Learning-based Bayesian Inference for Testing of Autonomous Systems

Anjali Parashar[1], Ji Yin[2], Charles Dawson[1], Panagiotis Tsiotras[2], and Chuchu Fan[1]

***Abstract*— For safe operation of robotic systems, it is important to understand its failure modes accurately using prior testing. Hardware testing of robotic infrastructure is known to be extremely slow and costly. Instead, failure prediction in simulation can help in analyzing the system before deployment. Conventionally, large-scale naïve Monte Carlo simulations are used for testing; however, this method is only suitable for testing average system performance. For safety-critical systems, worst-case performance is more crucial as failures are often rare events, and the size of test batches increases substantially to discover rare occurrences of failures. Rare-event sampling methods can be helpful, however, they exhibit slow convergence and cannot handle constraints. This research introduces a novel sampling-based testing framework for autonomous systems which bridges these gaps by utilizing a discretized gradient-based second-order Langevin algorithm combined with learning-based techniques for accurate prior estimation. Our method can predict more diverse failures by exploring search space efficiently and ensures feasibility with respect to temporal and implicit constraints. We demonstrate the application of our testing methodology on two categories of testing problems via simulations and hardware experiments. Our method discovers upto 2X failures compared to naïve Random Walk sampling, using only 0.5X samples.**

I. Introduction

Testing, verification and model validation are integral to ensuring the safety of robotic systems. Discovering possible failure modes of the system through testing is challenging for several reasons. First, the underlying mathematical problem of failure discovery in a realistic setting, is highly non-convex with an unpredictable loss landscape. Second, the sufficient exploration of the search space is important, as there are multiple failure modes of varying probability of occurence. To ensure proper coverage of the state space, state-of-the-art methods for testing, such as Random Walk MCMC sampling [1], need a significant amount of computational resouces, and can be extremely time-consuming [2].

In the past few years, some new favorable alternatives have emerged. Primarily, there are two ways to approach the problem of failure discovery, given a dynamic system and a known environment. The first set of methods relies on adversarial optimization [3], where access to gradient information allows discovery of failure modes. One key caveat of adversarial optimization is that it typically renders one failure case per search. Additionally, while implementing gradient-based methods for non-convex optimization, initialization plays a key role [4]. Therefore, inappropriately chosen initial conditions are likely to converge to a few high probability failure modes, thereby completely missing out on some rare failures that are equally detrimental to the safety of the system. Some works have proposed heuristic solutions akin to domain randomization to mitigate the initial condition bias [5], however, they suffer from sample inefficiency.

The second approach utilizes recent advancements in machine learning, and attempts to reconstruct the complex landscape of failure occurrence by using failure data on input [6]. These methods employ generative modelling tools to learn failure representations. A key challenge with this approach is the lack of generalization guarantees of neural network-based models for out of distribution data. In practice, there is no way to guarantee that the training dataset is sufficiently rich to cover all kinds of possible failures. Therefore, the generated instances of failure using such models are implicitly biased towards replicating the failures observed in the training dataset. Moreover, generation of failure data is a challenging problem in itself, since failures are rare-events that occur with very low probability [7].

Some works have recently analyzed failure discovery and model validation from the lens of Bayesian inference [7], [8], which combines the convergence properties of adversarial optimization and search-space exploration capabilities of sampling based methods. While these methods do not suffer from initialization limitations of determinisitic optimization [9], they typically carryover some of the drawbacks of sampling methods, such as uneven search space exploration and the curse of dimensionality. Generally, Bayesian inference techniques conduct an extensive search over a large prior distribution which becomes even more challenging for high-dimensional explorations and in addition, they suffer from slow convergence.

In this paper, we study the question — what is the most efficient way of testing a known dynamic system with known environmental interactions? The objective of this work is to discover failure modes of a given dynamic system with known controller, where we focus on failures caused due to environmental interactions (Fig. 1). We exploit the awareness of testing scenario to discover failure modes in simulation, with an aim to reduce the dependency on field testing as much as possible. Building upon the fundamental structure of Bayesian inference, we provide a modular approach to testing via learning-based stochastic optimization.

A. Key Contributions

This paper introduces a novel testing framework for control and planning algorithms for robotic systems by formulat-

[1] Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, USA. Email: {anjalip,cbd,chuchu}@mit.edu

[3] D. Guggenheim School of Aerospace Engineering, Georgia Institute of Technology, USA. Email:{jyin81,tsiotras}@gatech.edu

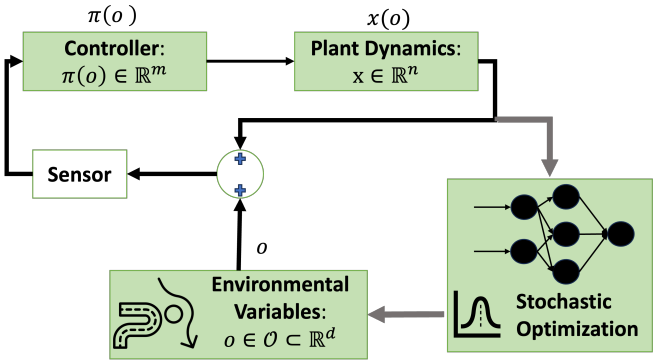


Fig. 1: Approach Overview

ing testing as a Bayesian inference problem (Section-II). One of the main objectives of this study is to extract meaningful information about the system without necessitating a heavy expenditure on the testing budget. A key contribution of this work is the proposal of learning-based methods for reducing the volume and dimensionality of prior distributions, promoting better performance for Bayesian inference techniques. Additionally, our framework can efficiently handle constrained stochastic optimization (Section-IV).

To address the convergence gap of sampling-based techniques in Bayesian inference, we exploit system awareness and leverage end-to-end differentiability [5], [8] for gradient-based sampling. We make use of the Underdamped Langevin Algorithm [10] (also known as 2^{nd}-order LA), which provides accelerated convergence guarantees and better coverage of search space, leading to discovery of diverse failure modes. In Section-III-A we provide a theoretical intuition for the same.

In Section-V we present three case studies of falsification of dynamic systems with different control algorithms to understand the key limitations of autonomous systems that lead to failures in the first place. Finally, we validate the observed failure modes through experimental demonstrations, analyzing the sim-to-real transfer.

