risk estimation using the cross-entropy method. In *AIAA Modeling and Simulation Technologies Conference*, 2015.

[62] Houssam Abbas, Bardh Hoxha, Georgios Fainekos, and Koichi Ueda. Robustness-guided temporal logic testing and verification for stochastic cyber-physical systems. In *IEEE International Conference on CYBER Technology in Automation, Control, and Intelligent Systems*, 2014.

[63] Sriram Sankaranarayanan and Georgios Fainekos. Falsification of temporal properties of hybrid systems using the cross-entropy method. In *Hybrid Systems: Computation and Control*, 2012.

[64] Marta Kwiatkowska, Gethin Norman, and David Parker. Prism 4.0: Verification of probabilistic real-time systems. In *International Conference on Computer Aided Verification*, 2011.

[65] Marta Kwiatkowska, Gethin Norman, and David Parker. Advances and challenges of probabilistic model checking. In *Allerton Conference on Communication, Control, and Computing*, 2010.

[66] David Henriques, Joao G. Martins, Paolo Zuliani, Andre Platzer, and Edmund M. Clarke. Statistical model checking for markov decision processes. In *International Conference on Quantitative Evaluation of Systems*, 2012.

[67] Paolo Zuliani, Christel Baier, and Edmund M. Clarke. Rare-event verification for stochastic hybrid systems. In *Hybrid Systems: Computation and Control*, 2012.

[68] P. Zuliani, A. Platzer, and E. M. Clarke. Bayesian statistical model checking with application to simulink/stateflow verification. In *Hybrid Systems: Computation and Control*, 2010.

[69] Radu Grosu and Scott A. Smolka. Monte carlo model checking. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 2005.

[70] Andrew Wendorff, Juan Alonso, and Stefan Bieniawski. Using multiple information sources to construct stochastic databases to quantify uncertainty in certification maneuvers. In *AIAA Structures, Structural Dynamics, and Materials Conference*, 2016.

[71] Yu Wang, Nima Roohi, Matthew West, Mahesh Viswanathan, and Geir E. Dullerud. Statistical verification of dynamical systems using set oriented methods. In *Hybrid Systems: Computation and Control*, 2015.

[72] Thao Dang and Noa Shalev. Test coverage estimation using threshold accepting. In *Automated Technology for Verification and Analysis*, pages 115–128, Sydney, Australia, November 2014.

[73] Douglas C. Montgomery. *Design and Analysis of Experiments.* Wiley and Sons, 8th edition, 2013.

[74] C. Devon Lin and Boxin Tang. *Latin Hypercubes and Space-Filling Designs*, pages 593–625. CRC Press, 2015.

[75] Felix Berkenkamp, Matteo Turchetta, Angela P. Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. In *Neural Information Processing Systems (NIPS)*, 2017.

[76] Felix Berkenkamp, Riccardo Moriconi, Angela P. Schoellig, and Andreas Krause. Safe learning of regions of attraction for uncertain, nonlinear systems with gaussian processes. In *IEEE Conference on Decision and Control*, 2016.

[77] Bernhard Scholkopf. Statistical learning and kernel methods. Technical report, Microsoft Research, 2000.

[78] Bernhard Scholkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, 2002.

[79] Vladimir N. Vapnik. *Statistical Learning Theory.* Wiley-Interscience, 1998.

[80] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Journal of Machine Learning Research*, 46(1):131–159, 2002.

[81] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer, 1st edition, 2007.

[82] Davide Anguita, Alessandro Ghio, Sandro Ridella, and Dario Sterpi. K-fold cross validation for error rate estimate in support vector machines. In *International Conference on Data Mining (DMIN)*, 2009.

[83] Zhe Wang and Xiangyang Xue. Multi-class support vector machine. In Yunqian Ma and Guodong Guo, editors, *Support Vector Machines Applications*, chapter 2, pages 23–48. 2014.

[84] Mohamed Aly. Survey on multiclass classification methods, 2005. Online.

