Active Sampling for Constrained Simulation-based Verification of Uncertain Nonlinear Systems

John F. Quindlen\textsuperscript{1} \hspace{1cm} Ufuk Topcu\textsuperscript{2} \hspace{1cm} Girish Chowdhary\textsuperscript{3} \hspace{1cm} Jonathan P. How\textsuperscript{4}

\textbf{Abstract}—Increasingly demanding performance requirements for dynamical systems necessitates the adoption of nonlinear and adaptive control techniques. While satisfaction of the requirements motivates the use of nonlinear control design methods in the first place, the nonlinearity of the resulting closed-loop system often complicates verification that the system does indeed satisfy the requirements at all possible conditions. Current techniques mostly rely upon analytical function-based approaches to certify satisfaction of the requirements. However, for many complex nonlinear systems, it is difficult to find a suitable function with minimally-conservative results to provably certify the system’s response, if one can be found at all. This work presents data-driven procedures for simulation-based, statistical verification in place of analytical proofs. The use of simulations also enables the computation of an approximate solution within a set time or computational threshold, which is not possible with analytical techniques. Active sampling algorithms are developed to iteratively select additional training points to maximize the accuracy of the predictions while constrained by a sample budget. In particular, sequential and batch active sampling procedures using a new binary classification entropy-based selection metric are developed. These approaches demonstrate up to 40\% improvement over existing approaches when applied to two case studies in control systems.

I. INTRODUCTION

In high performance systems such as aircraft, the system is expected to satisfy numerous criteria in order for that system to be considered “safe” or “satisfactory”. For example, not only does an aircraft need to be stable, but the aircraft design must also meet handling, maneuvering, and emergency requirements [1], [2]. In such applications, control systems are implemented to help satisfy these criteria under various uncertainties by controlling the closed-loop trajectory response. Verification procedures can then be used to ensure the resulting closed-loop system does in fact satisfy the requirements under all possible conditions.

As the demand for higher performance and/or efficiency grows, more advanced nonlinear control techniques will be relied upon to achieve the increasingly-complex requirements associated with such demands. While methods such as model reference adaptive control (MRAC) [3] and reinforcement learning (RL) [4] have demonstrated large improvements in performance and efficiency, the challenge with complex control techniques is certifying that the closed-loop system can actually meet the requirements for all possible uncertainties. This is due to the nonlinear (and possibly adaptive) nature of the controller, which may result in drastically different trajectories given only slightly different operating conditions. This is exacerbated by any nonlinearities in the open-loop plant itself.

Various verification techniques attempt to address this certification problem. If closed-form difference or differential equations of the closed-loop system are known or can be formed, then it may be possible to construct analytical certificates [5]–[7]. These certificates the basis for proofs that provably verify the closed-loop system satisfies the necessary requirements. While such proofs are extremely useful, they are difficult to implement of many systems of interest. For one, closed-form equations of the system may not be known or feasible. Simulation-guided analytical methods [7]–[9] have been developed to help address this, but are limited to certain dynamics. Even if a proof can be constructed, the results may be overly conservative due to the reliance upon specific analytical functions for construction of the proof [10]. Thus, existing analytical verification techniques are not able to suitably certify the satisfaction of the requirements in many nonlinear systems.

Data-driven methods [11] are an alternative to analytical verification techniques to address their limitations for such systems. In particular, simulations of the closed-loop system can be used to construct statistical certificates of safety. While statistical certificates are less absolute than analytical proofs, they do not suffer from the same limitations as the previous proof-based techniques and apply to a wider range of systems. In fact, data-driven verification techniques can be used on any virtually system with a suitable simulation model of the dynamics. A suitable simulation model could include everything up to the highest-fidelity imitations of reality such as a full-motion flight training simulator.

While statistical certificates produced by data-driven verification are able to produce a solution to a wider range of systems, the accuracy of the certificate is restricted by the quality of the simulation data. If the observed simulation data fails to adequately cover the entire space of possible perturbing uncertainties, then the accuracy of the statistical certificate’s predictions in those unobserved regions will be limited. Conversely, while one solution is to obtain a large, highly-discretized set of simulations spanning the space, simulations can be expensive to obtain. High-fidelity simulation models of the system may be computationally intensive and can form a bottleneck during the verification process. Therefore, it best to carefully select simulations so

\textsuperscript{1}Graduate Student, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology

\textsuperscript{2}Assistant Professor, Department of Aerospace Engineering and Engineering Mechanics, University of Texas at Austin

\textsuperscript{3}Assistant Professor, Departments of Agricultural & Biological Engineering and Aerospace Engineering, University of Illinois Urbana-Champaign

\textsuperscript{4}Richard C. Maclaurin Professor of Aeronautics and Astronautics, Aerospace Controls Laboratory (Director), MIT
as to minimize utilization of the model while maximizing the accuracy of the predictions.

In many applications, verification is only part of a higher-level process such as robust nonlinear control policy design or optimization. In those cases, verification is used to test the suitability of a candidate control policy over the set of given uncertainties. The extra challenge with that is the verification procedure should be able to return a robustness estimate within a reasonable amount of time or effort because the procedure will have to repeated for every candidate control policy. Data-driven methods are particularly suited to this application as they can produce a solution within a set limit by bounding the number of simulations. While this cap will have implications on the accuracy of the robustness estimate, analytical proof-based techniques simply can not guarantee those restrictions will be met. The expected cap on the number of simulations also reinforces the need for careful selection of simulations to maximize accuracy of the predictions.

This work addresses those restrictions on the number of simulations with active learning-based sampling procedures. Active sampling [12] methods are closed-loop procedures that iteratively construct a model and then select new training locations to most improve that model. In particular, signal temporal logic [13]–[15] (STL) is used to measure satisfaction of the performance requirements during the simulation trajectories. These measurements are then used to train Gaussian process [16] (GP) regression models that predict the satisfaction of the requirements at unseen perturbations. Existing active learning selection metrics [12], [17] are not ideally suited to this verification problem. Binary support vector machine (SVM) approaches [17] fail to take advantage of the availability of real-valued STL measurements while variance reduction-based sampling criteria [12], [18] ignore the binary nature of predictions: requirements are either satisfied or they’re not. Instead, new tailor-made selection metrics based on binary classification entropy are introduced. Binary classification entropy captures the binary nature of the procedure and can be readily computed using the GP regression model. This new selection metric can then be incorporated into sequential and batch (multiple samples at once) active sampling procedures to select future training locations to improve the predictions.

The paper is structured as follows. Section II introduces the problem and describes STL methods for measuring satisfaction of the system requirements from simulation trajectories. Section III then discusses how these measurements can be used to construct a Gaussian process prediction model and estimate the confidence in those predictions. Closed-loop active sampling procedures that iteratively select training locations to improve the GP prediction model are shown in Section IV. Given a budget on the number of samples, these procedures will attempt to form the best training set to produce the most accuracy statistical certificate. Finally, Section V demonstrates these processes on two example problems: a model reference adaptive controller and an aircraft autopilot. Both of these case studies have uncertain nonlinear dynamics that are exceedingly difficult to analyze using proof-based techniques. The results will also show the entropy-based active sampling procedures outperform existing approaches for verification problems.

II. PROBLEM FORMULATION

Consider the following nonlinear system

\[ \dot{x}(t) = f(x(t), u(t), \theta) \]  

(1)

where \( x(t) \in \mathbb{R}^n \) is the state vector and \( u(t) \in \mathbb{R}^m \) is the control input vector. In this work, an exact simulation model of \( f(x(t), u(t), \theta) \) is assumed to be known and available for verification. The system is also perturbed by parametric uncertainties \( \theta \in \mathbb{R}^p \) that affect the state dynamics. While the exact values of \( \theta \) at run-time are unknown during the time of verification, their effects in \( f(x(t), u(t), \theta) \) and the set of all feasible perturbations \( \theta \in \Theta \) are assumed to be known. For example, \( \theta \) could include initial conditions \( x(0) \) and mass/inertia properties of a physical system. Lastly, as the emphasis is on verification of the closed-loop system, the control policy \( u(t) = g(x(t)) \) is assumed to be known.

Let the trajectory of the closed-loop system in the time interval \( t \in [0, T_f] \) be given by \( \Phi(x(t), x_0, \theta) \). The trajectory \( \Phi(x(t)\mid x_0, \theta) \) can be completely determined from \( x_0 \) and \( \theta \). Here, \( x_0 \) is a fixed nominal initial condition. If the true initial condition \( x(0) \) is uncertain (\( x(0) \neq x_0 \)), then \( x(0) \) can still be modeled as the combination of the known nominal initial condition \( x_0 \) and the unknown perturbations, i.e. \( x(0) = x_0 + \theta \). The resulting trajectory \( \Phi(x(t)\mid x_0, \theta_1) \) will ultimately determine whether the performance requirements were satisfied at a particular \( \theta_1 \) perturbation.