II. Problem Statement

We consider continuous time closed-loop dynamic systems of the form $\dot{x} = f(x, \pi(x, o))$ for given state $x \in \mathbb{R}^n$, where $a = \pi(x, o) \in \mathbb{R}^m$ is a known controller which outputs actions $a \in \mathcal{A} \subseteq \mathbb{R}^m$ based on system state and environmental observations $o \in \mathcal{O} \subset \mathbb{R}^d$ in a fully observable setting. In the rest of the paper, we refer to o as Environment Variables (EVs). The testing architecture takes a user-defined property as an input such as "obstacle avoidance" for which we aim to find compelling failure cases that reveal loopholes in the controller performance and underlying limitations of our dynamic system.

For a given user-defined property, we design a cost function $c(o, X)$ where $X = (x_i)_{i=1}^T$ such that a failure of the underlying dynamic system in the presence of a chosen controller can be defined as the corresponding cost function exceeding a threshold c^*. That is, $c(o, X) > c^*$ corresponds to the violation of the desired property. The testing problem can then be mathematically formulated as:

$$\text{find} \;\; \mathcal{O}_{\text{fail}} = \{o | c(o, X) > c^*\}. \tag{1}$$

III. Approach: Optimization via Sampling

We employ Bayesian inference to solve (1), as it can discover more than one failure per search. This is done by constructing a Bayesian inference problem that is equivalent to (1) using o as a decision variable. This can be understood as sampling from the conditional distribution of o that triggers failures, that is,

$$o \sim p(o | c(o, X) > c^*) \tag{2}$$

Failures are low probability events, consequently, regions representing $p(o|c(o, X))$ are extremely low density regions. To facilitate efficient sampling from these regions, we construct a posterior distribution $p(c(o, X) > c^*|o)$ that is highly biased towards failures, given by:

$$p(c(o, X) > c^*|o) \propto \exp(-\gamma[c^* - c]_+)$$

Here $[\cdot]_+$ represents the ReLU operator and assigns equal cost to every failure mode with $c > c^*$. This construction ensures that the target density from which we wish to sample, p^* indeed is a solution to (1) and contains all failure modes regardless of their severity [8]. Using the Bayes rule, the inference problem can therefore be written as:

$$p(o|c(o, X) > c^*) \propto p(o) \cdot \exp(-\gamma[c^* - c]_+) \tag{3}$$

Here, $p(o)$ represents the prior belief. In this work, we utilize spatio-temporal constraints imposed on o to inform the design of prior using machine-learning based methods, discussed in Section-IV. This ensures that the constraints are always satisfied. In the next subsection, we provide insights on how to sample from distribution $p \propto \exp(-\gamma[c^* - c]_+)$ using 2^{nd}-order Langevin algorithm (LA).

A. Stochastic Optimization via Underdamped Langevin

Recently, reference [8] demonstrated that for end-to-end differentiable systems, gradient-based Langevin Monte Carlo (also known as 1^{st}-order LA) is a helpful tool for solving problems that can be written as in (3). Langevin dynamics (LD) based sampling tools have emerged as a promising alternative to gradient-free MCMC algorithms such as Random walk MCMC, which are popularly used in verification and falsification tools for stochastic optimization [1]. Consider the problem of sampling from a target distribution p^* of the form $p^*(o) \propto \exp(-U(o^*))$, where $o^* = \operatorname{argmin} U(o)$ for a differentiable function $U(o)$. 1^{st}-order LA approaches this problem by constructing gradient-based iterates corresponding to a first-order Stochastic Differential Equation (SDE) given by:

$$do_t = -\gamma \nabla U(o_t) dt + \sqrt{2\gamma} dB_t \tag{4}$$

Here dB_t denotes the time derivative of standard Brownian motion and γ corresponds to the learning rate.

Locating all optimal solutions in a non-convex optimization setting is the optimization equivalent to sampling from a multimodal distribution. Therefore, in multimodal distributions, a sufficient amount of search space exploration is extremely important to discover all possible solutions, in addition to the speed of convergence. In [11] it was shown that accelerated-gradient based methods such as Nesterov's method [12] are able to locate the optimal solution with faster speed of convergence in non-convex optimization, compared to first-order gradient-based methods. This is primarily due to the added momentum term in accelerated methods, which leads to faster exploration of the global optimum. We hypothesize that Underdamped Langevin Algorithm (also known as 2^{nd}-order LA) [10] performs better than 1^{st}-order LA and Random Walk MCMC in discovering diverse failure modes and converging faster. The second-order SDE corresponding to 2^{nd}-order LA can be written as:

$$\begin{aligned} do_t &= -r_t dt \\ dr_t &= -\nabla U(o_t)dt - \gamma r_t dt + \sqrt{2\gamma} dB_t \end{aligned} \tag{5}$$

Note that the momentum variable r_t is also present in accelerated gradient descent and it helps in avoiding the issue of getting stuck in local minima, which is commonly seen in first-order methods. We hypothesize that 2^{nd}-order LA can provide better search space exploration for a similar reason, and identify diverse failure modes. Recently, it was shown that non-reversible SDEs such as 2^{nd}-order LA search for global optimum at a much faster timescale [13]. Additionally, [10] shows that 2^{nd}-order LA converges to ϵ accuracy of the global minimum in $\sqrt{(d/\epsilon)}$ steps, (as opposed to d/ϵ steps needed for 1^{st}-order LA, [14]), under certain assumptions. This shows that for high-dimensional non-convex optimization, 2^{nd}-order LA is a promising approach, ensuring speed, sufficient search space exploration and scalability. Furthermore, [8, Theorem 5.1] shows that using an appropriate cost design $c(\cdot)$, the problem of discovering failure modes written as (3) meets these assumptions. Therefore, we can utilize the convergence guarantees available for 2^{nd}-order LA in our framework.

For this work, we adopt the discretization of Underdamped Langevin proposed in [10], where each step can be defined as sampling from a normal distribution with mean $\mu(o)$ and covariance Σ introduced in [10, Algorithm-1], namely,

$$o_j \sim \mathcal{N}(\mu(o_{j-1}), \Sigma) \tag{6}$$

IV. Prior design

In practice, standard prior distributions such as normal, uniform distribution, etc. are used for sampling within a given region. However, for testing purposes, the region of failures are low-density distributions and naïvely chosen priors can severely over-estimate the likelihood of failure which impacts the efficiency of sampling, as a lot more steps are needed to converge to the true distribution. By having stronger prior estimates, we can improve the efficiency of sampling significantly. We remodel the prior distribution as constraints on the EVs, and create learnt-representations of EVs using available physical information about the environmental variable o and its interaction with the dynamic system. Using this method, we can simultaneously construct a strong prior and satisfy hard constraints imposed on the EVs.

