Probably Approximately Correct Nonlinear Model Predictive Control (PAC-NMPC)

Adam Polevoy, Marin Kobilarov, and Joseph Moore, *Member, IEEE*

***Abstract*—Approaches for stochastic nonlinear model predictive control (SNMPC) typically make restrictive assumptions about the system dynamics and rely on approximations to characterize the evolution of the underlying uncertainty distributions. For this reason, they are often unable to capture more complex distributions (e.g., non-Gaussian or multi-modal) and cannot provide accurate guarantees of performance. In this letter, we present a sampling-based SNMPC approach that leverages recently derived sample complexity bounds to certify the performance of a feedback policy without making assumptions about the system dynamics or underlying uncertainty distributions. By parallelizing our approach, we are able to demonstrate real-time receding-horizon SNMPC with statistical safety guarantees in simulation and on hardware using a 1/10th scale rally car and a 24-inch wingspan fixed-wing unmanned aerial vehicle (UAV).**

***Index Terms*—Integrated planning and control, planning under uncertainty, robot safety.**

I. Introduction

NONLINEAR model predictive control (NMPC) has proven to be a powerful approach for controlling high-dimensional, complex robotic systems (e.g., [1], [2], [3], [4]). Nevertheless, although these methods can handle large state spaces, nonlinear dynamics, and system constraints, their performance can be adversely affected by the presence of uncertainty even in the context of real-time replanning [5]. A number of approaches have been proposed to compensate for this marginal robustness, the simplest of which is to generate a feedback policy to track the current receding-horizon plan (e.g., [2]). These approaches, however, do not account for uncertainty or closed-loop performance during the planning process. To address this, approaches such as robust NMPC (RNMPC) and stochastic NMPC (SNMPC) attempt to reason about the response of the policy to disturbances during planning. This is usually accomplished by making simplifying assumptions about the dynamics and disturbances. For instance, in RNMPC, disturbance sets may be approximated by ellipsoids [6], or robustness guarantees may require the existence of a control contraction metric (CCM) [7]; in SNMPC, uncertainty distributions are often approximated via Gaussians [8] or sampling [9].

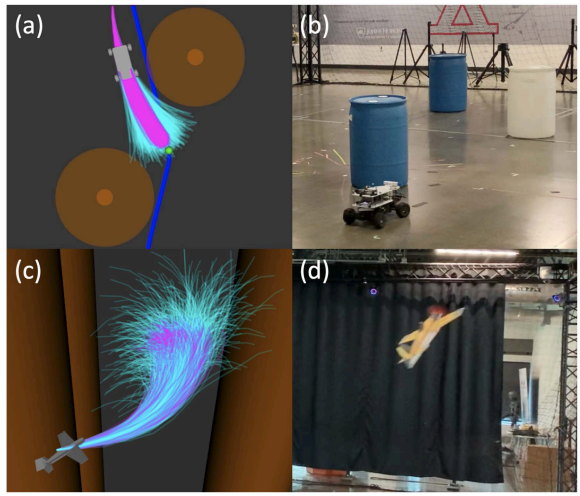

Fig. 1. PAC-NMPC optimizes high confidence bounds on the expected costs/constraint of feedback policy distributions. We demonstrated our approach on a 1/10th-scale Rally Car (a), (b) and an Edge 540 fixed-wing UAV (c), (d).

In this letter, we present a sampling-based SNMPC algorithm capable of controlling stochastic dynamical systems without applying limiting assumptions to the structure of the stochastic dynamics or underlying uncertainty distributions. In addition, our approach leverages recently derived sample complexity bounds [10] to provide a probabilistic guarantee on system performance. Our algorithm builds on Probably Approximately Correct Robust Policy Search (PROPS) [11] to directly optimize an upper confidence bound on the expected cost and the probability of constraint violation for a receding-horizon feedback policy. Not only does this approach encourage robust policy generation, but the minimized bounds themselves provide a statistical guarantee for each planning interval. To achieve real-time performance, we use a graphics processing unit (GPU) to parallelize our algorithm. We then evaluate our approach in simulation and on hardware using a 1/10th scale rally car and a 24-inch wingspan fixed-wing UAV (Fig. 1).

Our contributions are:

1) A novel algorithm, PAC-NMPC, for receding horizon SNMPC with probabilistic performance guarantees.
2) A real-time implementation via GPU acceleration.

Manuscript received 7 May 2023; accepted 24 August 2023. Date of publication 13 September 2023; date of current version 27 September 2023. This letter was recommended for publication by Associate Editor P. Di Lillo and Editor C. Gosselin upon evaluation of the reviewers' comments. This work was supported by JHU APL internal R&D. *(Corresponding author: Adam Polevoy.)*

Adam Polevoy and Joseph Moore are with the Applied Physics Lab, Johns Hopkins University, Laurel, MD 20723 USA, and also with the Whiting School of Engineering, Department of Mechanical Engineering, Johns Hopkins University, Baltimore, MD 21218 USA (e-mail: adam.polevoy@jhuapl.edu; joseph.moore@jhuapl.edu).

Marin Kobilarov is with the Whiting School of Engineering, Department of Mechanical Engineering, Johns Hopkins University, Baltimore, MD 21218 USA (e-mail: mkobila1@jhu.edu).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2023.3315209, provided by the authors.

Digital Object Identifier 10.1109/LRA.2023.3315209

3) Demonstration of our algorithm on complex underactuated systems via simulation and hardware experiments.

II. Related Work

Prior research addresses the impact of uncertainty on NMPC via robust NMPC (RNMPC) or stochastic NMPC (SNMPC). RNMPC represents uncertainty in the system as bounded disturbances. While min-max tube NMPC methods [6], [12], [13] and multi-stage NMPC approaches [14] have been proposed, constraint-tightening based tube NMPC [15] has proved to be the most common and computationally tractable RNMPC approach. Most of these approaches rely on tube size derived from a computed Lipschitz constant [16], [17], CCMs [7], [18], [19], or bounds on incremental stability [20]. These methods for computing invariant sets are often complex, may place significant restrictions on the dynamics, and can lead to overly conservative results. Recent work has proposed sampling-based reachability analysis for arbitrary continuously differentiable dynamics with less conservative results [21], [22].

Stochastic NMPC represents uncertainty as a probability distribution and optimizes the expectation over costs and constraint violations. Oftentimes, they optimize a feedback policy, rather than a control sequence. Our method, PAC-NMPC, falls into this category of approaches. Early approaches focused on a local iterative solution to the stochastic Hamilton-Jacobi-Bellman equation [23], [24], which relies on a first order approximation of the dynamics for noise propagation. Stochastic differential dynamic programming approaches [25] would later allow for second order approximation of the dynamics. More recent methods utilize the unscented transform to propagate noise through nonlinear dynamics [8], [26]. In [27], the authors leverage path integral control to generate open-loop trajectories and create a sampling-based NMPC algorithm. More recent extensions [9], [28] include a feedback policy for robust performance, [29] provides guarantees on free-energy growth, and [30] penalizes Conditional Value-at-Risk (CVaR).