A. Requirement Modeling

The closed-loop system trajectory is expected to satisfy certain performance criteria, assumed to be given a-priori. These criteria may be given by a wide variety of sources and can include relatively straightforward concepts like stability or boundedness as well as more complex spatio-temporal requirements such as those specified in civil or military aviation regulations [1], [2]. In order to capture the wide range of possible specifications, the performance requirements are modeled using temporal logic specifications. In particular, these specifications are given by signal temporal logic (STL). Signal temporal logic provides a mathematical language for expressing the requirements and measuring their satisfaction in \( \Phi(x(t)\mid x_0, \theta) \). The following subsection will briefly overview STL and its applications to the problem. More formal, comprehensive descriptions of STL are found in [13]–[15].

A requirement of interest is given as a STL formula \( \varphi \). This formula consists of predicates \( \zeta \) that are functions of the state (and possibly control input) values and Boolean and temporal operations on those predicates. The predicate \( \zeta \) defines inequalities for functions of the state, input values such that \( \zeta > 0 \). For example, if state \( x_1(t) \) must remain above 2, then the corresponding predicate is \( x_1(t) > 2 \). Temporal operators include \( \square_{[t_1,t_2]} \), \( \Diamond_{[t_1,t_2]} \), and \( U_{[t_1,t_2]} \).
which express that a predicate must hold “always”, “at some time”, and “until another predicate” between between times $t_1$ and $t_2$. Meanwhile, the Boolean operators $\land$, $\lor$, and $\neg$ can be used alongside the temporal operators to express negation, conjunction, and disjunction. Tuple $(\Phi, t) \models \varphi$ then signifies that trajectory $\Phi$ satisfies requirement formula $\varphi$ at time $t$ and $(\Phi, t) \models \neg \varphi$ signifies failure. More complex formulas can be constructed from combinations of these operators and simpler formulas. For example, consider formula $\varphi_3 = \square_{[t_1, t_2]}(\varphi_1 \land \Diamond_{[t_2, t_3]} \varphi_2)$, which states $\varphi_1$ must hold for all times between $t_1$ and $t_2$ and $\varphi_2$ must occur at some point between $t_2$ and $t_3$. In a similar fashion, the conjunction operator can also be used when multiple requirements, say $i = 1 : N$, must be considered simultaneously in order to construct one large formula, i.e. $\varphi = \bigwedge_{i=1}^{N} \varphi_i$.

One difference of STL over other temporal logic representations is the inclusion of a real-valued robustness degree $\rho^\varphi \in \mathbb{R}$. This robustness degree not only classifies whether the trajectory satisfied the requirement $\varphi$, but also quantifies how well it satisfied the requirement or how badly it failed to meet it. Essentially, $\rho^\varphi > 0$ signifies satisfaction of $\varphi$ while $\rho^\varphi \leq 0$ signifies failures to satisfy $\varphi$.

**Remark 1** Note that $\rho^\varphi = 0$ is slightly ambiguous and is assumed to represent failure to meet $\varphi$, as in [15]; however, $\rho^\varphi = 0$ has been deemed satisfactory in other works [14].

A more detailed description of the robustness degree is found in [13], but a short summary of the terms used in this paper are included below, taken from [14]:

$$\rho^\varphi(t) = \zeta(t)$$

$$\rho^\varphi_{\square_{[t_1, t_2]}\varphi}(t) = \min_{t' \in [t + t_1, t + t_2]} \rho^\varphi(t')$$

$$\rho^\varphi_{\Diamond_{[t_2, t_3]}\varphi}(t) = \max_{t' \in [t + t_2, t + t_3]} \rho^\varphi(t')$$

$$\rho^\varphi_{\square_{[t_1, t_2]} \land \Diamond_{[t_2, t_3]} \varphi}(t) = \min \left( \rho^\varphi_{\square_{[t_1, t_2]} \varphi}(t), \rho^\varphi_{\Diamond_{[t_2, t_3]} \varphi}(t) \right).$$

The term $\rho^\varphi(t)$ refers to the robustness degree measured at time $t$ and $\rho^\varphi(t) > 0$ indicates that $\varphi$ is satisfied for all times starting from time $t$. In this paper, the entire trajectory from $t = 0$ to $t = T_f$ is considered; therefore, $\rho^\varphi(t = 0)$ is used to verify the entire trajectory $\Phi$ satisfies the requirements. For simplicity, the time indices will be dropped for the remainder of this paper, i.e. $\rho^\varphi = \rho^\varphi(t = 0)$.

**Remark 2** The trajectory $\Phi(x(t)|x_0, \theta)$ will change according to $\theta$, which will in turn affect the satisfaction of $\varphi$ and its correspond robustness degree $\rho^\varphi$. To emphasize this fact, the robustness degree will be subsequently written as $\rho^\varphi(\theta)$.

### B. Problem Description

**Problem 1** The goal of the verification procedure is to separate $\Theta$ into the set of all $\theta$ values for which $\rho^\varphi(\theta) > 0$, $\Theta_s$, and the set of all perturbations for which $\rho^\varphi(\theta) \leq 0$, $\Theta_f$. By definition, $\Theta_s \cup \Theta_f = \Theta$ and $\Theta_s \cap \Theta_f = \emptyset$.

In many applications, the system in Eq. 1 is too complex to suitably analyze and verify with any other method than through simulation rollouts. In that case, simulation trajectories are used to observe $\rho^\varphi(\theta)$ and estimate $\Theta_s$ and $\Theta_f$.

**Assumption 1** There exists a sufficiently fine discretization of $\Theta$, $\Theta_d$, at whose $\theta$ values simulations can be performed.

In order for the discretization $\Theta_d$ to suitably capture the potential non-convexity of $\Theta_s$ and $\Theta_f$, the size $|\Theta_d|$ will be large. Even with this discretization of $\Theta$, it will be infeasible to perform simulations at every $\theta \in \Theta_d$. To model this effect, the verification procedure is assumed to be constrained by a computational budget.

**Assumption 2** The simulation-based verification procedure is constrained by a computational budget. This computational budget is assumed to manifest as a cap on the number of simulations, $N_{\text{total}}$, allocated to the verification procedure.

This assumption constraining the number of simulations allocated to the verification is motivated by the work in [19] where a simulator of forest fire dynamics required a large amount of computational resources for the simulation of a single fire trajectory (roughly 1-3 minutes per trajectory on a desktop computer). It would be impractical to run a large number of simulations at every possible condition, especially when a predicted would need to be computed within a reasonable time window (1-2 hours) in realistic scenarios.

### III. Simulation-based Predictions

With the constraints from Assumption 2, the problem now becomes the construction of an accurate approximation of $\Theta_s$ and $\Theta_f$ from Problem 1 while limited to $N_{\text{total}}$ simulation trajectories. Previous work in simulation-based proof techniques bridged the gap between analytical proof techniques and pure data-driven methods; however, these approaches were either unable to produce an accurate estimate of $\Theta_s/\Theta_f$ [10] or failed to guarantee an answer could be produced at all. Instead, this work focuses entirely on the data-driven approach. A regression model can be constructed from observed simulation trajectories and $\rho^\varphi(\theta)$ in order to predict $\rho^\varphi(\theta)$ at all other $\theta \in \Theta_d$ and estimate whether the trajectories will satisfy $\varphi$ or not.

#### A. Gaussian Process Regression Model

A Gaussian process (GP) regression model can be constructed from a training dataset $\mathcal{L}$ of initial observations. This training dataset consists of $N$ pairs of $\theta$ values and their measured $\rho^\varphi(\theta)$ robustness degrees. The set of $N$ observed $\theta$ locations is vector $\theta_{\mathcal{L}} = [\theta_1, \theta_2, \ldots, \theta_N]^T$ while the $\rho^\varphi(\theta)$ are grouped in vector $y_{\mathcal{L}} = [\rho^\varphi(\theta_1), \rho^\varphi(\theta_2), \ldots, \rho^\varphi(\theta_N)]^T$. The training dataset is then $\mathcal{L} = \{\theta_{\mathcal{C}}, y_{\mathcal{L}}\}$.