Prior distribution design can be classified by the nature of the environment variables under consideration. Specifically, we consider two cases:

(i) *Static EVs*: ($o \in \mathbb{R}^{d_1 \times m_1}$) These variables remain static throughout the evolution of the dynamic system. Here, m_1 defines the dimension of our search space and d_1 corresponds to the number of identical kinds of EVs. In this study, we consider the problem of locating d_1 circular obstacles of fixed radius as an example.

(ii) *Sequence EVs*: ($o = (o_t)_{t=1}^{n_1}, o_t \in \mathbb{R}^{d_1 \times m_1}$) These variables have a sequential structure, such that at a given instance of time, only a few elements of the sequence interact with the dynamic system. Examples include reference paths and non-ego vehicles. Here n_1 denotes the length of sequence.

This distinction is motivated by the fact that Static EVs are low dimensional but have complex, implicit spatial constraints but do not change their position. On the other hand, Sequence EVs have constraints of temporal consistency, and are high dimensional. Therefore, different approaches suit their prior construction. In this paper, we focus on *independent* EVs, that do not take the ego-agent's behavior into consideration, such as a reference path or obstacles, that do not change based on vehicle's current location. An example of *dependent* EVs in this context would be non-ego vehicles in a traffic scenario, which also simultaneously adjust their manuevers based on the ego-vehicle's location.

The spatio-temporal behavior (i.e., location, evolution with time, etc) of *independent* EVs can be predicted with known confidence, hence, data-driven tools are a highly attractive means for learning the prior representation of these variables. We use neural network-based architectures for learning accurate priors and further refine the process of drawing samples from these areas. The design approach for both cases is discussed in detail in the next sections.

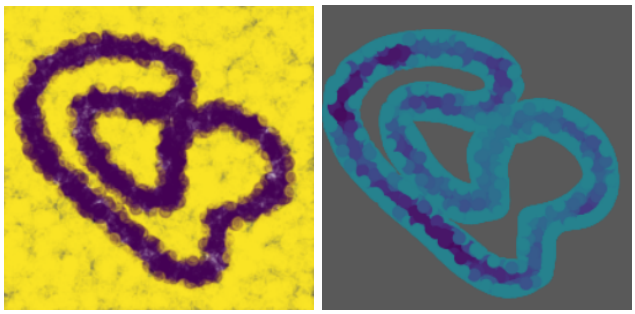
A. Neural Projection for Static Environment Variables

For a fixed number of obstacles, consider the problem of finding obstacle locations on a racetrack for autonomous driving that are highly likely to collide with the vehicle. Here, the environment variable $o \in \mathbb{R}^{d_1 \times 2}$ represents d_1 obstacles scattered in a 2D environment around a pre-designed reference path.

Regions far away from the reference path have an extremely low probability of collision, hence, the right choice of prior would include a region in the vicinity of the reference track. Based on this hypothesis, for a given reference track,

(a) (b)

Fig. 2: Neural Projection for AutoRally racetrack. Fig. 2a shows the AR racetrack. Fig. 2b shows the decision boundary learned by g_ϕ



(a) (b)

Fig. 3: Fig. 3a shows classification of uniformly sampled 2D datapoints across the 2D box classified by g_ϕ, Fig. 3b shows the points projected on Ω after applying Algorithm-1

we construct a region Ω defined by inner and outer track boundaries of a chosen width w. It is hard to express the region Ω as an explicit function always, since a parametric expression of the racetrack is not always available. We first construct a box $\mathcal{D}$ circumscribing the region Ω such that $\Omega \subseteq \mathcal{D}$ and sample 2D datapoints uniformly from $\mathcal{D}$. We then train a DNN based binary classifier $g_\phi(x, y)$ to learn the likelihood of a previously unseen datapoint (x, y) being within Ω or in $\mathcal{D}\backslash\Omega$. Fig. 2b shows an example of the learned decision boundary for the AutoRally racetrack [15] using $g_\phi(x, y)$.

To refine our Bayesian inference using this information, for any point within $\mathcal{D}$, we propose a Neural Projection Operator $\boldsymbol{P} : \mathcal{D} \to \Omega$ as follows:

$$\boldsymbol{P}[o] = o - \nabla g_\phi(o) g_\phi(o) / \|\nabla g_\phi(o)\|^2 \tag{7}$$

This is a Newton's method-based projection operator, inspired by the Neural Projection technique presented in [16]. $\boldsymbol{P}[o]$ ensures that any point lying outside Ω is mapped to the nearest point within Ω while a point lying within Ω remains there. Fig. 3 shows the implementation of the Neural projection for projecting samples within the AutoRally track region Ω, given uniformly generated samples in a 2D box $\mathcal{D}$.

Finally, for sampling static environment variables, we propose Algorithm-1, which combines 2^{nd}-order LA with Neural Projection operator at each iteration to strictly sample from within Ω. The momentum step of 2^{nd}-order Langevin can sometimes lead to exploration of low-likelihood regions outside Ω which is of no interest from testing perspective. The Neural projection operator also acts as a filter and prevents exploration outside Ω.

Algorithm 1 Neural Projection with Langevin

Initial conditions $o_0 \sim \mathcal{U}[\mathcal{D}]$
for $k = 1$ to N **do**
  $\tilde{o}^0 \sim \mathcal{N}(\mu(o_{k-1}), \sum)$
  for $j = 1$ to M **do**
    $\tilde{o}^j \leftarrow \boldsymbol{P}[\tilde{o}^{j-1}]$
  end for
  $o_{k+1} = o^M$
end for

B. Latent Variable Encoding for Sequence Environment Variables

As a motivating example, given a reference-tracking controller, we consider the problem of finding reference paths that are hard for a dynamic system to follow. For this task, the environment variable under consideration is a reference path $o \in \mathbb{R}^{N\times 2}$ of length N, where $o_k = [x_k, y_k]^\mathsf{T}$ for $k = 0, \ldots, N-1$.

1) Challenges with optimization for sequence EVs: Firstly, the decision variable o is high dimensional and finding falsifying examples in a high dimensional space can be quite inefficient and slow. Secondarily, our search space is constrained to be a set of "realisitic"trajectories which should have temporal consistency. This constraint is hard to realize in practice, since it cannot be expressed explicitly.