[85] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning.* MIT Press, 2006.

[86] M. Kuss. *Gaussian Process Models for Robust Regression, Classification, and Reinforcement Learning.* PhD thesis, 2006.

[87] D. Barber. *Bayesian Reasoning and Machine Learning.* Cambridge University Press, 2012.

[88] G. Chowdhary, H. A. Kingravi, J. P. How, and P. A. Vela. Bayesian nonparametric adaptive control using gaussian processes. *IEEE Transactions on Neural Networks and Learning Systems*, 26(3):537–550, March 2015.

[89] Robert Grande, Girish Chowdhary, and Jonathan P. How. Nonparametric Adaptive Control using Gaussian Processes with Online Hyperparameter Estimation. In *IEEE Conference on Decision and Control (CDC)*. IEEE, 2013.

[90] Marc Peter Deisenroth. *Efficient reinforcement learning using gaussian processes*, volume 9. KIT Scientific Publishing, 2010.

[91] Marc Peter Deisenroth, Dieter Fox, and Carl Edward Rasmussen. Gaussian processes for data-efficient learning in robotics and control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2), 2015.

[92] Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *International Conference on Machine Learning*, 2010.

[93] Alkis Gotovos, Nathalie Casati, and Gregory Hitz. Active learning for level set estimation. In *International Joint Conference on Artificial Intelligence*, pages 1344–1350, 2013.

[94] Yehong Zhang, Trong Nghia Hoang, Kian Hsiang Low, and Mohan Kankanhalli. Near-optimal active learning of multi-output gaussian processes. In *AAAI Conference on Artificial Intelligence*, 2016.

[95] Gang Chen, Zachary Sabato, and Zhaodan Kong. Active learning based requirement mining for cyber-physical systems. In *IEEE Conference on Decision and Control*, 2016.

[96] Thomas Desautels, Andreas Krause, and Joel W Burdick. Parallelizing exploration-exploitation tradeoffs in gaussian process bandit optimization. *Journal of Machine Learning Research*, 15:4053–4103, 2014.

[97] Zohar Manna and Amir Pnueli. *The Temporal Logic of Reactive and Concurrent Systems*. Springer-Verlag, 1992.

[98] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008.

[99] Eric M. Wolff. *Control of Dynamical Systems with Temporal Logic Specifications*. PhD thesis, California Institute of Technology, 2014.

[100] Tichakorn Wongpiromsarn, Ufuk Topcu, and Richard M. Murray. Receding horizon temporal logic planning. *IEEE Transactions on Automatic Control*, 57(11):2817 – 2830, 2012.

[101] Xu Chu Ding, Stephen L. Smith, Calin Belta, and Daniela Rus. Ltl control in uncertain environments with probabilistic satisfaction guarantees. In *18th World Congress of The International Federation of Automatic Control*, 2011.

[102] Oded Maler and Dejan Nickovic. *Monitoring Temporal Properties of Continuous Signals*, pages 152–166. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[103] Sadra Sadraddini and Calin Belta. Feasibility envelopes for metric temporal logic specifications. In *IEEE Conference on Decision and Control*, 2016.

[104] Sadra Sadraddini and Calin Belta. Robust temporal logic model predictive control. In *Allerton Conference on Communication, Control, and Computing*, 2015.

[105] Dorsa Sadigh and Ashish Kapoor. Safe control under uncertainty with probabilistic signal temporal logic. In *Robotics: Science and Systems Conference*, June 2016.

[106] Alexandre Donze and Oded Maler. Robust satisfaction of temporal logic over real-valued signals. In *International Conferences on Formal Modeling and Analysis of Timed Systems*, 2010.

[107] Alexandre Donze. On signal temporal logic, 2014. Lecture Notes: EECS 294, University of California, Berkeley.

[108] James Kapinski and Jyotirmoy Deshmukh. Discovering forward invariant sets for nonlinear dynamical systems. In *International Conference on Applied Mathematics, Modeling and Computational Science*, 2013.