Gaussian processes have been used extensively to create Bayesian nonparametric models of unknown functions using a finite collection of training data. In short, a GP defines...
a distribution over possible functions that can be used to predict the response over unobserved regions of $\Theta$. A more in-depth discussion of GPs is found in [16]. The end result of the training process is a posterior predictive mean $\mu(\theta)$ and covariance $\Sigma(\theta)$ to estimate $\rho^2(\theta)$, where the estimated $\hat{\rho}^2(\theta)$ is $\hat{\rho}^2(\theta) \sim N(\mu(\theta), \Sigma(\theta))$. The posterior mean and covariance are computed by

$$\mu(\theta) = \kappa(\theta, \theta_L) \kappa(\theta_L, \theta_L)^{-1} y_L$$

$$\Sigma(\theta) = \kappa(\theta, \theta) - \kappa(\theta, \theta_L) \kappa(\theta_L, \theta_L)^{-1} \kappa(\theta_L, \theta),$$

where $\kappa(\theta, \theta_L)$ is a $1 \times N$ vector of $\kappa(\theta_i, \theta_L)$ $\forall i = 1 : N$ and $\kappa(\theta_L, \theta_L)$ is a $N \times N$ matrix of $\kappa(\theta_i, \theta_j)$ $\forall i, j = 1 : N$. The term $\kappa(\theta, \theta')$ refers to the kernel function, of which there are many possible candidates. This paper uses the squared exponential kernel with automatic relevance determination (ARD)

$$\kappa(\theta, \theta') = \sigma^2_f \exp\{-0.5(\theta - \theta')^T D^{-1}(\theta - \theta')\}$$

$$D = \text{diag}(\sigma_1^2, \sigma_2^2, \ldots, \sigma_p^2).$$

The kernel hyperparameters are $\sigma = \{\sigma_f, \sigma_1, \sigma_2, \ldots, \sigma_p\}$. In comparison to the isotropic squared exponential kernel with $\sigma_1 = \sigma_2 = \ldots = \sigma_p$, the ARD kernel enables the hyperparameters to vary and thus deemphasize $\theta$ components with minimal impact upon $\rho^2(\theta)$ or emphasize those with high sensitivity.

**Remark 3** The Gaussian process regression model in Eq. 3 is a noise-free GP [16].

Most Gaussian process models use a homoscedastic Gaussian noise term to model measurement noise in the observations. That assumption does not apply in this application with deterministic simulations of the trajectory and $\rho^2(\theta)$ measurements taken directly from $\Phi(x(t)|x_0, \theta)$. Upcoming work by the authors will address the problem of stochastic simulations.

### B. Prediction Confidence

While the GP regression model predicts the real-valued robustness degree, the verification problem is fundamentally a binary classification problem: concerned with whether $\rho^2(\theta) > 0$ or $\rho^2(\theta) \leq 0$, rather than the exact value of $\rho^2(\theta)$ itself. This highlights the fact that not all predictions are weighted equally. Higher estimation error is acceptable further away from the boundary $\rho^2(\theta) = 0$ than nearby. The predicted satisfaction, $z(\theta|L) = \{1, -1\}$, corresponding to $\rho^2(\theta) > 0$ and $\rho^2(\theta) \leq 0$ is fairly straightforward:

$$z(\theta|L) = \text{sign} \mu(\theta)$$

and $z(\theta|L) = -1$ at $\mu(\theta) = 0$.

The use of STL with real-valued robustness degrees rather than binary measurements such as those used by Metric temporal logic also enables the confidence in the predictions from Eq. 5 to be quantified. Earlier work in data-driven verification [11] focused entirely on binary measures of satisfaction which effectively indicated only the sign of $\rho^2(\theta)$.

Now with the real-valued measurements, the probability of satisfaction at each $\theta$ location can be computed through a Gaussian cumulative distribution function (CDF) using $\mu(\theta)$ and $\Sigma(\theta)$

$$\mathbb{P}(\rho^2(\theta) > 0|L, \sigma) = \mathbb{P}_+(\theta|L, \sigma) = \frac{1}{2} + \frac{1}{2} \text{erf}\left(\frac{\mu(\theta)}{\sqrt{2\sigma(\theta)}}\right).$$

(6)

It is important to note that this probability of satisfaction is dependent upon the training data $L$ and the corresponding GP mean $\mu(\theta)$ and covariance $\Sigma(\theta)$ functions. Changes to the latter two will affect the distribution and thus affect the computed probability. The following subsection will discuss the importance of the hyperparameters used to construct $\mu(\theta)$ and $\Sigma(\theta)$ and their effect on the prediction confidence.

### C. Hyperparameter Optimization

The choice in hyperparameters $\sigma$ greatly affects the GP regression model and the resulting posterior predictive distribution $\hat{\rho}^2(\theta) \sim N(\mu(\theta), \Sigma(\theta))$. In the vast majority of related works [18], [20]–[22], there is little-to-no mention of the choice of hyperparameters. In some works [21], the convergence guarantees of the algorithms presented are dependent upon the fact the ideal (true) choice of hyperparameters is known apriori and fixed. In practice, this assumption cannot be made as little is known about the distribution of the robustness degree over $\Theta$ until after simulations are performed. Instead, the hyperparameters must be chosen with only the current available information, training dataset $\mathcal{L}$.

The distribution of the hyperparameters with respect to the training set $\mathcal{L}$ is given by

$$\mathbb{P}(\sigma|y_L, \theta_L) = \frac{\mathbb{P}(y_L|\theta_L, \sigma)\mathbb{P}(\sigma)}{\int \mathbb{P}(y_L|\theta_L, \sigma)\mathbb{P}(\sigma)d\sigma}.$$ (7)

Without prior information about the choice of $\sigma$, the distribution $\mathbb{P}(\sigma)$ can be assumed to be uniform. The issue with Eq. 7 is that there is no closed-form solution to compute $\mathbb{P}(y_L|\theta_L, \sigma)$ so sampling-based approximation or Markov Chain Monte Carlo (MCMC) methods must be used to construct the probability distribution [16]. However, such approaches are intractable for the resource-constrained problem considered here; every sample from $\mathbb{P}(y_L|\theta_L)$ requires the inversion of the $N \times N$ matrix $\kappa(\theta_L, \theta_L)$, an $O(N^3)$ operation. Instead, hyperparameter optimization can be used to approximate the maximum likelihood estimate (MLE) of the hyperparameters.

From Eq. 7, the hyperparameter distribution $\mathbb{P}(\sigma|\theta_L, y_L)$ is proportional to $\mathbb{P}(y_L|\theta_L)$; therefore, the MLE of $\mathbb{P}(y_L|\theta_L)$ will also correspond to the MLE of $\sigma$. A local maximum to $\mathbb{P}(y_L|\theta_L)$ can be computed using steepest descent methods as in [16] or gradient-based methods from [23]. As mentioned earlier, the ARD squared exponential kernel enables less-influential $\theta$ components to be deemphasized. When little is known about the hyperparameters apriori, this optimization procedure is particularly useful for discovering these less-influential components from the observed data $\mathcal{L}$ and subsequently adjusting the sensitivity. Even though the optimization framework can only guarantee local optimums,
the results in Figure 6 of Section V indicate hyperparameter optimization will provide improved prediction error over static hyperparameters.

The choice in hyperparameters will also affect the CDF from Eq. 6 describing the probability of satisfaction of \( \varphi \). If the true hyperparameters, labeled \( \sigma^* \), are known, then the actual probability \( P(\varphi > 0|L) = P(\rho^2(\theta) > 0|L, \sigma^*) \); however, this is a very restrictive assumption. In the more likely case when the true hyperparameters \( \sigma^* \) are not known, the the probability of satisfaction is the marginal likelihood over all possible \( \sigma \)

\[
P(\varphi > 0|L) = \int P(\varphi > 0|L, \sigma)P(\sigma|y_L, \theta_L)d\sigma.
\]

Since this integral requires \( P(\sigma|y_L, \theta_L) \), the computation of the total probability in Eq. 8 is intractable as well. Since there is no closed-form solution, the same approximation or sampling methods must be used. All the effort used to approximate \( P(\sigma|y_L, \theta_L) \) could be used to obtain additional simulations.

IV. Active Sampling

Given a set of training data, a data-driven statistical model can be constructed to provide predictions over the entire \( \Theta \) space; however, these predictions are highly dependent upon the training dataset \( L \). With \( L \) limited to \( N_{total} \) Samples, it is crucial that the best possible set is chosen, especially when \( N_{total} \) is small compared to \( |\Theta_d| \). As would be expected, the difficulty lies in the fact that the ideal training set is unknown ahead of time. Rather than passively select training datapoints, either randomly or in a structured grid, training samples can be actively chosen in a closed-loop fashion.

Active learning procedures [12] include a variety of different approaches, all of which attempt to identify the “best” samples to obtain in order to improve a statistical model. Depending on the application and objectives, the definition of the “best” sample will change, even for the same exact model. One of the most general approaches is the expected model change (EMC)/uncertainty sampling metric [17]. Though widely-applied to support vector machine (SVM)-based models, EMC also applies to Gaussian process regression models. The underlying selection metric for the “best” sample \( \overline{\theta} \) is the point most likely to induce the largest expect change in the model; in short, this means \( \overline{\theta} = \text{argmax } |\mu(\theta)| \). Another approach is variance reduction [12], [18]. While various approximations or derivations exist [18], the end goal is to select \( \overline{\theta} \) which most reduces the posterior variance of the GP model. Since it is comparatively expensive, although possible, to calculate the change in posterior variance, the most common approximation is to select \( \overline{\theta} = \text{argmax } \Sigma(\theta) \). Regardless of the efficiency or accuracy of the approaches or their approximations, none of these approaches directly address the underlying objective in this verification problem: to predict whether \( \rho^2(\theta) > 0 \) or not.