2) Approach: To tackle the second problem of temporal consistency, we use evolution of a dynamic system to generate o. Consider a control affine dynamic system with open loop control $\tilde{u} \in \mathbb{R}^m$ and continuous functions $f(\cdot) : \mathbb{R}^p \to \mathbb{R}^p$ and $g(\cdot) : \mathbb{R}^m \to \mathbb{R}^p$. We generate every element of the sequence o_k using the discretization of the open-loop dynamic system $\dot{\tilde{o}} = f(\tilde{o}) + g(\tilde{u})$ such that $o_k = \tilde{o}_{k[0:2]}$. Therefore, we have

$$\tilde{o}_k = \tilde{o}_{k-1} + \Delta t(f(\tilde{o}_{k-1}) + g(\tilde{u}_{k-1})), \quad \text{for} \quad k = 0, \ldots, N-1.$$

For an appropriate choice of $f(\tilde{o})$, $g(\tilde{u})$ and Δt, we can generate a wide variety of sequences. For the reference path tracking examples considered in Section-V-B, we choose $f(\tilde{o})$ and $g(\tilde{u})$ corresponding to bicycle dynamic model.

To resolve the first challenge of dimension reduction, we use Variational Autoencoders (VAE) to learn a lower dimension time-invariant representation of a continous control input $\tilde{u}(t)$ such that $\tilde{u}_k = \tilde{u}(k\Delta t)$ defined in the range $t \in [0, N\Delta t]$. To learn encodings corresponding to a diverse range of sequence EVs, we need to train the model on a rich dataset that consists of a wide range of EVs and learn an appropriate time-invariant representation. To generate the appropriate training dataset, we formulate the control inputs

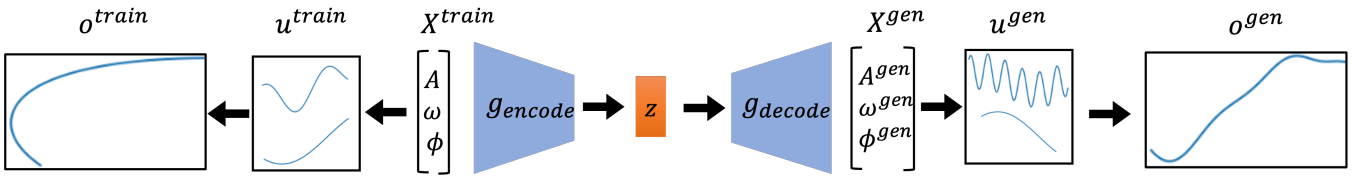


Fig. 4: Encoding latent variables for generating sequential environment variables 'o' using VAE presented in Section-IV-B

at every timestep t as a finite sum of n_1 sinusoidal component functions $h(A_j, \omega_j, \phi_j)$ where A_j, ω_j, ϕ_j represent the amplitude, frequency, and phase lag corresponding to each sinusoidal function. Mathematically, for a given $t \in [0, N\Delta t]$, $u(t)$ can be expressed as:

$$u(t) = \sum_{j=1}^{n_1} A_j \sin(\omega_j t + \phi_j). \tag{8}$$

We generate the training data by uniformly sampling across a variety of $\{A_j, \omega_j, \phi_j\}_{j=1}^{n_1}$ for a fixed sum length n_1 and chosen open-loop dynamic system. This ensures that the generated trajectories have temporal consistency and diversity, and do not correspond to a restrictive family of parametric curves such as circles, ellipsoids, etc. An inspiration behind this is the reference path design discussed in Section-V-B, where we aim to test across realistic reference paths which consist of a combination of curves with varying radii of curvature and lengths.

With $X = (A_j, \omega_j, \phi_j)_{j=1}^{n_1} \in \mathbb{R}^{n_1 \times 3}$ as the input and output of the Autoencoder, we train a DNN-based VAE, where the encoder network $g_{\text{enc}} : \mathbb{R}^{n_1 \times 3} \to \mathbb{R}^{d_2}$ learns a latent representation of the input space such that $d_2 \leq 3n_1$. The network is trained to minimize the KL-divergence between the ground truth sequences and sequences generated by the control inputs rendered by the decoder network $g_{\text{dec}} : \mathbb{R}^{d_2} \to \mathbb{R}^{n_1 \times 3}$. The training pipeline is summarized in Fig. 4.

Therefore, we compress the problem of finding N dimensional sequence to an equivalent problem of finding a d-dimensional latent variable $z \in \mathbb{R}^{d_2}$ such that $d \leq N$. After reducing the dimension of our search space, we use Langevin-based optimization to find the set of latent variables $\mathcal{Z}_{\text{fail}} = \{z | \bar{c}(g_{\text{dec}}(z)) > c^*\}$, where $\bar{c}$ measures the cost $c(o, X)$ generated by this method.

V. Simulation & Experiments

As an application of the technique discussed in this paper, we present case-study of testing using a sequence environment variables in Section-V-B and two case studies of falsification using static environment variable in Section-V-C and V-D. We also perform experimental validation for two selected case-studies, discussed in Section-V-E.

A. Baselines & Metrics

Because of the modularity of our approach, it is possible to use the EV design architecture (Section-IV) to improve the performance of any suitable stochastic optimization method. To demonstrate the same, in addition to 2nd-order LA, we present an implementation of our framework on 1st-order LA [14] and gradient-free Random sampling (Brute force method). We test for the effect of acceleration, speed of convergence and coverage by comparing the failure discovery rate, mean and max cost value and dispersion metric C_{disp} respectively, across all baselines.

For comparing the coverage of the search space, we introduce our own metric C_{disp} given in (9) below. To calculate C_{disp}, sampled failure modes are clustered using Gaussian Mixture Clustering, and the maximum Euclidean distance between the mean positions of all clusters across all dimensions is computed. Mathematically, C_{disp} can be written as:

$$C_{\text{disp}} = \max_{1 \leq k \leq d} \{ \max_{1 \leq i,j \leq M} \{ \|\mu_i^k - \mu_j^k\|_2^2 \} \}. \tag{9}$$

Here M denotes the number of cluster components, μ_i^k denotes the mean of the i^{th} cluster in the k^{th} dimension. A higher value of C_{disp} implies better coverage of search space.