Sampling-based stochastic optimal control methods are closely related to policy search in reinforcement learning. A general background and survey of common policy search methods can be found in [31]. In [11], researchers developed an approach for robust policy search using a Probably Approximately Correct (PAC) learning framework called PAC Robust Policy Search (PROPS) based on recently derived sample complexity bounds in [10]. Researchers have also recently explored (PAC)-Bayes theory to generate collision-avoidance policies with performance guarantees in novel environments [32] and achieve vision-based planning with motion primitives [33]. Our PAC-NMPC method builds on the work in [10] and [11] to develop a highly generalizable sampling-based stochastic feedback motion-planning algorithm that can provide statistical performance guarantees for a large class of uncertain dynamical systems.

III. Background

Before presenting our approach for Probably Approximately Correct NMPC, we first review the recently derived PAC bounds for Iterative Stochastic Policy Optimization which are the foundation for our approach.

Algorithm 1: Episodic ISPO.

```
Inputs: ν̂*_0, i ← 0;
while termination condition not met do
    for j = 1, ..., M do
        Sample Trajectory: (τ_j, ξ_j) ~ p(·, ·|ν̂*_i);
        Evaluate Cost: J_j = J(τ_j);
    ν̂*_{i+1} = arg min_ν Ĵ(ν) + αD(ν, ν̂*_i);
    i ← i + 1
return ν̂*_i
```

A. Iterative Stochastic Policy Optimization

Consider the stochastic dynamics given by $p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$ defined by a vector of state values $\mathbf{x}_t \in \mathbb{R}^{N_x}$, a vector of control inputs $\mathbf{u}_t \in \mathbb{R}^{N_u}$, and probability density p. Iterative Stochastic Policy Optimization (ISPO) [10] formulates the search for a control policy $\mathbf{u}_t = \boldsymbol{\pi}_t(\mathbf{x}_t, \boldsymbol{\xi})$ as a stochastic optimization problem, where $\boldsymbol{\xi}$ is a set of policy parameters. Specifically, ISPO introduces a surrogate distribution $p(\boldsymbol{\xi}|\boldsymbol{\nu})$ where policy parameters $\boldsymbol{\xi}$ are dependent on distribution hyper-parameters $\boldsymbol{\nu}$. This induces a joint distribution

$$p(\boldsymbol{\tau}, \boldsymbol{\xi}|\boldsymbol{\nu}) = p(\boldsymbol{\tau}|\boldsymbol{\xi})\, p(\boldsymbol{\xi}|\boldsymbol{\nu}) \tag{1}$$

where $p(\boldsymbol{\tau}|\boldsymbol{\xi})$ represents the natural stochasticity of the system and is given as $p(\boldsymbol{\tau}|\boldsymbol{\xi}) = p(\mathbf{x}_0) \prod_{t=0}^{T} p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$, where $\mathbf{u}_t = \boldsymbol{\pi}(\mathbf{x}_t, \boldsymbol{\xi})$. Here $\boldsymbol{\tau}$ represents the discrete time trajectory sequence $\{\mathbf{x}_0, \mathbf{u}_0, \mathbf{x}_1, \mathbf{u}_1 \ldots, \mathbf{u}_{N_T}, \mathbf{x}_{N_T+1}\}$, where N_T is the number of timesteps.

In general, the objective of ISPO is to solve the optimization problem $\boldsymbol{\nu}^* = \arg\min_{\boldsymbol{\nu}} \mathcal{J}(\boldsymbol{\nu})$ where $\mathcal{J}(\boldsymbol{\nu}) = \mathbb{E}_{\boldsymbol{\tau}, \boldsymbol{\xi} \sim p(\cdot, \cdot|\boldsymbol{\nu})}[J(\boldsymbol{\tau})]$. To do this, ISPO iteratively samples from (1) and solves the optimization problem $\widehat{\boldsymbol{\nu}}^*_{i+1} = \arg\min_{\boldsymbol{\nu}} \widehat{\mathcal{J}}(\boldsymbol{\nu}) + \alpha D(\boldsymbol{\nu}, \widehat{\boldsymbol{\nu}}^*_i)$ for each iteration i until convergence. Here $\widehat{\mathcal{J}}(\boldsymbol{\nu})$ is an empirical approximation of the expected cost, D is a distance between distributions, often the $\mathcal{KL}$-divergence, and $\alpha > 0$ is hand-tuned weight or Lagrange multiplier [34] computed for a constraint on $D(\boldsymbol{\nu}, \widehat{\boldsymbol{\nu}}^*_i)$. Algorithm 1 is a general framework for ISPO.

B. PAC Bounds for Stochastic Policy Search

Instead of directly minimizing an empirical approximation of the expected cost, it can be beneficial to minimize *an upper confidence bound* on the expected cost. Not only does such an approach encourage robust policies, but it also can provide guarantees on future performance. In this letter, we directly optimize the PAC bounds derived in [10]. While these bounds are derived in [10] and discussed in [11], we review them here since they are fundamental to our approach.

In [10] the presented PAC bound $\mathcal{J}^+_\alpha(\boldsymbol{\nu})$ takes the form

$$\mathcal{J}^+_\alpha(\boldsymbol{\nu}) \triangleq \widehat{\mathcal{J}}_\alpha(\boldsymbol{\nu}) + \alpha d(\boldsymbol{\nu}) + \Phi_\alpha(\delta) \tag{2}$$

where $\widehat{\mathcal{J}}_\alpha(\boldsymbol{\nu})$ is a robust estimator [35] of the expected cost, $d(\boldsymbol{\nu})$ is a distance between distributions, and $\Phi_\alpha(\delta)$ is a concentration-of-measure term which accounts for discrepancies between the true mean, $\mathcal{J}(\boldsymbol{\nu})$, and the robust estimator, $\widehat{\mathcal{J}}_\alpha(\boldsymbol{\nu})$. For a particular choice of $\widehat{\mathcal{J}}_\alpha(\boldsymbol{\nu})$, $d(\boldsymbol{\nu})$, and $\Phi_\alpha(\delta)$, [10] proves that $\mathcal{J}^+_\alpha(\boldsymbol{\nu})$ bounds the expected cost $\mathcal{J}(\boldsymbol{\nu})$ with a probability of $1 - \delta$.