Instead of the other selection metrics, binary classification entropy is used to identify informative sample locations.

Unlike variance methods, which may also use the term “entropy” [18], binary classification entropy considers the probability \( P(\rho^2(\theta) > 0) \) from Eq. 6 rather than the entropy of \( \Sigma(\theta) \). The binary classification entropy at a location \( \theta \) for a given \( (L, \sigma) \) is

\[
H(\theta|L, \sigma) = -\left( P_+(\theta|L, \sigma) \log_2 P_+(\theta|L, \sigma) + (1 - P_+(\theta|L, \sigma)) \log_2 (1 - P_+(\theta|L, \sigma)) \right).
\]

The total entropy over \( \Theta_d \) can be determined by summing Eq. 9 over all unobserved \( \theta \) locations (entropy \( \forall \theta \in L \) is 0). This set of unobserved sample locations is listed as \( U = \Theta_d \setminus \theta_L \)

\[
H(U|L, \sigma) = \sum_{i=1}^{|U|} H(\theta_i|L, \sigma)
\]

A. Sequential Sampling

Ideally, the best sample location \( \overline{\theta} \) would induce the largest possible change in total entropy (Eq. 10).

\[
\overline{\theta} = \text{argmax } \theta \in U \left( H(U|L \cup \{\theta', y(\theta')\}, \sigma) - H(U|L, \sigma) \right).
\]

While it is possible, albeit computationally expensive, to compute the posterior reduction in variance, it is actually impossible to compute the posterior classification entropy \( H(U|L \cup \{\theta', y(\theta')\}, \sigma) \) in Eq. 11. The computation of the total posterior classification entropy requires the mean function \( \mu(\theta) \) in order to determine the corresponding \( P(\rho^2(\theta) > 0|L \cup \{\theta', y(\theta')\}, \sigma) \). Since the posterior predictive mean requires the \( \rho^2(\theta) \) measurement at each training location, the measurement \( \rho^2(\overline{\theta}) \) would have to be known in order to compute the entropy. If this measurement were already known, there would be no need to choose that location as a future sample. Note that variance-focused methods are able to compute the posterior reduction in the variance, albeit expensively, precisely because variance \( \Sigma(\theta) \) is independent of the actual measurements and only depends on the measurement locations.

In place of the infeasible selection criteria proposed in Eq. 11, a new selection metric is given below:

\[
\overline{\theta} = \text{argmax } \theta \in U \left( H(\theta'|L \cup \{\theta', y(\theta')\}, \sigma) - H(\theta'|L, \sigma) \right)
\]

In this metric, only the current measurements are used, thus avoiding the problem from before. This approach selects the sample location from \( U \) with the largest magnitude of classification entropy because it will have the largest change in local entropy. Once a sample is taken, the entropy at that location is 0; therefore, the point with the largest magnitude of classification entropy would have the largest change. Note that the binary classification entropy is strictly non-positive.
A closed-loop sampling procedure using the selection metric from Eq. 12 is listed in Alg. 1. As the process must first start with an initial model in order to compute \( H(\theta | \mathcal{L}, \sigma) \), the algorithm assumes a GP model has been trained from some initial training set \( \mathcal{L} \). Assuming this initial training set counts against the sampling budget \( N_{\text{total}} \), there are \( T \) additional samples that can be selected, where \( T = N_{\text{total}} - |\mathcal{L}| \). With the initial GP model, the algorithm first computes the binary classification entropy (Step 3). Steps 4-8 select the “best” sample \( \theta \), obtain the measurement at that location, update \( \mathcal{L} \) and \( \mathcal{U} \) and retrain the GP model. As part of the retraining process in Step 8, the hyperparameters are re-optimized to reflect the new \( \mathcal{L} \) with the new sample. This process in Steps 3-8 is repeated until the sampling budget is filled.

**Algorithm 1 Sequential active sampling algorithm**

1. **Input:** training set \( \mathcal{L} \), available sample locations \( \mathcal{U} \), \# of additional samples \( T \), trained regression model \( \mathcal{GP} \)
2. for \( i = 1 : T \) do
3. Compute entropy \( H(\theta | \mathcal{L}, \sigma) \) \( \forall \theta \in \mathcal{U} \)
4. Select sample \( \theta \) from \( \mathcal{U} \) according to Eq. 12
5. Run simulation, obtain measurement \( y(\theta) \)
6. \( \mathcal{L} \leftarrow \mathcal{L} \cup \{ \theta, y(\theta) \} \)
7. \( \mathcal{U} \leftarrow \mathcal{U} \setminus \{ \theta \} \)
8. Retrain regression model \( \mathcal{GP} \) with new \( \mathcal{L} \)
9. end for

**B. Batch Sampling**

While the sequential sampling procedure in Alg. 1 will correctly guide the selection of sample points as intended, it requires a large amount of computational effort to continuously recompute the GP after each iteration. Due to the matrix inversion, the cost of each retraining step in Step 8 is \( \mathcal{O}(|\mathcal{L}|^3) \) or \( \mathcal{O}(|\mathcal{L}|^2) \) at best, and some multiple of \( \mathcal{O}(|\mathcal{L}|^3) \) when hyperparameter optimization is performed. Additionally, the cost of predicting the satisfaction of \( \varphi \) and computing the entropy at all prospective sampling locations \( \theta \in \mathcal{U} \) is \( \mathcal{O}(|\mathcal{L}|^2|\mathcal{U}|) \). While Assumption 2 assumes that the primary source of computational cost is the simulation trajectories, the cost associated with each training and prediction step is non-negligible and grows with \( |\mathcal{L}| \). Thus, it is also best to minimize the number of retraining steps as much as possible without sacrificing accuracy.

One of the most practical methods for reducing the retraining cost is to obtain samples in batch - select multiple \( \mathcal{B} \) samples in between retraining steps. Batch active sampling methods [12], [17], [21], [22] are a common tool to lower the training cost and can also inherently take advantage of any parallel computing capabilities to compute multiple simulation trajectories in parallel. In these approaches, \( M \) datapoints are chosen at once and their corresponding trajectories are performed in parallel. The challenge with such approaches is ensuring adequate diversity in the chosen set of \( \mathcal{B} \) locations to avoid the repeated selection of the same point.

**1) Batch Version of Algorithm 1:** A straightforward conversion of Algorithm 1 into a batch procedure is shown in Algorithm 4 in Appendix VII-A. As part of the procedure in Alg. 4, the remaining budget allocated to simulations after the initial training set is assumed to be broken evenly into \( T \) batches of \( M \) points each, for \( T \times M = N_{\text{total}} - |\mathcal{L}| \). A detailed discussion of the procedure is found in the appendix. As touched upon in that discussion, there are two main issues with the procedure in Alg. 4.

First, it is impossible to correctly measure the effects of points previously selected in the batch when choosing the remaining points of the batch. The sample locations previous selected for the batch are stored in a temporary set \( \mathcal{S} \). One way to encourage diversity is to update the entropy \( H(\theta | \mathcal{L}, \sigma) \) to reflect the change in \( \mathcal{P}_+(\theta | \mathcal{L}, \sigma) \) caused by the preceding points in \( \mathcal{S} \). While it is possible to compute the change in \( \Sigma(\theta) \) without the corresponding measurements \( y_S \), it is impossible to update \( \mu(\theta) \) without those. Algorithm 4 approximates the change in entropy by updating \( \Sigma(\theta) \) with each additional point in \( \mathcal{S} \) but freezes the mean \( \mu(\theta) \) to the values from the start of the batch. However, this scales poorly with \( M \) since the approximate entropy further diverges from the true entropy with each additional point in \( \mathcal{S} \).

The second problem with Alg 4 is the high computational load. Because \( \Sigma(\theta) \) is updated and the entropy is recomputed \( \forall \theta \in \mathcal{U} \) (Step 7 in Alg. 4), each additional point in \( \mathcal{S} \) requires \( \mathcal{O}(|\mathcal{L}|^3) + \mathcal{O}(|\mathcal{L}|^2|\mathcal{U}|) \) operations, assuming best-case matrix inversion and hyperparameters \( \sigma \) are fixed during the batch. Upon completion of the batch, the hyperparameters will have to be re-optimized with the new training dataset \( \mathcal{L} \leftarrow \mathcal{L} \cup \{ \mathcal{S}, y_S \} \) and incur the same \( \mathcal{O}(|\mathcal{L}|^3) + \mathcal{O}(|\mathcal{L}|^2|\mathcal{U}|) \) cost as before in Step 8 of Alg 1. While this approach still ultimately allows for parallel simulations to be performed and introduces the same computational savings associated with those, the cost of selecting the \( \theta \) perturbations for those simulations is non-negligible.