B. Case-study 1: Falsification of reference path tracking using speed and steering LQR control

1) Problem Description: We consider the problem of tracking a given reference path using LQR controller, adopted from [17], for a bicycle dynamic model ($n = 4$) with steering angle and acceleration as control input ($u = [\delta, a]^\mathsf{T}$) and specified control limits ($\pm\delta^{\max}, \pm a^{\max}$). We define the performance metric as the average distance from the reference path o, with the cost function designed as:

$$c(o, X) = \sum_{i=1}^{N} \|\bar{x}_i - o_i\|_2^2. \tag{10}$$

Here $\bar{x} = x_{[0:2]}$. Failure is defined as $c > c^*$, with $c^* = 200$ chosen by trial and error. For this task, the environment variable is a reference path $o \in \mathbb{R}^{N \times 2}$ of length $N = 100$, where $o_k = [x_k, y_k]^\mathsf{T}$ for $k = 0, \ldots, N-1$. Hence, the testing objective is to find the set of reference paths for which LQR fails to generate a suitable tracking trajectory.

For generating falsifying racelines, we utilize the Sequence Environment variable design approach explained in Section IV-B. We chose the bicycle dynamic model with steering and throttle as inputs to define $f(\tilde{o})$ and $g(\tilde{u})$. Here, $\tilde{o} = [o_x, o_y, o_\theta, o_v]^\mathsf{T}$ and $o = \tilde{o}_{[0:2]}$.

2) Dataset Design: To train the VAE, we chose reference trajectories from a distribution that is very close to the actual failure distribution. This is done intentionally to observe the learnt distribution and generalization capability of the VAE. We achieved this by introducing several instances

of reference trajectories that require violation of control limits to track correctly. This is done by uniformly sampling around regions of amplitude $\{A_i\}_{i=1}^n$ higher than control limit magnitude i.e., for a chosen σ:

$$\{A_i\}_{i=1}^n \sim \mathcal{U}[\delta^{\max} - 3\sigma, a^{\max} - 3\sigma] \times [\delta^{\max} + \sigma, a^{\max} + \sigma]. \tag{11}$$

By this method, we learn the **Nominal** distribution of our four dimensional latent space ($\mathcal{Z} \subseteq \mathbb{R}^4$) centered around $\mu_z = [0.8, 0.9, 0.95, 9]^\mathsf{T}$ shown in Fig. 6 in blue.

3) Observations: We apply the 2nd-order LA to generate a failure distribution by sampling from the latent space $\mathcal{Z}$. Eventually, we notice that our method discovers a failure region far from the learnt (nominal) distribution (**Region-1,** $\mu_z = [0.06, 0.03, 0.03, 0]^\mathsf{T}$, highlighted in red, Fig. 6) in addition to a low variance distribution near the nominal distribution (**Region-2**, $\mu_z = [0.99, 0.93, 0.89, 0.99]^\mathsf{T}$, Fig. 6). Additionally, we notice that there is significant difference in the failure modes corresponding to **Region-1** and **Region-2**, as the former primarily represents paths where radius of curvature is smaller than the size of the vehicle, hence are hard to track. The latter represents paths that cannot be tracked due to control limits, as expected from the training dataset's bias towards this failure mode. Fig. 5 shows examples of different kinds of reference path generated by our method, and corresponding trajectories generated by LQR.

We compare between three baseline algorithms (10 chains, 100 epochs each), namely, Brute force method, 1st and 2nd-order LA using the Sequence EV design framework. The results have been summarized in Table-I. We observe that 2nd-order LA outperforms in discovering higher frequency of failures as well as covering larger search-space. It must be noted however, that by reducing the sampling size and volume, we are able to leverage all methods for good performance, which is reflected in the marginal difference in failure rates and mean cost between Brute force method and 1st-order LA.

This application demonstrates the generalization capability of our method in discovering new kinds of failure modes that are not captured by the training dataset. Such a technique can be used in generalized failure discovery in scenarios where avialable information is limited to certain kinds of failures.

TABLE I: LQR Speed+Steering Control (Bicycle Model) Baseline comparison

	Failure rate	Mean cost	Max cost	C_{disp}
Random Walk MH	0.53	0.45	1.27	0.91
1st-order LA	0.43	0.36	1.27	1.43
2nd-order LA	**0.75**	**0.51**	**1.52**	**1.48**

C. Case-study 2: Falsification of Obstacle avoidance via SQP

1) Problem Description: In this example, we use a JAX library called trajax [18] to solve a reference tracking problem with obstacle avoidance on a semicircular track with obstacle locations of fixed radius as the decision variable. Here, the environment variable $o \in \mathbb{R}^{d_1 \times 2}$ represents d_1 obstacles scattered in a 2D environment within the semi-circular track for a unicycle dynamic model with speed and angular velocity as control inputs $u = [v, \omega]^\mathsf{T}$. The inner and outer track boundaries are used for constructing a region Ω such that $o \in \Omega$, i.e, the prior $p(o)$ is well defined by the constraint of being within the boundaries, but is not explicitly known. We use the neural projection technique presented in Section IV-A to learn g_ϕ that outputs the likelihood of a point being in or out of the track.

In the initial simulations we observe that the collision rate with obstacles is very low, which can be explained intuitively as the solver always finds the most optimal path from a set of feasible paths, hence the state constraints are satisfied.

Therefore, we aim to find obstacles that cause a sig-

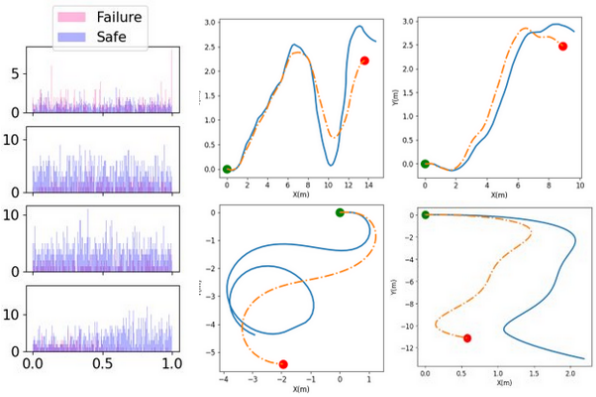


Fig. 5: Encoding distribution generated using Brute force method to reflect the high proportion of non-failure encodings (Left). Only 20% of the samples correspond to failure cases. Our method generates 1.9X failures using 0.5X samples with 75% failure discovery rate (Table-I). Examples of reference trajectory tracking failure case generated using our method (Right)

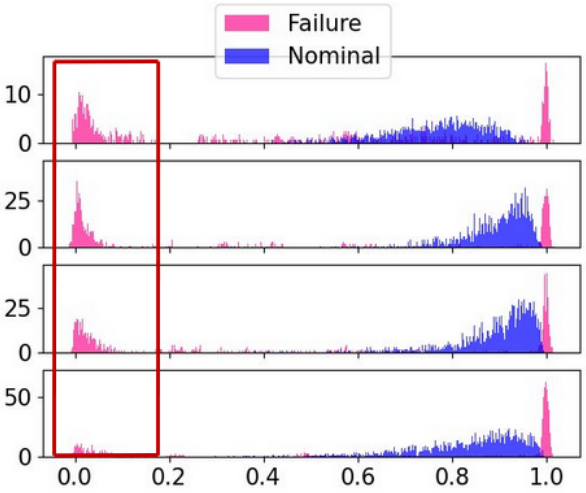


Fig. 6: VAE Latent space visualization for LQR on Bicycle Model example. The **Nominal** distribution is the latent distribution learnt by the VAE on the training dataset. The **Failure** distribution corresponds to that discovered using our approach.