To derive an expression for $\widehat{\mathcal{J}}_\alpha(\boldsymbol{\nu})$, we must estimate

$$\mathcal{J}(\boldsymbol{\nu}) = \mathbb{E}_{\boldsymbol{\tau},\boldsymbol{\xi}\sim p(\cdot,\cdot|\boldsymbol{\nu}_0)}\left[J(\boldsymbol{\tau})\frac{p(\boldsymbol{\xi}|\boldsymbol{\nu})}{p(\boldsymbol{\xi}|\boldsymbol{\nu}_0)}\right]. \tag{3}$$

Because the likelihood ratio $\frac{p(\boldsymbol{\xi}|\boldsymbol{\nu})}{p(\boldsymbol{\xi}|\boldsymbol{\nu}_0)}$ can be unbounded, a robust estimation technique is needed. Following [35], the expected value of a random variable X can be approximated as $\mathbb{E}[X] \approx \frac{1}{\alpha M}\sum_{i=0}^{M}\psi(\alpha X_i)$ for some and $\alpha > 0$, where X_i is a sampled value of X and $\psi(x) = \log(1 + x + \frac{1}{2}x^2)$. Given L prior surrogate distributions with hyper-parameters $\boldsymbol{\nu}_0$, $\boldsymbol{\nu}_1$,... $\boldsymbol{\nu}_{L-1}$ and M iid samples from each joint distribution, $(\boldsymbol{\tau}_{i0}, \boldsymbol{\xi}_{i0})$, $(\boldsymbol{\tau}_{i1}, \boldsymbol{\xi}_{i1})$,..., $(\boldsymbol{\tau}_{iM}, \boldsymbol{\xi}_{iM}) \sim p(\cdot,\cdot|\boldsymbol{\nu}_i)$, the robust estimate of the expected cost can then be written as

$$\widehat{\mathcal{J}}_\alpha(\boldsymbol{\nu}) \triangleq \frac{1}{\alpha LM}\sum_{i=0}^{L-1}\sum_{j=1}^{M}\psi\left(\alpha\ell_{ij}\right) \tag{4}$$

$$\ell_{ij} = J(\boldsymbol{\tau}_{ij})\frac{p(\boldsymbol{\xi}_{ij}|\boldsymbol{\nu})}{p(\boldsymbol{\xi}_{ij}|\boldsymbol{\nu}_i)} \tag{5}$$

To compute the distance between the distributions, the Renyi divergence, D_2, is used. $d(\boldsymbol{\nu})$ is given as

$$d(\boldsymbol{\nu}) \triangleq \frac{1}{2\,L}\sum_{i=0}^{L-1} b_i^2 e^{D_2(p(\cdot|\boldsymbol{\nu})||(p(\cdot|\boldsymbol{\nu}_i))} \tag{6}$$

$$0 \le J(\boldsymbol{\tau}_{ij}) \le b_i \;\forall j = 0,\ldots,M \tag{7}$$

The concentration-of-measure term $\Phi_\alpha(\delta)$ is given as

$$\Phi_\alpha(\delta) \triangleq \frac{1}{\alpha LM}\log\frac{1}{\delta}. \tag{8}$$

It tightens the bound as the number of samples increases and as the bound confidence, $1-\delta$, decreases.

In addition to cost-functions, we can also incorporate the probability of constraint violation, $\mathcal{C}(\boldsymbol{\nu})$, of the form $\mathcal{C}(\boldsymbol{\nu}) = \mathbb{P}(g(\boldsymbol{\tau}) > 0) = \mathbb{E}_{\boldsymbol{\tau},\boldsymbol{\xi}\sim p(\cdot,\cdot|\boldsymbol{\nu})}[\mathbb{I}_{\{g(\boldsymbol{\tau})>0\}}]$, where $\mathbb{I}$ is the indicator function. Similarly, we can derive $\mathcal{C}_\alpha^+(\boldsymbol{\nu})$ which upper bounds $\mathcal{C}(\boldsymbol{\nu})$ with probability $1-\delta$.

To combine costs and constraints, we optimize a weighted sum of $\mathcal{J}_\alpha^+(\boldsymbol{\nu})$ and $\mathcal{C}_\alpha^+(\boldsymbol{\nu})$, where $\gamma > 0$ is a heuristically selected weighting coefficient. We solve

$$\boldsymbol{\nu}_{i+1} = \boldsymbol{\nu}^* = \arg\min_{\boldsymbol{\nu}}\min_{\alpha>0}\left(\mathcal{J}_\alpha^+(\boldsymbol{\nu}) + \gamma\mathcal{C}_\alpha^+(\boldsymbol{\nu})\right) \tag{9}$$

using a GPU implementation of L-BFGS-B [36]. When optimizing the bounds, we used self-normalized importance weights to compute ℓ_{ij} in (5) and analytical gradients, as described in [11].

IV. Approach

To formulate a receding-horizon NMPC algorithm that leverages the PAC bounds in [10], we first address the stochastic trajectory optimization problem. We then extend this approach to enable real-time feedback motion planning.

A. PAC Stochastic Trajectory Optimization

To formulate stochastic trajectory optimization using PAC bounds, we consider a parameterization of a discrete-time open-loop policy $\mathbf{u}_t = \boldsymbol{\pi}_t(\boldsymbol{\xi}) = \boldsymbol{\zeta}_t$, where $\boldsymbol{\xi} = \begin{bmatrix}\boldsymbol{\zeta}_0^T & \boldsymbol{\zeta}_1^T & \ldots & \boldsymbol{\zeta}_{N_T}^T\end{bmatrix}^T$, $\boldsymbol{\zeta}_t \in \mathbb{R}^{N_u}$, and N_T is the time horizon for the trajectory.

Algorithm 2: PAC Stochastic Trajectory Optimization.

```
Inputs: ν̂*_0, x_0, i ← 0;
while termination condition not met do
    for j = 1,...,M do                    // GPU Parallelized
        τ_ij, ξ_ij ~ p(·,·|ν_i, x_0);
        J_ij = J(τ_ij), c_ij = c(τ_ij);
    ν̂*_{i+1} ← arg min_ν min_{α>0}(J⁺_α(ν) + γC⁺_α(ν));
    i ← i + 1;
return ν̂*_i, J⁺_α(ν̂*_i), C⁺_α(ν̂*_i)
```

Algorithm 3: $\boldsymbol{\tau} \sim p(\cdot|\boldsymbol{\xi}, \mathbf{x}_0)$.

```
Inputs: ξ = [u_0^T, ..., u_{N_T}^T]^T, x_0;
for t = 0,...,N_T do                      // Stochastic Rollout
    x_{t+1} ~ p(·|x_t, u_t);
Return τ = {x_0, u_0, x_1, u_1, ..., u_{N_T}, x_{N_T+1}}
```

We parameterize the surrogate distribution $p(\boldsymbol{\xi}|\boldsymbol{\nu})$ as a multivariate Gaussian over this discrete-time control trajectory so that $\boldsymbol{\xi} \sim \mathcal{N}(\boldsymbol{\xi}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$. For computational efficiency, we assume a diagonal covariance matrix $\boldsymbol{\Sigma}$ so the distribution parameters are given as $\boldsymbol{\nu} \triangleq \left[\boldsymbol{\mu}^T, diag(\boldsymbol{\Sigma})^T\right]^T$ where $diag(\boldsymbol{\Sigma}) = \begin{bmatrix}\boldsymbol{\eta}_0^T & \boldsymbol{\eta}_1^T & \ldots & \boldsymbol{\eta}_{N_T}^T\end{bmatrix}^T$ and $\boldsymbol{\eta}_t \in \mathbb{R}^{N_u}$. Thus, a control trajectory with an N_u-dimensional control input and N_T timesteps, the surrogate distribution is parameterized as a $N_u N_T$-dimensional multivariate Gaussian where $\boldsymbol{\mu} \in \mathbb{R}^{N_u N_T}$ and $diag(\boldsymbol{\Sigma}) \in \mathbb{R}^{N_u N_T}$.