**2) Related Work:** Just as there are multiple sequential sampling selection metrics, numerous batch sampling methods have been proposed, usually as a derivate of the aforementioned sequential procedures. Batch EMC algorithms [12], [17] incorporate a diversity measure based upon temporary RBF kernels placed at each \( \theta \in \mathcal{S} \) to penalize points in close proximity to \( \theta \in \mathcal{S} \). Batch variance methods are mostly unchanged from their sequential versions [12], [18]. As discussed in the proceeding paragraphs, the covariance function can be updated for \( \mathcal{S} \) without requiring the corresponding measurements. Therefore, the change in variance is explicitly known before the simulations are even performed and the selection metric need not be modified.

The closest prior work is the sampling procedure in [21]. Gaussian process regression models are used to identify level sets and active learning identifies future training points to improve level set estimation. The approach uses the mean function along with confidence intervals based upon the covariance function to predict level sets. The procedure addresses a variant of batch processes not considered by this paper: measurements of selected samples are delayed.
by a set number of iterations, much like a time delay. While not perfectly suited to the problem here, the approach does face similar issues with selection of training points before measurements are available at all previously selected sample locations. It was shown that the information gain can be bounded by a fixed quantity and bounded the change in confidence intervals even without measurements at the “time-delayed” sample locations.

Unfortunately, this bound on the information gain is not appropriate to the verification problem considered in this work. First, the bound on information gain strictly assumes the ideal hyperparameters are known and thus the hyperparameters are static and not updated. Additionally, the information bound is inversely proportional to the square of the measurement noise. For problems were there is little-to-no measurement noise, as is the case for deterministic simulations of Eq. 1, the bound on the information gain is infinite and of no use. Lastly, the procedure requires updates to the covariance function at each iteration, much like Alg. 4 and batch-variance methods, and therefore suffers from the same heavy computational overhead.

3) Importance Weighting: Rather than rely upon the expensive and flawed batch approximation of the sequential sampling algorithm (Alg. 4), two parallel approaches for batch active sampling are introduced. Both of these approaches utilize importance weighting, essentially importance sampling [24] without the subsequent probability estimate, to efficiently select samples using only the current entropy. Unlike the other batch procedures that update the covariance function after the selection of each point in \( S \), these two approaches freeze the information and select the entire batch of \( M \) samples without any intermediate retraining. The crux of the procedures is importance-weighted random sampling: samples for \( S \) are randomly chosen based upon some probability distribution. Here, the current entropy \( H(\theta|\mathcal{L}, \sigma) \) is used to construct the probability distribution from which samples are chosen, where regions with a large magnitude of entropy will have a higher probability of selection.

The more straightforward of the two approaches is the baseline importance-weighted sampling procedure shown in Alg. 2. Given some initial training set \( \mathcal{L} \) and an initial GP model trained from this \( \mathcal{L} \), the entropy \( H(\theta|\mathcal{L}, \sigma) \) can be readily determined (Step 3). This entropy can then be used to construct a probability distribution over \( \mathcal{O}_H, \mathcal{P}_H(\theta) \)

\[
\mathcal{P}_H(\theta) = \frac{1}{Z_H} H(\theta|\mathcal{L}, \sigma) \quad \text{where} \quad Z_H = \sum_{i=1}^{||\mathcal{O}_H||} H(\theta_i|\mathcal{L}, \sigma).
\]

This k-DPP sample selection process can then be inserted into the batch active sampling procedure, shown in Alg. 3.

Algorithm 2 generates samples that are generally clustered in or around regions with large \( |H(\theta|\mathcal{L}, \sigma)| \); however, it does face some issues ensuring adequate diversity in the set of samples. Particularly when batch size \( M \) is low, it is desirable to spread samples out across regions with similarly-high probability. The problem is that the randomly-selected samples may inadvertently cluster into one area rather than spread out, thus lacking diversity in the sample set. In order to address this issue, importance-weighting can be augmented with random matrix theory (RMT) methods to encourage diversity in the samples while still steering samples towards regions with high probability/entropy.

The main tool used to encourage diversity is Determinantal Point Processes (DPP) [25]. DPPs are probabilistic models for efficient sampling that penalize correlations in the samples and thus can be used to discourage similarities between samples. More theoretical discussions can be found in the original literature [25], [26], but an overview of the process with respect to the data-driven verification problem is included in Algorithm 5 of Appendix VII-B. Essentially, \( M_T \) samples from \( \mathcal{P}_H(\theta) \) are used to construct a kernel matrix \( \mathcal{L} \) and form the DPP. Here, \( M_T \) must be large enough for \( \mathcal{L} \) to adequately represent \( \mathcal{P}_H(\theta) (M_T \approx 1000) \). Because \( M_T \) must be so large, but \( M \) is generally small, a variant of DPPs called k-DPPs [26] can be used to obtain \( M \ll M_T \) samples in the correct manner. This k-DPP sampling procedure sequentially selects \( M \) points from the set of \( M_T \) samples. Once a sample has been selected, the remaining samples are weighted towards points with corresponding eigenvectors of \( \mathcal{L} \) that are orthogonal to that recently-selected sample. This k-DPP sample selection process can then be inserted into the batch active sampling procedure, shown in Alg. 3.

The majority of the procedure is unchanged from the baseline importance-weighting procedure in Alg 2. The main difference is that \( \mathcal{P}_H(\theta) \) is used to generate many more samples (\( M_T \gg M \)) in order to construct the k-DPP in Step 6. This k-DPP sample selection process does require some additional computational overhead in comparison to the baseline importance-weighted algorithm. The cost of constructing and sampling from the k-DPP requires an additional
\(O(M_T^3) + O(M_T M^3)\) operations per batch. While this is higher than Algorithm 2, for small \(M\) and \(M_T << |U|\), Algorithm 3 is still faster than the batch-sequential algorithm in Alg. 4. A visual comparison of importance-weighted random sampling and the RMT-based sampling is shown in Figure 1, which depicts the resulting set of 30 samples for each approach starting from the same initial \(P_H(\theta)\). Note that the samples generated by the k-DPP are much more diverse than the baseline samples. The results in the next section will compare the actual performance and cost of Algorithms 2 and 3.

Algorithm 3 Batch active sampling using random matrix theory (RMT) sampling.

1: **Input**: training set \(\mathcal{L}\), available sample locations \(U\), trained regression model \(GP\), \(T\) batches, \(M\) points in each batch, empty set \(S\)
2: for \(i = 1 : T\) do
3: Compute entropy \(H(\theta|L, \sigma) \forall \theta \in U\)
4: Transform \(H(\theta|L, \sigma)\) into probability distribution 
5: Obtain \(M_T\) random samples of \(\theta\) from \(P_H(\theta)\)
6: Construct k-DPP from the \(M_T\) samples and obtain \(M\) samples from the k-DPP (see Algorithm 5)
7: Add \(M\) samples to set \(S\)
8: Run simulations \(\forall \theta \in S\), obtain \(y_S\)
9: \(L \leftarrow L \cup S, y_S\)
10: Reinitialize empty \(S\), retrain \(GP\) with new \(L\)
11: end for

V. EXAMPLES

The sampling algorithms are demonstrated on two example problems of interest. Both are deterministic nonlinear systems with structured uncertainties. The first, an adaptive controller, is a 2D problem \((\theta \in \mathbb{R}^2)\) and is used to visualize the evolution of the sampling algorithms, highlight their differences, and discuss the importance of hyperparameter estimation. The second example, a heading autopilot for a 6 degree of freedom aircraft model, demonstrates the process on a very complex problem with control saturation, sensor models, and switching logic in addition to nonlinearities in the open-loop plant.