[109] John F. Quindlen, Ufuk Topcu, Girish Chowdhary, and Jonathan P. How. Region-of-Convergence Estimation for Learning-Based Adaptive Controllers. In *American Control Conference*, 2016.

[110] Michael E. Tipping. Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244, June 2001.

[111] Giuseppe Bombara, Cristian-Ioan Vasile, Francisco Penedo, Hirotoshi Yasuoka, and Calin Belta. A decision tree approach to data classification using signal temporal logic. In *International Conference on Hybrid Systems: Computation and Control*, 2016.

[112] John Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74, 1999.

[113] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International Joint Conference on Artificial Intelligence*, volume 2, pages 1137–1145, 1995.

[114] Alnur Ali, Rich Caruana, and Ashish Kapoor. Active learning with model selection. In *AAAI Conference on Artificatial Intelligence*, 2014.

[115] Alexandar Kozarev, John F. Quindlen, Jonathan P. How, and Ufuk Topcu. Case Studies in Data-Driven Verification of Dynamical Systems. In *Hybrid Systems: Computation and Control*, 2016.

[116] Klaus Brinker. Incorporating diversity in active learning with support vector machines. In *International Conference on Machine Learning*, 2003.

[117] Burr Settles. *Active Learning.* Morgan and Claypool, 2012.

[118] Jan Kremer, Kim Steenstrup Pedersen, and Christian Igel. Active learning with support vector machines. *Data Mining and Knowledge Discovery*, 4(4):313–326, July 2014.

[119] Burr Settles and Mark Craven. An analysis of active learning strategies for sequence labeling tasks. In *Conference on Empirical Methods in Natural Language Processing*, 2008.

[120] Weehong Tan. *Nonlinear Control Analysis and Synthesis using Sum-of-Squares Programming.* PhD thesis, University of California, Berkeley, 2006.

[121] Girish V. Chowdhary. *Concurrent Learning for Convergence in Adaptive Control Without Persistency of Excitation.* PhD thesis, Georgia Institute of Technology, December 2010.

[122] Suresh K. Kannan and Eric N. Johnson. Model reference adaptive control with a constrained linear reference model. In *IEEE Conference on Decision and Control*, pages 48–53. IEEE, 2010.

[123] Kirthevasan Kandasamy, Jeff Schneider, and Barnabas Poczos. High dimensional bayesian optimization and bandits via additive models. In *International Conference on Machine Learning*, 2015.

[124] Trong Nghia Hoang, Quang Minh Hoang, and Kian Hsiang Low. A unifying framework of anytime sparse gaussian process regression models with stochastic variational inference for big data. In *International Conference on Machine Learning*, 2015.

[125] Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9:235–284, 2008.

[126] D. J. Tylavsky and G.R.L. Sohie. Generalization of the matrix inversion lemma. *Proceedings of the IEEE*, 74:1050–1052, 1986.

[127] James Bucklew. *Introduction to Rare Event Simulation.* Springer, 2004.

[128] Pierre L'Ecuyer, Michel Mandjes, and Bruno Tuffin. *Rare Event Simulation using Monte Carlo Methods*, chapter Importance Sampling in Rare Event Simulation, pages 17–38. Wiley, 2009.

[129] C. K. Wong and M. C. Easton. An efficient method for weighted sampling without replacement. *SIAM Journal on Computing*, 1980.

[130] Alex Kulesza and Ben Taskar. Determinantal point processes. *Foundations and Trends in Machine Learning*, 5(2-3):123–286, 2012.

[131] Alex Kulesza and Ben Taskar. k-dpps: Fixed-size determinantal point processes. In *International Conference on Machine Learning*, 2011.

[132] H.-L. Choi, L. Brunet, and J. P. How. Consensus-based decentralized auctions for robust task allocation. *IEEE Transactions on Robotics*, 25(4):912–926, August 2009.