For each optimization iteration, M trajectories are sampled from the joint distribution (1) and evaluated using the augmented cost in (9). Sampling from the joint distribution is achieved by first sampling policy parameters from the surrogate distribution and then using these policy parameters to sample from the stochastic dynamics as shown in Algorithm 3. The augmented costs of the samples are used to find a new set of policy parameters that minimize the weighted sum of the cost and constraint bounds (Algorithm 2). This process iterates until a termination condition (e.g. maximum time, convergence metric, etc.) is reached. The optimization not only returns the optimized policy parameters, but also the optimized bounds themselves.

An analytical formulation for the surrogate distribution is necessary to compute the robust estimates (4) and Renyi divergences (7). By contrast, the only requirement on the stochastic dynamics is to permit sampling. This property allows our algorithm to generalize to a large class of complex (potentially "black-box") stochastic dynamics models. In practice, the cost and constraint functions can also be stochastic (e.g. $J_{ij}, c_{ij} \sim p(\cdot,\cdot|\boldsymbol{\tau}_{ij})$).

B. PAC Feedback Motion Planning

While the ability to compute PAC open-loop policies is valuable, ideally, we would like to compute closed-loop policies of the form $\boldsymbol{\pi}_t(\mathbf{x}_t, \boldsymbol{\xi})$. A closed-loop policy should enable us to generate tighter state-space trajectory distributions and improve the ability of our policies to satisfy control objectives. However, designing feedback policies is not straightforward and can be very system specific. One fairly general approach

Algorithm 4: $\boldsymbol{\tau} \sim p(\cdot|\boldsymbol{\xi}, \mathbf{x}_0)$ With Feedback.

Inputs: $\boldsymbol{\xi} = [\mathbf{u}_0, ..., \mathbf{u}_{N_T}]$, $\mathbf{x}_0$;
$\mathbf{x}_0^d = \mathbf{x}_0$;
for $k = 0, \ldots, N_T$ **do** // Nominal Rollout
  $\mathbf{x}_{t+1}^d = \mathbf{x}_t^d + f(\mathbf{x}_t^d, \mathbf{u}_t^d)\Delta t$;
$\boldsymbol{\tau}^d = \{\mathbf{x}_0^d, \mathbf{u}_0^d, \mathbf{x}_1^d, \mathbf{u}_1^d ..., \mathbf{u}_{N_T}^d, \mathbf{x}_{N_T+1}^d\}$;
$\boldsymbol{\kappa} = \text{TVLQR}(\boldsymbol{\tau}^d)$;
for $t = 0, \ldots, N_T$ **do** // Feedback Rollout
  $\mathbf{u}_t = \mathbf{K}_t \left(\mathbf{x}_t - \mathbf{x}_t^d\right) + \mathbf{u}_t^d$;
  $\mathbf{x}_{t+1} \sim p(\cdot|\mathbf{x}_t, \mathbf{u}_t)$;
Return $\boldsymbol{\tau} = \{\mathbf{x}_0, \mathbf{u}_0, \mathbf{x}_1, \mathbf{u}_1, ..., \mathbf{u}_{N_T}, \mathbf{x}_{N_T+1}\}$

might be to select a local feedback control policy of the form $\mathbf{u}_t = \mathbf{K}_t(\mathbf{x}_t^d - \mathbf{x}_t) + \mathbf{u}_t^d$, where $\boldsymbol{\kappa} = \{\mathbf{K}_0, \mathbf{K}_1, \ldots, \mathbf{K}_{N_T}\}$ is a sequence of time-varying gains, $\mathbf{K}_t \in \mathbb{R}^{N_u \times N_x}$. $\mathbf{x}_t^d \in \mathbb{R}^{N_x}$ is a nominal state trajectory and $\mathbf{u}_t^d \in \mathbb{R}^{N_u}$ is a nominal input trajectory. All of the elements of these sequences are parameters of the feedback policy. Assuming the surrogate distribution is a mulitvariate Gaussian with a diagonal covariance matrix, in this case, $\boldsymbol{\xi} \in \mathbb{R}^{2(N_x N_u + N_x + N_u)N_T}$. Given the size of the decision space, this parameterization of the policy is unlikely to be computationally tractable. To overcome this challenge, we instead use a local closed-loop policy of the form

$$\mathbf{u}_t = \mathbf{K}_t\left(\boldsymbol{\tau}^d(\boldsymbol{\xi}, \mathbf{x}_0)\right)\left(\mathbf{x}_t^d(\boldsymbol{\xi}, \mathbf{x}_0) - \mathbf{x}_t\right) + \mathbf{u}_t^d(\boldsymbol{\xi}). \tag{10}$$

Here we refer to $\boldsymbol{\tau}^d \triangleq \{\mathbf{x}_0^d, \mathbf{u}_0^d, \mathbf{x}_1^d, \mathbf{u}_1^d \ldots, \mathbf{u}_{N_T}^d, \mathbf{x}_{N_T+1}^d\}$ as the nominal trajectory, which is computed using a nominal deterministic discrete-time dynamics model, $\mathbf{f}(\mathbf{x}_t^d, \mathbf{u}_t^d)$,

$$\mathbf{x}_{t+1}^d = \mathbf{x}_t^d + \mathbf{f}(\mathbf{x}_t^d, \mathbf{u}_t^d)\Delta t \tag{11}$$

where $\mathbf{u}_t^d = \boldsymbol{\zeta}_t$. To sample from this feedback policy, we must first compute the nominal trajectory, $\boldsymbol{\tau}^d$. However, $\boldsymbol{\tau}^d$ itself is dependent on $\boldsymbol{\xi}$; for this reason, we employ a two-stage sampling process. In the first stage, we sample from the trajectory distribution $p(\boldsymbol{\tau}^d|\boldsymbol{\xi})$ using the deterministic dynamics model (11) and the surrogate distribution $p(\boldsymbol{\xi}|\boldsymbol{\nu})$. For each sample trajectory $\boldsymbol{\tau}^d$, we the compute the time-varying gains $\mathbf{K}_t(\boldsymbol{\tau}^d)$. In the second stage, we sample from the closed-loop stochastic dynamics using the feedback policy (10). This approach, summarized in Algorithm 4, certifies the feedback policy by allowing our sampled costs and constraints to capture the impact of our local feedback policy on our stochastic dynamics model.

We use the finite horizon, discrete, time-varying linear quadratic regulator (TVLQR) [37] to compute the feedback gains $\mathbf{K}_t(\boldsymbol{\tau}^d)$. For each sampled nominal trajectory, the gains are computed by integrating the finite horizon discrete-time Riccati equations backward in time using the nominal dynamics linearized about $\boldsymbol{\tau}^d$.