A. Adaptive Controller

The first example is a concurrent learning model reference adaptive control (CL-MRAC) system [27]. This problem has been examined before with other approaches, namely proof-based techniques that produced overly-conservative analytical certificates [10] and simulation-hungry, passive data-driven certificates [11]. The following subsection will demonstrate that entropy-based active sampling procedures can produce accurate data-driven certificates of satisfaction without relying upon large datasets. In this problem, the open-loop plant dynamics are given by the uncertain second-order system

\[
\dot{x}_1 = x_2, \quad (14)
\]

\[
\dot{x}_2 = (-0.2 + \theta_1)x_1 + (-0.2 + \theta_2)x_2 + u(t).
\]

The perturbations \(\theta = [\theta_1, \theta_2]^T\) are uncertain parameters of the system. The goal of the system is to track a linear reference model with a desirable second-order response

\[
\dot{x}_{m_1} = x_{m_2} \quad (15)
\]

\[
\dot{x}_{m_2} = -\omega_n^2 x_{m_1} - 2\zeta_n \omega_n x_{m_2} + \omega_n^2 z_{cmd}(t),
\]

with \(\zeta_n = 0.5\) and \(\omega_n = 1\). This reference model is excited with reference commands \(z_{cmd} = 1\) between 0 and 2 seconds, \(z_{cmd} = 1.5\) between 10 and 12 seconds, and \(z_{cmd} = -1.5\) between 20 and 22 seconds. At all other times in the 40 second trajectory length \((T_f = 40)\), the reference command is set to 0. The tracking error, the distance between the actual and reference trajectories, is given by \(e(t) = x_m(t) - x(t)\). In the absence of uncertainties, i.e. \([\theta_1, \theta_2] = [0,0]\), a linear feedback controller

\[
u_{pd}(t) = K_p e_1(t) + K_d e_2(t) \quad (16)
\]

is suitable for ensuring closed-loop tracking, with controller gains set to \(K_p = 1.5\) and \(K_d = 1.3\).

Since the parameters \([\theta_1, \theta_2]\) are assumed to be uncertain, the adaptive controller is introduced to estimate these parameters online and ensure the system tracks the desired reference trajectory. The nominal control input \(u(t)\) is augmented with an additional adaptive input \(u_{ad}(t)\) consisting...
of online estimates of the parameters

\[ u_{ad}(t) = \begin{bmatrix} \hat{\theta}_1(t) \\ \hat{\theta}_2(t) \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}. \] \tag{17}

These estimated parameters \( \{\hat{\theta}_1(t), \hat{\theta}_2(t)\} \) are updated online according to the concurrent learning adaptive law, a special derivative of the standard MRAC adaptive law. In short, the CL-MRAC adaptive law ensures that both the tracking error \( e(t) \) and the parameter estimation error \( \hat{\theta}(t) = \hat{\theta}(t) - \theta \) will converge to 0 within a finite amount of time without requiring persistency of excitation in the reference commands \( z_{cmd}(t) \). More details on the CL-MRAC adaptive law can be found in [27]. As a result of the adaptive law and the uncertainties, the closed-loop dynamics are nonlinear and difficult to analyze.

While the CL-MRAC adaptive law ensures the asymptotic convergence of the tracking and parameter estimation errors, and thus also the Lyapunov stability of the state \( x(t) \) trajectory, it is difficult to prove boundedness of the transient errors. In particular, the performance requirement considered in this example is that the tracking error \( e(t) \) remains bounded within the unit ball at all times. This can be written in STL format as

\[ \varphi_{\text{bound}} = \square_{[0,T]} (1 - |e[t]| \geq 0). \] \tag{18}

Note that Eq. 18 displayed the requirement in its most-compact, easy-to-read form; however, \( \varphi_{\text{bound}} \) can also be written in terms of \( x[t] \) and \( x_{\text{rm}}[t] \) to emphasize that only \( x(t) \) changes with \( \theta \). For each simulation, the robustness degree \( \rho^\varphi(\theta) \) can be written as:

\[ \rho^\varphi(\theta) = \min_{t' \in [0,T]} \rho^{\varphi_{\text{bound}}[t']}(\theta). \] \tag{19}

The goal of the active sampling procedures is to measure \( \rho^\varphi(\theta) \) at various training locations and predict \( \mathbb{P}(y(\theta) > 0) \) at unobserved \( \theta \) locations in \( \Theta_d \). For this example, \( \Theta_d \) covers the space between \(-10 \leq \theta_1 \leq 10 \) and \(-10 \leq \theta_2 \leq 10 \) in increments of 0.1 for a total of 40,401 possible sample locations in \( \Theta_d \). It is unlikely \( N_{\text{total}} = 40,401 \) in most computationally-constrained scenarios, so the samples must be carefully selected to improve the predictions.

1) Sequential Sampling: First, the sequential sampling procedure from Alg. 1 is demonstrated and compared against existing techniques discussed in Section IV. Additionally, \( M = 1 \) versions of the batch procedures in Algs. 2 and 3 are also included for comparison to the sequential algorithm.

To illustrate the sampling process, Figures 2 and 3 display one iteration of the for-loop in Algorithm 1. An initial training set of 50 randomly selected samples is used to construct the prediction model shown in Figure 2(a). The (unknown) true model of \( \rho^\varphi(\theta) \) is shown in Fig. 2(b). The prediction model is used to compute the binary classification entropy shown in Fig. 3. The next training sample is selected according to Eq. 12 from Step 4 of Alg. 1, highlighted as the magenta star. Once the measurement at this \( \bar{\theta} \) has been obtained, the prediction model is retrained with this additional information.

Figure 4 compares the performance of Algorithm 1, averaged over 100 random initializations, against the average performance of existing passive, expected model change (EMC), and variance reduction methods. The prediction error corresponds to misclassifications of \( \rho^\varphi(\theta) > 0 \) or \( \rho^\varphi(\theta) \leq 0 \): whether points that actually satisfy \( \varphi_{\text{bound}} \) were predicted to not satisfy or points that do not satisfy were incorrectly predicted to do so. The sequential entropy-based sampling algorithm outperforms all the other approaches. The approach also outperforms the importance-weighted random sampling procedures from Algs. 2 and 3. Because
only one sample is selected, the k-DPP random matrix theory (RMT) procedure behaves exactly like the baseline importance-weighting algorithm.

2) Batch Sampling: In many applications, it is too computationally-expensive to retrain the GP model after each individual measurement is selected and obtained. In these problems, it is more advantageous to select points in batches of multiple samples and perform simulations at these points all at once. This can also take advantage of parallelization more effectively than sequential algorithms as the simulations can be done in parallel. While it is more efficient, batch sampling must be carefully arranged so as to ensure diversity in the sample set while still selecting relevant points. Section IV-B proposed multiple algorithms based upon binary classification entropy to address those limitations. A comparison of sample selections using Alg. 2 and Alg. 3 for this problem was actually shown earlier in Figure 1. The two approaches were initialized with the same training set and GP model from Fig. 2 and each algorithm forms a relatively large dataset of 30 points. In practice, such a large number of datapoints is not advised during the initial few iterations because the model changes rapidly with new training batches; however, the figure does effectively illustrate the difference between the two selection processes.

The performance of the batch approaches is shown in Figure 5. The two batch algorithms are compared against batch versions of random passive sampling and the EMC algorithm. Two comparable batch versions of the sequential variance reduction method were also developed and their results are plotted in the figure. These batch variance reduction methods are effectively the same as Algorithms 2 and 3 except the probability distribution $P_H(\theta)$ is instead based upon the current variance $\Sigma(\theta)$ rather than binary classification entropy. The figure illustrates the performance at batch sizes of $M = 5$ and $M = 10$ points in each batch.

In both plots, the entropy-based sampling methods are shown to outperform the existing techniques. The k-DPP RMT procedure performs slightly better than the baseline importance-weighting technique. This is not unexpected as the comparison in Figure 1 demonstrated that RMT sampling would spread the samples more diversely, albeit for an exaggerated case. Regardless, the utility of the entropy-based sampling algorithms for simulation-based verification has been established in both sequential and batch sampling settings.

3) Hyperparameter Optimization: As was discussed in Section III-C, the true hyperparameters are not assumed to be known apriori and therefore the hyperparameters must be estimated from the current training set $\mathcal{L}$. Especially during the initial steps of the process, the new measurements obtained the simulation trajectories can drastically differ from the predicted response from the GP model. While the measurements themselves will change the GP prediction model regardless of whether or not the hyperparameters are varied, the new information added by the recent samples can also change the distribution $P(\sigma | y_{\mathcal{L}}, \theta_{\mathcal{L}})$ in Eq. 7, requiring the hyperparameters to be re-optimized. The downside of hyperparameter optimization is that it requires multiple $O(|\mathcal{L}|^3)$ operations to obtain a local optimum.

In order to illustrate the worth of hyperparameter optimization, Figure 6 compares two versions of Algorithms 2 and 3, one with hyperparameter optimization and one with fixed $\sigma$. The results indicate that fixing the hyperparameters when the true/ideal $\sigma$ are unknown can lead to suboptimal performance, even when active sampling is performed. Note
that since the mean and covariance of GPs are dependent upon the hyperparameters, the entropy will change and the points selected for the next batch will change accordingly. Thus, the training sets will diverge after the first iteration. For both the baseline importance-weighting and RMT procedures with static hyperparameters, the prediction error is actually worse than it was with the initial, smaller training set. After a number of iterations, the prediction error begins to decrease, although this may be due to the fact the size of $\mathcal{L}$ begins to become a noticeable larger fraction of the total $\Theta_d$ set. Figure 6 only plots the results for a single batch size of $M = 10$, but additional runs at various batch sizes and other approaches produced consistent results and behaviors.