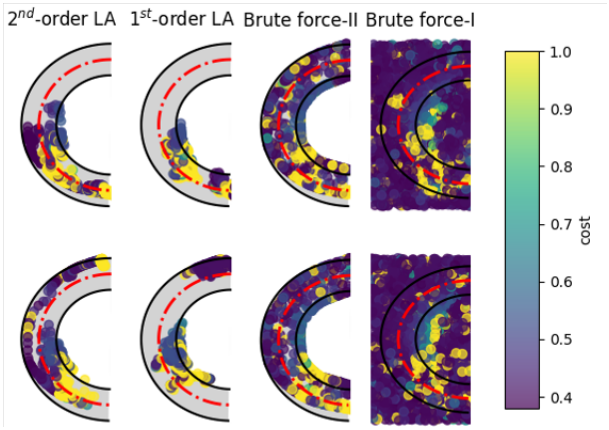


Fig. 7: Comparison of samples generated by 2nd-order LA, 1st-order LA, Random walk with Metropolis Hastings and Brute force (left to right). Obstacle-1 (Obstacle-2) positions are shown at the top (bottom). 2nd-order LA provides more spatial coverage of search space than the baselines. The colorbar shows the cost where failure corresponds to samples with cost ≤ 0.3. Our algorithm discovers more than 2X failures with 0.3X the sample size compared to Brute force method (w/o projection)

nificant deviation from the reference trajectory, and find instances where the solver favors feasibility over optimality. Algorithm-1 is applied for testing against the performance metric of the distance from the reference trajectory, with the cost function as in (10).

2) Observations: We conduct simulations for the case $d_1 = 2$, that is, scattering two obstacles in the environment and compare between different methods across previously discussed metrics. In addition to comparison between 1st and 2nd-order LA, we also compare with two variants of the Brute force method, namely, Brute-force-I and Brute-force-II. Brute-force-I is a naïve implementation of random sampling from a two-dimensional box $\mathcal{D}$ circumscribing Ω. Brute-force-II implements random sampling with neural projection, by applying the neural projection $\boldsymbol{P}[o]$ (7) to each $o \sim \mathcal{U}(\mathcal{D})$. We implement three chains with 200 epochs each for Brute-force-II, 1st and 2nd-order LA and compare it against 2,200 randomly generated samples by Brute-force-I method. The results are summarized in Table-II

Fig. 7 shows a distribution of obstacles generated by different algorithms. 2nd-order LA is able to move across low-probability regions to discover newer failure locations, while 1st-order LA often gets stuck in local optimas. Table-II shows that 2nd-order LA has higher dispersion and failure discovery rate compared to Brute-force-II, despite its even coverage of search space. Table-II also highlights the improvement achieved by neural projection, as Brute-force-II discovers more failures with significantly smaller sample size (0.3X).

We also note that the initial condition x_0 also impacts the feasibility of solution, and have not been considered in this study, as we assume fixed initial conditions, but can be very easily accomodated in our testing framework by augmenting environment variable with x_0 and creating a new environment variable o_{new}, such that

$$o_{\text{new}} = [o|x_0]^{\mathsf{T}} \tag{12}$$

TABLE II: SQP Baseline comparison (Optimization)

	Failure rate	Mean cost	Max cost	C_{disp}
Brute force-I	0.17	0.50	1.0	28.68
Brute force-II	0.25	0.58	1.0	27.13
1st-order LA	0.40	0.67	1.0	26.76
2nd-order LA	**0.44**	**0.69**	**1.0**	**30.69**

D. Case-study 3: Falsification of Obstacle Avoidance for Trajectory Tracking via MPPI

1) Problem description: We use MPPI for autonomous racing on the AutoRally racing platform presented in [15] with a single-track bicycle dynamic model ($n = 7, m = 2$). Unlike previously discussed examples, MPPI is a stochastic control algorithm which requires randomly sampled trajectories to synthesize an optimal control policy. The environment consists of a bounded reference trajectory and sets of d_1-obstacles scattered in the 2D space. The cost function of MPPI is fine-tuned to enable obstacle avoidance in addition to reference trajectory tracking. Here, we construct a testing scenarios of scattering 2 sets ($d_1 = 2$) of circular 2D obstacles with fixed radius ($r = 1.5$) and implement Algorithm-1 to find falsifying locations of obstacles. Failure is defined as a collision and the corresponding cost function $c(\cdot)$ is:

$$c(o, X) = \frac{1}{k} \log \sum_{i=1}^{d_1} \sum_{j=1}^{T} \exp\left(k \|\bar{x}_t - o_i\|_2^2\right) \tag{13}$$

Here T is the length of simulation, k is a constant, set to $k = 1,000$. The chosen cost function is a smooth approximation of maximum. Therefore, the cost function will try to minimize the maximum distance of either obstacle from the trajectory. This is done to ensure that both obstacles are in close proximity of the trajectory and can effectively contribute towards a possible collision.

2) Observations: Our method identifies several local minima on the racetrack. Additionally, we observe that collision of an obstacle depends on local state behavior, i.e., the speed, the angular velocity and the position at the time of collision and a few steps before that, and is relatively unaffected by the state of the vehicle long before. Based on this hypothesis, we classify the discovered failure modes into two categories:

(i) *Type-1*: Each instance of this failure mode is a combination of two independent local minima, i.e, the vehicle has a high probability of collision with both obstacles individually. For such failure instances we can write:

$$p_{\text{fail}}(o_i|o_j) = p_{\text{fail}}(o_i) \;\text{ for } i = 1, 2,\; j = 2, 1. \tag{14}$$

(ii) *Type-2*: This failure mode consists of two obstacles such that the probability of collision of the vehicle is increased due to the presence of the other obstacle. Hence, the location of first obstacle informs the location

of the other obstacle and this combination creates new local minima. For such failure :

$$p_{\text{fail}}(o_2, o_1) > \max\{p_{\text{fail}}(o_2), p_{\text{fail}}(o_1)\}$$
$$p_{\text{fail}}(o_1|o_2) = p_{\text{fail}}(o_1). \quad (15)$$

Here, o_1 is the first obstacle encountered by the vehicle and o_2 is encountered after o_1. Note that we cannot necessarily say that $p_{\text{fail}}(o_2|o_1) > p_{\text{fail}}(o_2)$, as several *Type-2* collisions happen in regions between the position of **Obstacle-1** and **Obstacle-2**, i.e., new local minima are created.