C. PAC-NMPC

Finally, we propose a receding-horizon NMPC approach based on our PAC feedback motion planning algorithm (Algorithm 5). At each planning interval, with a period of H, the trajectory distribution and corresponding feedback policy is optimized given the current state, $\mathbf{x_0}$, and the prior hyper-parameters, $\widehat{\boldsymbol{\nu}}_0^*$. Then, the trajectory and feedback gains are trimmed by $\frac{H}{\Delta t}$ to account for time passed since the beginning of the planning interval and executed by the controller.

Algorithm 5: PAC-NMPC.

Input: $\widehat{\boldsymbol{\nu}}_0^*$;
$\mathbf{x}_0 = \text{GetCurrentState}()$;
while *objective not completed* **do**
  $\widehat{\boldsymbol{\nu}}^* = \text{Optimize}(\widehat{\boldsymbol{\nu}}_0^*, \mathbf{x}_0)$; // Alg. 2
  $\mathbf{u}^d = \text{MaximumLikelihoodEstimate}(\widehat{\boldsymbol{\nu}}^*)$;
  for $t = 0, \ldots, N_T$ **do** // Nominal Rollout
    $\mathbf{x}_{t+1}^d = \mathbf{x}_t^d + f(\mathbf{x}_t^d, \mathbf{u}_t^d)\Delta t$;
  $\boldsymbol{\tau}^d = \{\mathbf{x}_0^d, \mathbf{u}_0^d, \mathbf{x}_1^d, \mathbf{u}_1^d ..., \mathbf{u}_{N_T}^d, \mathbf{x}_{N_T+1}^d\}$;
  $\boldsymbol{\kappa} = \text{TVLQR}(\boldsymbol{\tau}^d)$;
  $\boldsymbol{\tau}^d, \boldsymbol{\kappa}, \widehat{\boldsymbol{\nu}}^* \leftarrow \text{Trim}(\boldsymbol{\tau}^d, \boldsymbol{\kappa}, \widehat{\boldsymbol{\nu}}^*)$;
  Execute $(\boldsymbol{\tau}^d, \boldsymbol{\kappa})$;
  $\mathbf{x}_0 = \text{GetCurrentState}()$;
  $\widehat{\boldsymbol{\nu}}_0^* = \text{InitializePrior}(\widehat{\boldsymbol{\nu}}^*, \boldsymbol{\tau}^d, \boldsymbol{\kappa}, \mathbf{x}_0)$; // Alg. 6
Return

Algorithm 6: Initialize Prior.

Input: $\widehat{\boldsymbol{\nu}}^*$, $\boldsymbol{\tau}^d$, $\boldsymbol{\kappa}$, $\mathbf{x}_0$;
$[\boldsymbol{\mu}_0^T, \boldsymbol{\mu}_1^T, ..., \boldsymbol{\mu}_{N_T - \frac{H}{\Delta t}}^T, \boldsymbol{\eta}_0^T, \boldsymbol{\eta}_1^T, ..., \boldsymbol{\eta}_{N_T - \frac{H}{\Delta t}}^T]^T = \widehat{\boldsymbol{\nu}}^*$;
$\mathbf{u}^d = \boldsymbol{\mu}$;
for $t = 0, \ldots, N_T - \frac{H}{\Delta t}$ **do** // Feedback Rollout
  $\mathbf{u}_t = \mathbf{K}_t \left(\mathbf{x}_t - \mathbf{x}_t^d\right) + \mathbf{u}_t^d$;
  $\mathbf{x}_{t+1} = \mathbf{x}_t + f(\mathbf{x}_t, \mathbf{u}_t)\Delta t$;
for $t = N_T - \frac{H}{\Delta t} + 1, \ldots, N_T$ **do** // Extend prior
  $\mathbf{u}_t = \mathbf{0}$;
  $\boldsymbol{\eta}_t = max(\boldsymbol{\eta}_{T - \frac{H}{\Delta t}}, \boldsymbol{\eta}_{min})$;
Return $\widehat{\boldsymbol{\nu}}_0^* = [\mathbf{u}_0^T, \mathbf{u}_1^T, ..., \mathbf{u}_{N_T}^T, \boldsymbol{\eta}_0^T, \boldsymbol{\eta}_1^T, ..., \boldsymbol{\eta}_{N_T}^T]^T$

The trim operation is given as $Trim(\boldsymbol{\kappa}) = \{\mathbf{K}_{\frac{H}{\Delta t}}, \ldots, \mathbf{K}_{N_T}\}$. Hyper-parameters determining the first $\frac{H}{\Delta t}$ time steps of the control trajectory represent control signals that the system is executing during the optimization and are not modified during the optimization. To execute the policy, we use the maximum likelihood estimate of the policy parameters, which is simply the mean, $\boldsymbol{\mu}$, of the multivariate Gaussian $p(\boldsymbol{\xi}|\boldsymbol{\nu})$.

Given that optimal trajectory distributions are expected to be similar between subsequent planning intervals, we use the optimal hyper-parameters from the last planning interval to initialize the prior policy distribution for the next. This initialization is extremely important since it allows the prior to start near an optimal solution, thus reducing the required number of iterations for convergence. After trimming, we apply the feedback policy to generate a modified policy parameter distribution with a mean feasible for the current initial state. Then, we extend the mean with zeros and the variance with its final value. We threshold the prior variance with $\boldsymbol{\eta}_{min}$ to prevent it from starting too small, which would inhibit exploration (Algorithm 6).

V. Simulation Experiments

A. Trajectory Optimization Experiments

We simulate a stochastic bicycle model with acceleration and steering rate inputs. We denote $\mathbf{x}_t = [x_0, x_1, x_2, x_3, x_4]^T$ as the state vector, $\mathbf{u}_t = [u_0, u_1]^T$ as the control vector, $l = 0.33$ as the wheel base, and $\boldsymbol{\Gamma} = diag([0.001, 0.001, 0.1, 0.2, 0.001])$ as

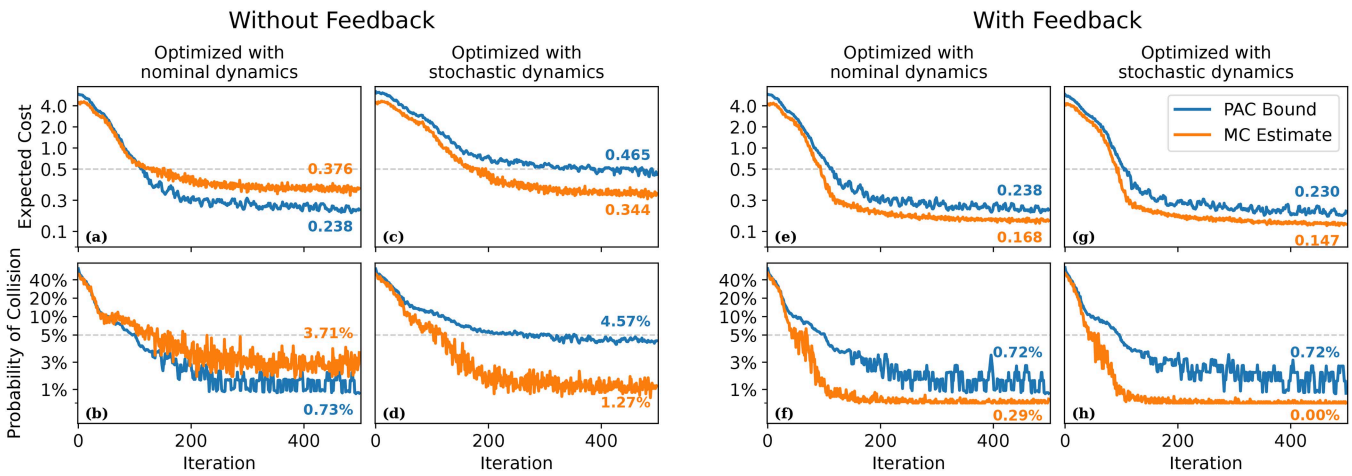

Fig. 2. Convergence of PAC Bound compared to MC estimates (evaluated with stochastic dynamics). Y-axis switches from log to linear scale halfway.