![Figure 6: Effect of fixed versus hyperparameter optimization. For the dashed lines, the hyperparameters are held constant from their initial setting while the solid lines depict the procedure where the hyperparameters are re-optimized after each iteration.](image)

B. Heading Autopilot

The preceding MRAC example clearly shows that active sampling, in particular active sampling with entropy-based selection metrics, can be used to reduce the prediction error without requiring large numbers of additional samples. The MRAC example is also included because it is 2D and can then be used to easily illustrate the prediction model and closed-loop sampling processes (Figs. 1-3). While the results in the preceding subsection demonstrate between a 75.2 – 77.8% reduction in prediction error from the initial model using entropy-based active sampling, they only demonstrate a 17.5 – 26.4% improvement over existing active sampling procedures. The following example will apply the procedures to a significantly more complex dynamical system and the entropy-based procedures will also demonstrate roughly twice the level of improvement over existing procedures than in the MRAC example.

This second example is verification of an aircraft autopilot for controlling lateral-directional flight modes. In particular, the “heading hold” autopilot mode is used to turn the aircraft to and hold a desired reference heading. The simulation model of the autopilot and DeHavilland Beaver airframe is provided by the “Aerospace Blockset” toolbox in Matlab/Simulink [28], [29]. This simulation model includes numerous nonlinearities such as the full nonlinear aircraft dynamics, nonlinear controllers, and actuator models with position and rate saturations. The provided “heading hold” autopilot then has several various requirements the closed-loop system to satisfy.

The requirement chosen for this example is an “altitude hold” requirement of the heading autopilot. In addition to turning the aircraft to the desired heading angle, the autopilot must also ensure the aircraft’s altitude remains within 35 feet of the initial condition when the reference command was given [28], [29]. This requirement can be given in temporal logic form as

$$\varphi_{\text{height}} = [0, T_f](35 - |x[t] - x[0]| \geq 0), \quad (20)$$

where $T_f$ is the final simulation time and $x[t]$ is the aircraft altitude (in feet) at time $t$. For this problem, $T_f = 50$ seconds. The STL robustness metric is similar to the last example with

$$\rho^\ast(\theta) = \min_{t' \in [0, T_f]} \rho^\varphi_{\text{height}}(t')(\theta). \quad (21)$$

The satisfaction of $\varphi_{\text{height}}$ was tested against different initial conditions of the Euler angles, $\Theta^T = [\text{roll}(0), \text{pitch}(0), \text{heading}(0)]. While satisfaction of other aspects of the heading hold autopilot can be explored, such as heading angle overshoot or steady-state tracking error, the altitude hold requirement dominated the other requirements during an initial trade-space exploration. The space of allowable perturbations ($\Theta$) spans $\text{roll}(0) \in [-60^\circ, 60^\circ]$, $\text{pitch}(0) \in [4^\circ, 19^\circ]$, and $\text{heading}(0) \in [75^\circ, 145^\circ]$. The desired reference heading was kept constant at $112^\circ$. The discrete sampling set $\Theta_d$ discretizes $\Theta$ into $1^\circ, 0.25^\circ$, and $1^\circ$ increments corresponding to the three Euler angles, resulting in a total of 524,051 possible sampling locations.

The performance of the different sampling procedures is shown in Figure 7. For each procedure, these results are averaged over 100 different runs with a randomly selected initial training set of 100 samples. Each procedure starts with the same 100 different training sets as the others. The active learning procedures then select 400 additional training samples out of the remaining $|\mathcal{U}| = 523,051$ available sample locations. Just as was seen in the previous example, the entropy-based procedures outperform the existing active learning procedures. For both batch sizes ($M = 5$ and $M = 10$), the entropy-based procedures produce below 3% prediction error after 500 iterations, about a 40% reduction over the closest variance-reduction method.

One important thing to note about this example is that a sequential approach strategy similar to Algs. 1 or 4 is not practical in most computationally-constrained environments when compared to batch methods. Due to the large size of $|\mathcal{U}|$, recomputation of the variance and entropy after the selection of every sample ($O(|\mathcal{L}|^2|\mathcal{U}|)$ operations) greatly increases the cost. A comparison of the prediction and training cost for sequential and batch methods is shown in Figure 8. The EMC and importance weighting procedures (for both entropy and variance) all have the same computational cost for training the GP and computing $\mathbb{P}(\rho^\ast(\theta) > 0)$, while the RMT methods have a slightly higher cost due to the
Fig. 7: Comparison of batch sampling algorithms at different batch sizes applied to the heading autopilot example. The performance is averaged over 100 difference runs with random initializations.

Fig. 8: Comparison of training and prediction costs for the autopilot example. Note that all of these costs do not include the cost of obtaining samples, only the cost of training the GP and computing $P(\rho^c(\theta) > 0)$. The cost for “best-case” passive sampling assumes all $T$ samples are taken at once and not sequentially or in batches.

VI. CONCLUSION

This work has presented data-driven verification methods for construction of simulation-based certificates of complex nonlinear systems. Given a suitable simulation model of the system, it is relatively straightforward to perform simulations at various uncertainties and use these to predict the system’s ability to satisfy performance requirements. Unlike previous analytical verification techniques, which relied upon the existence of known analytical functions to bound the system’s trajectories, simulation-based methods make no such assumptions and apply to a wider class of systems. The ultimate goal is to construct a statistical certificate that correctly classifies whether the closed-loop response at particular uncertainties will satisfy the requirements.

Methods for converting these qualitative requirements into quantifiable formula were given in Section II. Temporal logic predicates are shown to be a natural approach for expressing the requirements. Signal temporal logic (STL) can be used to produce a single, real-valued, scalar measurement of the satisfaction of the requirements over an entire trajectory. These real-valued measurements not only classify whether the trajectory continuously met all requirements, but also quantify the minimum robustness (or maximum exceedance) at some point in the entire trajectory. A set of the real-valued measurements at different uncertainties is used to construct a Gaussian process (GP) regression model (Section II). This model predicts the STL robustness measurement at unobserved locations along the entire space of feasible uncertainties ($\Theta$). The predicted STL measurements can then be used to estimate whether the corresponding trajectory would in fact satisfy the requirements and also measure the confidence in that estimate.

In many applications, the simulations themselves can be computationally expensive to obtain; therefore, it is advantageous to minimize the number of simulations required to obtain accurate predictions. The core of the sample-efficient approach is the active sampling procedures presented in Section IV. All of these procedures utilize binary classification entropy as the metric for selecting future training locations. In addition to a sequential algorithm, two batch sampling algorithms are presented to select multiple training locations at once. These batch methods directly exploit any parallelization in the simulation environment and can be used for additional computational savings. Two examples in Section V demonstrated the improvement in prediction error using the active sampling procedures over other approaches given a limited number of simulations.

Ultimately, data-driven verification procedures are aimed at use within higher-level problems such as robust, nonlinear planning or controller optimization. In those those problems, verification is performed on each candidate control policy.
in order to estimate their robustness and the generation of candidate control policies is often an iterative process. Thus, the process would typically cycle through a large number of candidate control policies and it is infeasible to sample the full discretized space of all possible uncertainties for every candidate policy. Sample-constrained data-driven verification aligns with those problems by providing the designer with the best approximation of the robustness while restricted to a budget on the number of allowable simulations during each verification step of the iterative process.

While this work has clearly illustrated the potential of data-efficient, simulation-based verification procedures for certification of uncertain nonlinear systems, there remain a few unanswered problems. The focus of this work was on deterministic systems: given the same initial conditions and uncertainties $\theta$, all trajectories and their corresponding STL measurements will be the same. In many applications, stochasticity will be present and can be incorporated into the simulation model. In these problems, trajectories and measurements will no longer be deterministic. Current work in progress has developed similar modeling and sampling procedures for stochastic nonlinear systems, explicitly leveraging the fact multiple measurements can be obtained at each training location. Regardless, the work in this paper presents a clear solution to the problem of computationally-constrained verification of complex, deterministic nonlinear systems subject to structured uncertainties.