[133] Christopher Elliott, Gregory Tallant, and Peter Stanfill. On example models and challenges ahead for the evaluation of complex cyber-physical systems with state of the art formal methods v&v. In *Air Force Research Laboratory Safe and Secure Systems and Software Symposium (S5) Conference*, Dayton, OH, June 2015.

[134] Christopher Elliott, Gregory Tallant, and Peter Stanfill. An example set of cyber-physical v&v challenges for s5. In *Air Force Research Laboratory Safe and Secure Systems and Software Symposium (S5) Conference*, Dayton, OH, July 2016.

[135] Shayegan Omidshafiei, Ali-Akbar Agha-Mohammadi, Yu Fan Chen, Nazim Kemal Ure, Shih-Yuan Liu, Brett T Lopez, Rajeev Surati, Jonathan P How, and John Vian. Measurable augmented reality for prototyping cyberphysical systems: A robotics platform to aid the hardware prototyping and performance testing of algorithms. volume 36, pages 65–87. IEEE, 2016.

[136] Roberto Galatolo Alberto Calia, Eugenio Denti and Francesco Schettini. Air data compution using neural networks. *Journal of Aircraft*, 45(6):2078–2083, November-December 2008.

[137] Vitezslav Hanzal Tomas Censky Pavel Paces, Karel Draxler and Ondrej Vasko. A combined angle of attack and angle of sideslip smart probe with twin differential sensor modules and doubled output signal. In *IEEE Sensors 2010 Conference*. Institute of Electrical and Electronics Engineers, 2010.

[138] Miguel Lázaro-Gredilla and Michalis Titsias. Variational heteroscedastic gaussian process regression. In *The 28th International Conference on Machine Learning*, Bellevue, Washington, July 2011.

[139] Jagdish K. Patel and Campbell B. Read. *Handbook of the Normal Distribution*. Marcel Dekker, 2nd edition, 1996.

[140] Everette S. Gardner. A simple method of computing prediction intervals for time series forecasts. *Management Science*, 34(4), 1988.

[141] Alfredo H-S. Ang and Wilson H. Tang. *Probability Concepts in Engineering.* Wiley, 2nd edition, 2007.

[142] Peng Kou, Deliang Liang, Lin Gao, and Jianyong Lou. Probabilistic electricity price forecasting with variational heteroscedastic gaussian process and active learning. *Energy Conversion and Management*, 89:298–308, 2015.

[143] Kristian Kersting, Christian Plagemann, Patrick Pfaff, and Wolfram Burgard. Most likely heteroscedastic gaussian process regression. In *International Conference on Machine Learning*, 2007.

[144] Paul W. Goldberg, Christopher K. I. Williams, and Christoper M. Bishop. Regression with input-dependent noise: A gaussian process treatment. In *Conference on Advances in Neural Information Processing Systems*, 1998.

[145] Arno Solin and Simo Sarkka. State space methods for efficient inference in student-t process regression. In *International Conference on Artificial Intelligence and Statistics*, 2015.

[146] Amar Shah, Andrew Gordon Wilson, and Zoubin Ghahramani. Student-t processes as alternatives to gaussian processes. In *International Conference on Artificial Intelligence and Statistics*, 2014.

[147] Pasi Jylanki, Jarno Vanhatalo, and Aki Vehtari. Robust gaussian process regression with a student-t likelihood. *Journal of Machine Learning Research*, 12:3227–3257, 2011.

[148] Rishit Sheth, Yuyang Wang, and Roni Khardon. Sparse variational inference for generalized gaussian process models. In *International Conference on Machine Learning*, 2015.

[149] David Seiferth, Girish Chowdhary, Maximilian Muhlegg, and Florian Holzapfel. Online gaussian process regression with non-gaussian likelihood. In *American Control Conference*, 2017.

[150] Lehel Csato. *Gaussian Processes - Iterative Sparse Approximations.* PhD thesis, Aston University, 2002.