the covariance.

$$\begin{aligned} \mathbf{x}_{t+1} &\sim p\left(\cdot|\mathbf{x}_t, \mathbf{u}_t\right) \triangleq \mathbf{x}_t + \left(f\left(\mathbf{x}_t, \mathbf{u}_t\right) + \boldsymbol{\omega}\right)\Delta t \\ f\left(\mathbf{x}_t, \mathbf{u}_t\right) &= \left[x_3\cos(x_2), x_3\sin(x_2), x_3\tan(x_4)/l, u_0, u_1\right] \\ \boldsymbol{\omega} &\sim \mathcal{N}\left(\cdot|\mathbf{0}, \boldsymbol{\Gamma}\right) \end{aligned} \tag{12}$$

A 20 timestep trajectory with $\Delta t = 0.1$ sec is optimized, with an initial state of $\mathbf{x_I} = [0.0, 0.0, 0.0, 1.0, 0.0]^T$ and a goal state of $\mathbf{x_G} = [3.0, 0.0, 0.0, 1.0, 0.0]^T$. The steering angle is constrained to $(-0.4, 0.4)$ rad, the steering rate to $(-1.0, 1.0)$ $\frac{\text{rad}}{\text{sec}}$, and the acceleration to $(-1.0, 1.0)$ $\frac{m}{s^2}$. We apply a quadratic cost on the final state of the trajectory: $J(\boldsymbol{\tau}) = \mathbf{x}_{N_T+1}^T \mathbf{Q}_f \mathbf{x}_{N_T+1}$ where $\mathbf{Q_f} = diag([2.0, 2.0, 0.0, 0.0, 0.0])$. An obstacle constraint is applied with circular obstacles at $(1.0, 0.75)$ and $(2.0, -0.75)$. The initial prior distribution has a mean of all zeros and a variance of all ones. Parameters were as follows: $L = 5$, $M = 1024$, $\delta = 0.05, \gamma = 10$. Simulations were run on a laptop with an Intel Core i9-9880H CPU and a Nvidia GeForce RTX 2070 GPU.

We ran trajectory optimization for 500 iterations. Once completed, our method returns the hyper-parameters, $\widehat{\boldsymbol{\nu}}^*$, and the optimized PAC bounds, $\mathcal{J}_\alpha^+(\widehat{\boldsymbol{\nu}}^*)$ and $\mathcal{C}_\alpha^+(\widehat{\boldsymbol{\nu}}^*)$. It is guaranteed that policies sampled from $p(\boldsymbol{\xi}|\boldsymbol{\nu}^*)$ will have an expected cost of $\leq \mathcal{J}_\alpha^+(\widehat{\boldsymbol{\nu}}^*)$ and an expected constraint (probability of collision) of $\leq \mathcal{C}_\alpha^+(\widehat{\boldsymbol{\nu}}^*)$ with a 95% chance. To validate the PAC bounds, we compare them against Monte Carlo estimates of the expected cost, $\widehat{\mathcal{J}}(\widehat{\boldsymbol{\nu}}^*)$, and probability of collision, $\widehat{\mathcal{C}}(\widehat{\boldsymbol{\nu}}^*)$, at each iteration.

To highlight the necessity of properly considering stochasticity to generate accurate guarantees, we compared performance while optimizing the PAC bounds with and without considering stochastic dynamics (replace line 3 of Algorithm 3 and line 9 of Algorithm 4 with $\mathbf{x}_{t+1} = \mathbf{x}_t + f(\mathbf{x}_t, \mathbf{u}_t)\Delta t$). Optimizing without stochastic dynamics caused $\widehat{\mathcal{J}}(\widehat{\boldsymbol{\nu}}^*)$ and $\widehat{\mathcal{C}}(\widehat{\boldsymbol{\nu}}^*)$ to be larger than the bounds, indicating that the guarantees were not accurate (Fig. 2(a), (b)). We also performed an ablation study in which we compared performance with and without feedback. On average, each iteration took 15.9 ms without feedback and 18.5 ms with feedback. Feedback enabled the optimizer to reduce the bounds to lower values (Fig. 2(e)–(h)) and resulted in a tighter distribution in state-space (Fig. 3).

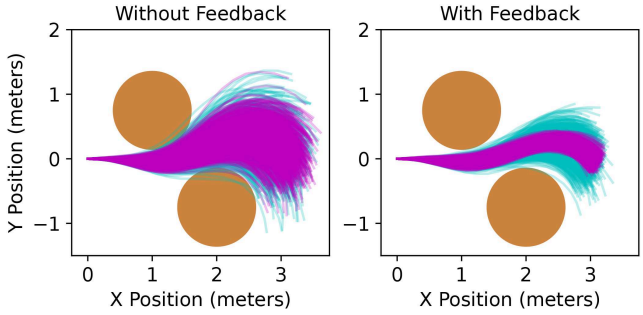

Fig. 3. Samples from optimized policy distribution and stochastic dynamics in cyan. Samples from mean of the optimized policy distribution and stochastic dynamics in magenta. Obstacles in brown.

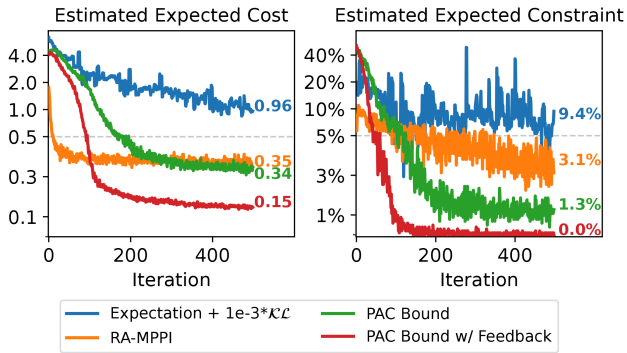

Fig. 4. Comparison between our approach and other methods.