All **Type-1** failure modes can be expressed as the union of independent failure modes of lower cardinality, which corresponds to $d_1 = 1$ in this case. An example of **Type-1** failure mode is shown in Fig. 8b. The shown location corresponds to $p_{\text{fail}}(o) = 0.94$. Fig. 9c shows an instance of **Type-2** failure mode. Here, $p_{\text{fail}}(o_2, o_1) = 0.81$, with $p_{\text{fail}}(o_1) = 0.34$, $p_{\text{fail}}(o_2) = 0.67$. Colliding (safe) trajectories shown in blue (orange). The likelihood of collision is calculated over 100 random trials.

To demonstrate the utility of our method, we perform Brute-force sampling without projection with 1,700 random samples to generate an approximate cost distribution of obstacles. Fig. 8a shows the distribution of obstacles generated using this approach. The high cost region marked in green is a local optima corresponding to the *Type-1* failure mode shown in Fig. 8b, which is generated by Algorithm-1. Note that random sampling is not able to locate *Type-2* failures effectively, as *Type-2* failures are sensitive to small perturbations in the location of **Obstacle-2**, but our algorithm succesfully locates several *Type-1* and *Type-2* failures with 0.14X the sample size of Brute-force method.

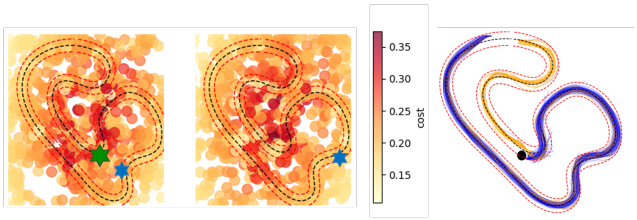
For demonstration of our technique, we also compare the performance with baseline algorithms for a shorter length trajectory tracking example over 5 chains of 50 epochs each. The results have been summarized in Table-III. Here, we notice that Brute-force-II performs equivalent to 2^{nd}-order LA, which is primarily due to the restricted search space of sampling region. This shows that for smaller sampling volumes, both gradient-based and gradient-free methods perform equally well, therefore, in the absence of true gradient information, the neural projection technique can be helpful in reducing sampling volume and thereby increasing the efficiency of sampling algorithm. The contribution of neural projection is evident from the increase in failure discovery rate of Brute-force-II compared to Brute-force-I.

We notice that 2^{nd}-order LA outperforms the baselines, by doing well across exploration and convergence.

While we observe faster convergence for gradient-based methods (higher mean cost), however, it is observed that convergence to local minima does not always lead to a collision due to the stochastic behavior of controller.

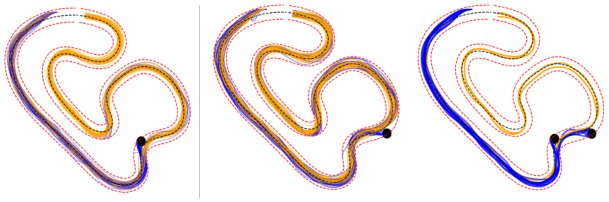
E. Experimental Validation

We analyze the sim-to-real transfer of our approach by conducting hardware experiments for testing of static EVs (obstacles, Section-V-D) and sequence EVs (reference path, Section-V-B) using the AutoRally racing platform and the F1/10 platform respectively.



(a) **(b)** Type-1 failure

Fig. 8: Fig. 8a shows Cost distribution of **Obstacle-1** (left) and **Obstacle-2** (right), generated using Brute Force method. Examples of obstacle corresponding to *Type-1* failure (green) and *Type-2* (blue) shown. Fig. 8b shows the collision likelihood with the *Type-1* obstacle (94%). Note that the collision happens regardless of the location of **Obstacle-2**



(a) Obstacle-1 **(b)** Obstacle-2 **(c)** Type-2 failure

Fig. 9: Fig. 9a and Fig. 9b shows the likelihood of collision of **Obstacle-1** (61%) and **Obstacle-2** (39%) respectively. Fig. 9c shows collision scenarios with both **Obstacle-1**, **Obstacle-2**. The combination has a 81% likelihood of collision (*Type-2 failure*).

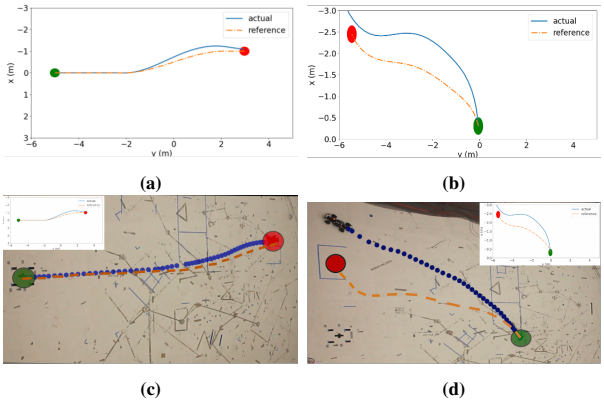
1) LQR reference path tracking falsification using F1/10 platform: We performed hardware demonstration for reference path falsification using the F1/10 autonomous racing platform [19]. The LQR controller for Speed+Steering Control presented in Section-V-B was implemented on the vehicle for tracking a known reference path. The vehicle's actual dynamics is not fully captured by a bicycle model, as contributions corresponding to tire dynamics and slip are neglected in simulation. The experiment was conducted on low reference speed, $v_{ref} = 0.5 - 0.8 m/s$ and steering limits of $\delta_{max} = 10°$. Within these limits, the vehicle was able to track several nominal reference paths.