[151] A. Y. Ng and S. Russell. Algorithms for inverse reinforcement learning. In *International Conference on Machine Learning*, 2000.

[152] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning*, Banff, Canada, 2004.

[153] Duy Nguyen-Tuong and Jan Peters. Using model knowledge for learning inverse dynamics. In *IEEE International Conference on Robotics and Automation*, 2010.

[154] Hande Topa, Agnes Jonas, Robert Kofler, Carolin Kosiol, and Antti Honkela. Gaussian process test for high-throughput sequencing time series: Application to experimental evolution. *Bioinformatics*, 31(11):1762–1770, 2015.

[155] Luca Bortolussi, Dimitrios Milios, and Guido Sanguinetti. Smoothed model checking for uncertain continuous time markov chains. *Information and Computation*, 2016.

[156] L. Bortolussi, Dimitrios Milios, and G. Sanguinetti. U-check: Model checking and parameter synthesis under uncertainty. In *Quantitative Evaluation of Systems*, 2015.

[157] Aviation Supplies and Academics (ASA). Aerodynamics: Vg diagram.

[158] Miao Liu, Girish Chowdhary, Bruno Castro da Silva, Shih-Yuan Liu, and Jonathan P. How. Gaussian Processes for Learning and Control: Tutorial with Examples. *IEEE Control Systems*, 2017 (submitted).

[159] Hassan A. Kingravi, Girish Chowdhary, Patricio A. Vela, and Eric N. Johnson. A reproducing kernel hilbert space approach for the online update of radial bases in neuro-adaptive control. *IEEE Transactions on Neural Networks and Learning Systems*, 36(7):1130–1141, July 2012.

[160] Robert M. Sanner and Jean-Jacques E. Slotine. Gaussian networks for direct adaptive control. *IEEE Transactions on Neural Networks*, 3(6):837–863, 1992.

[161] Xie Xu, Wenxing Ye, and Alireza Entezari. Bandlimited reconstruction of multidimensional images from irregular samples. *IEEE Transactions on Image Processing*, 22(10):3950–3960, 2013.

[162] J. Park and I. W. Sandberg. Universal approximation using radial-basis-function networks. *Neural Computation*, 3:246–257, 1991.

[163] Anas Alfaris, Olivier de Weck, and Karen Willcox. Multiobjective optimization, 2010. Lecture Notes: MIT 16.888, Massachusetts Institute of Technology - Open Courseware.

[164] Neeti Wagle and Eric W. Frew. Forward adaptive transfer of gaussian process regression. *Journal of Aerospace Information Systems*, 14(4):214–231, 2017.

[165] Felix Berkenkamp, Angela P Schoellig, and Andreas Krause. Safe conroller optimization for quadrotors with gaussian processes. In *International Conference on Robotics and Automation*, 2016.

[166] United States Department of Agriculture. The rising cost of wildfire operations: Effects on the forest service's non-fire work, August 2015.

[167] Javier Perez-Mato, Victor Arana, and Francisco Cabrera-Almeida. Real-time autonomous wildfire monitoring and georeferencing using rapidly deployable mobile units. *IEEE Aerospace and Electronic Systems Magazine*, 31(2):6–15, 2015.

[168] Den Boychuk, W. John Braun, Reg J. Kulperger, Zinovi L. Krougly, and David A. Stanford. A stochastic forest fire growth model. *Environmental and Ecological Statistics*, 16:133–151, 2009.

[169] C. Tymstra, R. W. Bryce, B. M. Wotton, S. W. Taylor, and O.B. Armitage. *Development and Structure of Prometheus: The Canadian Wildland Fire Growth Simulation Model*. Candian Forest Service, 2010.

[170] Sameera S. Ponda, Joshua Redding, Han-Lim Choi, Jonathan P. How, Matt A. Vavrina, and John Vian. Decentralized planning for complex missions with dynamic communication constraints. In *American Control Conference (ACC)*, Baltimore, MD, July 2010.