We compare against optimizing an empirical approximation of the expected costs/constraints, regulated by the $\mathcal{KL}$-divergence to prior policies, to demonstrate that optimizing the PAC bounds encourages more robust polices. We also compare against RA-MPPI [30], a state-of-the-art risk-aware sampling based NMPC method. We used the following parameters: $\Sigma_\epsilon = 0.01$, $M = 1024$, $N = 300$, $\eta = 0$, $\alpha = 0.3$, $A = 10$, $B = 1$, $C_u = 0.5$ for the cost, and $C_u = 0.05$ for the constraint. Our approach converged to the lowest estimated expected cost and probability of collision (Fig. 4).

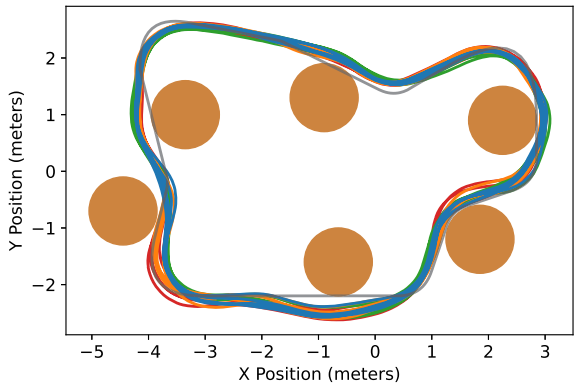

Fig. 5. Paths taken while following route (black) and avoiding obstacles (brown) in simulation. Blue/green paths use stochastic bounds, red/orange paths do not. Blue/orange paths use feedback, red/green paths do not.

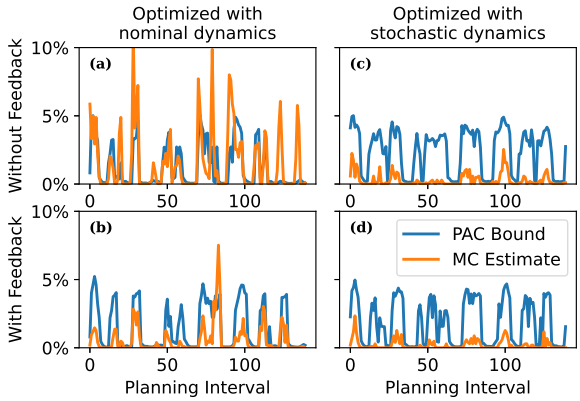

Fig. 6. Optimized PAC Bound compared to Monte Carlo estimates for probability of constraint violation at each planning interval during first laps. Variations in the PAC Bound and MC estimate over time are expected behavior due to the robot moving closer/farther from obstacles along the route.

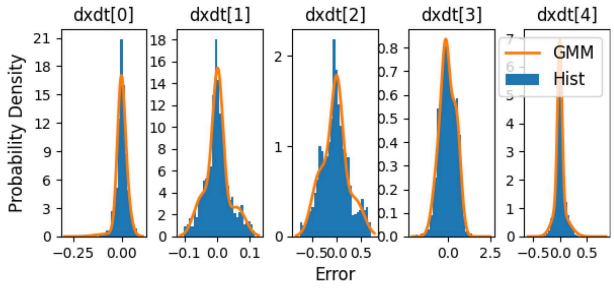

Fig. 7. Rallycar dynamics noise model.

B. NMPC Experiment

The system follows a receding horizon state generated on a looping path. The path runs along a set of circular obstacles. We optimize a 12 timestep trajectory with $\Delta t = 0.1$ second at a replanning period of $H = 0.2$ sec. We apply a quadratic cost on the final state of the trajectory: $J(\boldsymbol{\tau}) = \mathbf{x}_{N_T+1}^T \mathbf{Q}_f \mathbf{x}_{N_T+1}$ where $\mathbf{Q_f} = diag([1.0, 1.0, 0.1, 0.1, 0.0])$. Parameters were as follows: $L = 5$, $M = 1024$, $\delta = 0.05$, $\gamma = 10$. To allow for a higher control rate, the executed control trajectory is interpolated from 10 Hz to 50 Hz with the assumption that our PAC bounds will still hold. We perform an ablation study in which we compare performance with and without feedback, as well as optimizing the PAC bounds with and without stochastic dynamics.

PAC-NMPC was successfully able to control the system in real time and followed the path while avoiding nearby obstacles. In Fig. 5, we show the route taken by the system during five loops of the path.

During each planning interval of $H = 0.2$ sec, the controller achieved an average of 54 iterations of PAC stochastic trajectory optimization (approximately 3.7 ms per iteration). The controller consistently produced PAC bounds $\leq 5\%$ probability of collision, which accurately bounded the Monte Carlo estimates when optimizing with stochastic dynamics. The bound was exceeded only once out of 482 planning intervals. Thus, the estimates were bounded accurately in around 99.8% of planning intervals, well within the 95% confidence bound (Fig. 6(d)).

We demonstrate a quantitative difference in the observance of the PAC Bounds in Fig. 6. When optimizing the bounds with the nominal dynamics, the probability of collision estimates were only bounded in 93.9% of planning intervals (Fig. 6(b)). This decrease in bound accuracy is even more apparent when running the controller without feedback to compensate for unmodeled noise, resulting in only 64.7% of planning intervals having bounded probability of collision estimates (Fig. 6(a)).

VI. Hardware Experiments

A. Rally Car

1) Hardware: To evaluate our method on physical hardware, we ran PAC-NMPC on a 1/10th-scale Traxxas Rally Car platform. The algorithm runs on a Nvidia Jetson Orin mounted on the bottom platform. The control interface is a variable electronic speed controller (VESC), which executes commands and provides servo state information. The position and orientation is tracked with an OptiTrack system.

2) Setup: The route, virtual obstacle placements, and TVLQR costs are identical to those in the simulation experiments. We optimize a 12 timestep trajectory with $\Delta t = 0.1$ sec at a replanning period of $H = 0.2$ sec. We apply a quadratic cost to the final state of the trajectory: $J(\boldsymbol{\tau}) = \mathbf{x}_{N_T+1} \mathbf{Q}_f \mathbf{x}_{N_T+1}$ where $\mathbf{Q_f} = diag([1.0, 1.0, 0.4, 0.1, 0.0])$. Parameters were $L = 2$, $M = 1024$, $\delta = 0.05$, $\gamma = 10$.

We model the rally car dynamics as a stochastic bicycle model where noise is modeled as a 3 component Gaussian Mixture Model. This mixture model was fit from 2500 samples of data collected with the car (Fig. 7). We perform the same ablation study as done in the simulation experiments.

3) Results: PAC-NMPC ran onboard the rally car in real-time and followed the path while avoiding nearby obstacles. In Fig. 8, we display the path taken by the car with collisions annotated by arrows. During each planning interval of $H = 0.2$ sec, the controller achieved an average of 10 iterations of trajectory distribution optimization with approximately 20 ms per iteration.