Using our approach, we generated a falsifying reference path with high expected deviation and implemented it in real-time. As observed in simulation, the vehicle was not able to reach the goal and deviated significantly from reference path, shown in Fig. 10. We observe that the vehicle trajectory deviates marginally from simulated response, which is likely due to the difference in the true dynamics of the vehicle and that used in simulation. This deviation reflects the sim to real gap of our model-based testing framework, which we aim to address in the future iterations of our work.

2) MPPI Trajectory tracking with obstacle avoidance falsification using AutoRally platform : We validate the falsification of MPPI on the AutoRally platform in the presence of obstacles with a goal to replicate the *Type-1* and *Type-2* failures observed in Fig. 9. Fig. 11 shows the hardware demonstration of the failures discovered in simulation.

TABLE III: MPPI Baseline comparison

	Failure rate	Mean cost	Max cost	C_{disp}
Brute force-I	0.06	0.35	0.53	24.11
Brute force-II	0.116	0.37	0.55	23.57
1st-order LA	0.081	0.44	0.56	11.84
2nd-order LA	**0.1**	**0.48**	**0.59**	**17.06**



(a) (b)

(c) (d)

Fig. 10: Fig. 10c shows an example of a reference path that can be tracked by implementing the LQR controller explained in Section-V-B. Fig. 10d shows an example of a reference path that cannot be tracked by the vehicle, in simulation and real-time

VI. Conclusion & Future Work

This work presents a novel perspective to Bayesian-inference enabled testing using learning-based methods. While the direct deployment of neural network models for testing can be challenging due to the rarity and unfamiliarty of the failure modes, we exploit the expressivity of data-driven architectures in learning the behavior of environment variables. Firstly, this allows us to utilize differentiable optimization tools for testing and validation. As seen from various examples presented in the paper, learning-based representations can solve the curse of dimensionality using latent variable encoding and enable an explicit constraint satisfaction scheme using the neural projection technique.

We also provide theoretical insights exploring the role of acceleration in discovering diverse failure modes. Across all examples, we observe that 2nd-order LA provides an optimal way to balance coverage and failure discovery rate.

This paper analyzes the role of differentiability in testing, and the conclusion is that differentiability can substantially help in feedback control based architectures, as the explicit construction of gradients via backpropagation is easy. For optimal control architectures such as SQP and MPPI, we would benefit from an easy to implement sensitivity analysis. This forms the premise of future scope of this research. Furthermore, the question of hierarchical failures introduced in Section-V-D has promising scope in significantly improving the memory and convergence of testing methods.

(a) (b)

Fig. 11: Fig. 11a, Fig. 11b shows the hardware demonstration of *Type-1* and *Type-2* failure respectively found using our approach. Corresponding simulation and analysis in Fig. 8 and Fig. 9

VII. Acknowledgments

References

[1] Y. Annpureddy, C. Liu, G. Fainekos, and S. Sankaranarayanan, "S-taliro: A tool for temporal logic falsification for hybrid systems," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2011, pp. 254–257.

[2] P. Koopman and M. Wagner, "Autonomous vehicle safety: An interdisciplinary challenge," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 1, pp. 90–96, 2017.

[3] E. Wong and Z. Kolter, "Provable defenses against adversarial examples via the convex outer adversarial polytope," in *International conference on machine learning*. PMLR, 2018, pp. 5286–5295.

[4] S. S. Du, C. Jin, J. D. Lee, M. I. Jordan, A. Singh, and B. Poczos, "Gradient descent can take exponential time to escape saddle points," *Advances in neural information processing systems*, vol. 30, 2017.

[5] C. Dawson and C. Fan, "Robust counterexample-guided optimization for planning from differentiable temporal logic," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 7205–7212.

[6] A. Kundu, S. Gon, and R. Ray, "Data-driven falsification of cyber-physical systems," in *Proceedings of the 17th Innovations in Software Engineering Conference*, 2024, pp. 1–5.

[7] H. Delecki, A. Corso, and M. Kochenderfer, "Model-based validation as probabilistic inference," in *Learning for Dynamics and Control Conference*. PMLR, 2023, pp. 825–837.

[8] C. Dawson and C. Fan, "A bayesian approach to breaking things: efficiently predicting and repairing failure modes via sampling," in *7th Annual Conference on Robot Learning*, 2023. [Online]. Available: https://openreview.net/forum?id=fNLBmtyBiC

[9] B. Tzen, T. Liang, and M. Raginsky, "Local optimality and generalization guarantees for the langevin algorithm via empirical metastability," in *Conference On Learning Theory*. PMLR, 2018, pp. 857–875.

[10] Y.-A. Ma, N. S. Chatterji, X. Cheng, N. Flammarion, P. L. Bartlett, and M. I. Jordan, "Is there an analog of nesterov acceleration for gradient-based mcmc?" 2021.

[11] A. Parashar, "Accelerated algorithms for constrained optimization and control," Ph.D. dissertation, Massachusetts Institute of Technology, 2023.

[12] Y. Nesterov, *Lectures on convex optimization*. Springer International Publishing, 2018, vol. 137.

[13] X. Gao, M. Gurbuzbalaban, and L. Zhu, "Breaking reversibility accelerates langevin dynamics for non-convex optimization," *Advances in Neural Information Processing Systems*, vol. 33, pp. 17 850–17 862, 2020.

[14] Y.-A. Ma, Y. Chen, C. Jin, N. Flammarion, and M. I. Jordan, "Sampling can be faster than optimization," *Proceedings of the National Academy of Sciences*, vol. 116, no. 42, pp. 20 881–20 885, 2019.

[15] J. Yin, Z. Zhang, E. Theodorou, and P. Tsiotras, "Trajectory distribution control for model predictive path integral control using covariance steering," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 1478–1484.

[16] S. Yang, X. He, and B. Zhu, "Learning physical constraints with neural projections," *Advances in Neural Information Processing Systems*, vol. 33, pp. 5178–5189, 2020.

[17] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt, M. Sokolsky, G. Stanek, D. Stavens, A. Teichman, M. Werling, and S. Thrun, "Towards fully autonomous driving: Systems and algorithms," in *2011 IEEE Intelligent Vehicles Symposium (IV)*, 2011, pp. 163–168.

[18] S. Singh, J.-J. Slotine, and V. Sindhwani, "Optimizing trajectories with closed-loop dynamic sqp," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 5249–5254.

[19] B. D. Evans, R. Trumpp, M. Caccamo, H. W. Jordaan, and H. A. Engelbrecht, "Unifying f1tenth autonomous racing: Survey, methods and benchmarks," *arXiv preprint arXiv:2402.18558*, 2024.