REFERENCES

[1] Department of Defense MIL-HDBK-1797, “Flying qualities of piloted aircraft.”
[2] Federal Aviation Administration FAR Part 25, “Airworthiness standards: Transport category airplanes.”
[3] E. Lavretsky and K. A. Wise, Robust and Adaptive Control. Springer, 2013.
[4] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction. The MIT Press, 1998.
[5] J. Moore and R. Tedrake, “Control synthesis and verification for a perching UAV using LQR-trees,” in IEEE Conference on Decision and Control, 2012.
[6] S. Prajna, Optimization-Based Methods for Nonlinear and Hybrid Systems Verification. PhD thesis, California Institute of Technology, 2005.
[7] J. Kapinski, J. Deshmukh, S. Sankaranarayanan, and N. Arachiga, “Simulation-guided Lyapunov analysis for hybrid dynamical systems,” in Hybrid Systems: Computation and Control, 2014.
[8] U. Topcu, Quantitative Local Analysis of Nonlinear Systems. PhD thesis, University of California, Berkeley, 2008.
[9] P. Reist, P. V. Peeriswerk, and R. Tedrake, “Feedback-motion-planning with simulation-based LQR-trees.” Under review, 2015.
[10] J. F. Quindlen, U. Topcu, G. Chowdhary, and J. P. How, “Region-of-Convergence Estimation for Learning-Based Adaptive Controllers,” in American Control Conference, 2016.
[11] A. Kozarev, J. F. Quindlen, J. P. How, and U. Topcu, “Case Studies in Data-Driven Verification of Dynamical Systems,” in Hybrid Systems: Computation and Control, 2016.
[12] B. Settles, Active Learning. Morgan and Claypool, 2012.
[13] O. Maler and D. Nickovic, Monitoring Temporal Properties of Continuous Signals, pp. 152–166. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004.
[14] S. Sadreddini and C. Bela, “Robust temporal logic model predictive control,” in Allerton Conference on Communication, Control, and Computing, 2015.
[15] D. Sadigh and A. Kapoor, “Safe control under uncertainty with probabilistic signal temporal logic,” in Robotics: Science and Systems Conference, June 2016.
[16] C. Rasmussen and C. Williams, Gaussian Processes for Machine Learning. MIT Press, Cambridge, MA, 2006.
[17] J. Kremser, K. S. Pedersen, and C. Igel, “Active learning with support vector machines,” Data Mining and Knowledge Discovery, vol. 4, pp. 313–326, July 2014.
[18] Y. Zhang, T. N. Hoang, K. H. Low, and M. Kankanahalli, “Near-optimal active learning of multi-output gaussian processes,” in AAAI Conference on Artificial Intelligence, 2016.
[19] J. F. Quindlen and J. P. How, “Machine Learning for Efficient Sampling-based Algorithms in Robust Multi-Agent Planning under Uncertainty,” in AIAA SciTech Conference, 2017.
[20] G. Chen, Z. Sabato, and Z. Kong, “Active learning based requirement mining for cyber-physical systems,” in IEEE Conference on Decision and Control, 2016.
[21] A. Gotovos, N. Casati, and G. Hitz, “Active learning for level set estimation,” in International Joint Conference on Artificial Intelligence, pp. 1344–1350, 2013.
[22] T. Desautels, A. Krause, and J. W. Burdick, “Parallelizing exploration-exploitation tradeoffs in gaussian process bandit optimization,” Journal of Machine Learning Research, vol. 15, pp. 4053–4103, 2014.
[23] R. Grande, G. Chowdhary, and J. P. How, “Nonparametric adaptive control using gaussian processes with online hyperparameter estimation,” in IEEE Conference on Decision and Control (CDC), IEEE, 2013.
[24] R. Y. Rubinstein and D. P. Kroese, Simulation and the Monte Carlo Method. Wiley, 2008.
[25] A. Kulesza and B. Taskar, “Determinantal point processes,” Foundations and Trends in Machine Learning, vol. 5, no. 2-3, pp. 123–286, 2012.
[26] A. Kulesza and B. Taskar, “k-dpps: Fixed-size determinantal point processes,” in International Conference on Machine Learning, 2011.
[27] G. Chowdhary and E. Johnson, “Concurrent learning for convergence in adaptive control without persistency of excitation,” in IEEE Conference on Decision and Control, IEEE, 2010.
[28] C. Elliott, G. Tallant, and P. 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), June 2015.
[29] V. C. Elliott, G. Tallant, and P. 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.

VII. APPENDIX

A. Batch Active Sampling using Approximate Entropy Reduction

Just as with the sequential algorithm in Alg. 1, the process starts with an initial model and training set $\mathcal{L}$, which is used to compute the entropy (Step 3). The estimated entropy is then initialized to this quantity. Steps 4-8 are the main difference between the sequential procedure and this batch process. Step 5 selects the next sample from the current estimate of the entropy, $\hat{H}(\theta | \mathcal{L} \cup \mathcal{S}, \alpha)$. In order to encourage diversity in the points of the batch, this point added to a temporary holding set, $\mathcal{S}$, and removed from $\mathcal{U}$. Set $\mathcal{S}$ is then used to update the covariance function $\Sigma(\theta)$ to incorporate the effects of the previously selected points in the batch. The main problem that arises is that the mean function cannot be updated in a similar fashion. The update of $\mu(\theta)$ would require the measurements $\forall \theta \in \mathcal{S}$, which have not been obtained at that time. Therefore, the mean is “held” at the training set $\mathcal{L}$, defined as $\mu(\theta)$, while the covariance is updated with the current $\mathcal{S}$, defined as $\Sigma(\mathcal{L} \cup \mathcal{S})(\theta)$. The updated probability of satisfaction is then approximated using the
The approximate probability of satisfaction can then be used to compute the estimated entropy $\hat{H}(\theta; L \cup S, \sigma)$ in the same manner $\hat{P}_+ (\theta; L \cup S, \sigma)$ was used to compute $H(\theta; L, \sigma)$ in Eq. 9 (Step 7). Once a complete batch of $M$ points has been selected, simulations are performed at those locations in $S$ and the corresponding measurements $y_S$ are taken (Step 9). Both $S$ and $y_S$ are added to the training dataset $L$ and used to retrain the GP model. Once this has been completed, set $S$ is emptied and the process is repeated until $T$ iterations are complete.

Algorithm 4 Batch active sampling using approximate entropy reduction
1: **Input:** training set $L$, available sample locations $U$, trained regression model $\mathcal{GP}$, $T$ batches, $M$ points in each batch, empty set $S$
2: **for** $i = 1 : T$ **do**
3:     Compute entropy $H(\theta; L, \sigma) \forall \theta \in U$, set $\hat{H}(\theta; L \cup S, \sigma) = H(\theta; L, \sigma)$
4: **for** $j = 1 : M$ **do**
5:     Select $\theta = \arg\min_{\theta \in U} \hat{H}(\theta; L \cup S, \sigma)$
6:     $S \leftarrow S \cup \theta$ and $U \leftarrow U \setminus \theta$
7:     Update $\Sigma(\theta)$ with $\theta \in S$ (but not $\mu(\theta)$) and recompute $\hat{H}(\theta; L \cup S, \sigma)$
8: **end for**
9: Run simulations $\forall \theta \in S$, obtain $y_S$
10: $L \leftarrow L \cup \{S, y_S\}$
11: Reinitialize empty $S$, retrain $\mathcal{GP}$ with new $L$
12: **end for**

B. k-Determinantal Point Processes for Sampling

Determinantal Point Processes (DPPs) are useful tools for selecting sets of samples where diversity in the samples is important [25]. 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 are desired; however, the DPP loses its utility when it is constructed from a small number of initial samples. For these problems, k-DPPs [26] were developed to obtain a small set of samples (of size $k$) from a DPP with a larger initial set of samples. The following algorithm describes k-DPP based sampling as it relates to the simulation-based verification procedure.

The procedure assumes $M_T$ samples of $\theta$ have been generated according to $\mathcal{P}_H(\theta)$ formed from the entropy $H(\theta; L, \sigma)$. In order to have a suitable number of samples to construct the DPP, $M_T \approx 1000$ for the examples in this paper. 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 $L(i, j) \leq 1$ 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 $\lambda_j$ 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 compute the marginal and sampling from marginal in Step 10, adding indices to set $J$. Once the loop has run out of remaining indices, a sample $y_i$ from the set of all indices is 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 $V$ correspond to indices of $\theta$ terms in the initial input set of $M_T$ $\theta$ values sampled from $\mathcal{P}_H(\theta)$. The actual sample locations are taken from that set of $M_T$ points.

Algorithm 5 k-DPP sampling algorithm; adapted from [26].
1: **Input:** $M_T$ randomly generated samples of $\theta$, empty set $J$, $M$ points in batch
2: **for** $i = 1 : M_T$ **do**
3:     for $j = 1 : M_T$ **do**
4:         Compute $L(i, j) = e^{-||\theta_i - \theta_j||^2/l^2}$
5:     **end for**
6: **end for**
7: Eigendecomposition of $L \rightarrow \{v_j, \lambda_j\}$
8: Initialize $m = M$
9: for $j = M_T : -1 : 1$ **do**
10:     if $u \sim \text{Uniform}[0, 1] < \lambda_j/e_m$ 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 $\mathcal{P}(y_i) = \frac{1}{|V|} \sum_{v \in V} (v^T e_i)^2$
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: **Output:** sample set $Y$ of size $M$