The controller consistently produced PAC bounds $\leq 10\%$ probability of collision, which accurately bounded the Monte Carlo estimates of the probability of collision at all planning intervals (Fig. 9(d)). In this context, Monte Carlo estimates refer to sampled trajectories using the simulated stochastic dynamics model. When optimizing with stochastic dynamics and without feedback, the car collided with virtual obstacles twice, likely due to unmodeled dynamics not captured in the Gaussian mixture

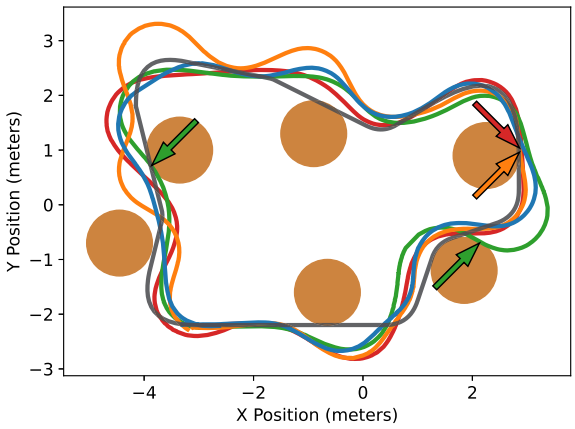

Fig. 8. Path taken while following route (black) and avoiding obstacles (brown) with rally car hardware. Blue/green paths use stochastic bounds, red/orange paths do not. Blue/orange paths use feedback, red/green paths do not. Collisions are annotated by arrows.

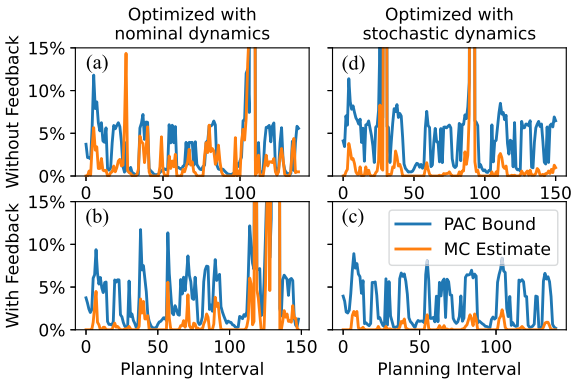

Fig. 9. Optimized PAC Bound compared to Monte Carlo estimates for probability of constraint violation at each planning interval.

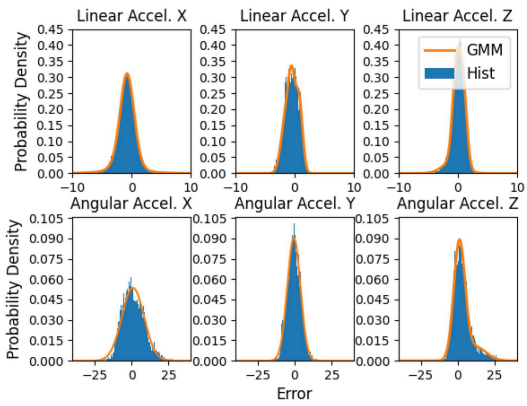

Fig. 10. Edge540 acceleration noise model.

model (Fig. 9(c)). When optimizing the bound with nominal dynamics, the car collided with virtual obstacles, with and without feedback (Fig. 9(a), (b)).

B. Fixed-Wing UAV

1) Hardware: To demonstrate that our approach can extend to more complex, high-dimensional systems, we use PAC-NMPC to control a 24" wingspan Edge 540 fixed-wing

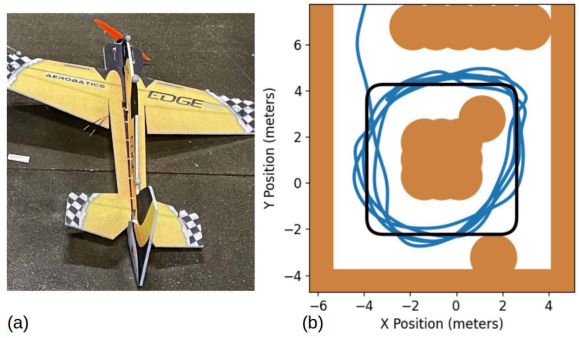

Fig. 11. (a) Edge540. (b) Path (blue) taken by the fixed-wing while following route (black) and avoiding obstacles (brown).

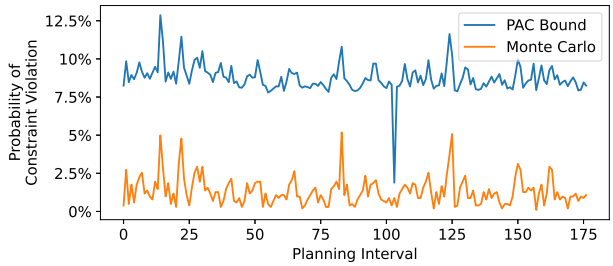

Fig. 12. Optimized PAC bound compared to Monte Carlo estimates for probability of constraint violation.

UAV. It has a controllable propeller, rudder, elevator, and kinematically linked ailerons. The position and orientation are tracked with an OptiTrack system.

2) Setup: We utilize a quaternion formulation of the fixed-wing described in [2] with RK2 integration. This system has a 17-dimensional state space and a 4-dimensional control space. The state of the system is $\mathbf{x} = [\mathbf{r}, \mathbf{q}, \boldsymbol{\delta}, \mathbf{v}, \boldsymbol{\omega}, p]$ where $\mathbf{r} \in \mathbb{R}^3$ is the position, $\mathbf{q} \in \mathbb{Q}$ is the orientation, $\boldsymbol{\delta} \in \mathbb{R}^3$ are control surface deflections of the ailerons, elevator, and rudder respectively, $\mathbf{v} \in \mathbb{R}^3$ is the linear velocity, $\boldsymbol{\omega} \in \mathbb{R}^3$ is the angular velocity, and p is the propeller speed. The control signal is the rate of change of the control surface deflections and of the propeller speed: $\mathbf{u} = [u_a, u_e, u_r, u_p]$. We place noise represented by a 3 component Gaussian Mixture Model over the body frame accelerations, which was fit from 2000 samples of data collected with the fixed-wing (Fig. 10). We optimize a 12 timestep trajectory with $\Delta t = 0.1$ sec at a replanning period of $H = 0.2$. We apply a quadratic cost of the form $J(\boldsymbol{\tau}) = \mathbf{x_{N_T+1}}^T \mathbf{Q_f} \mathbf{x_{N_T+1}} + \sum_{t=1}^{N_T} (\mathbf{x_t}^T \mathbf{Q} \mathbf{x_t} + \mathbf{u_t}^T \mathbf{R} \mathbf{u_t})$ and an obstacle constraint. Parameters were as follows: $L = 1, M = 1024, \delta = 0.05$, and $\gamma = 10$. We use feedback and use stochastic dynamics to optimize the PAC bounds. To enable real-time performance, we set the number of prior policies, L, to one and compute the linearized dynamics for the feedback policy using finite difference on the mean trajectory of the prior. The controller was run on a desktop computer with an AMD Ryzen 7 5800x and a NVIDIA GeForce RTX 3080.

3) Results: PAC-NMPC was successfully able to control the fixed-wing in real-time, following a square path while avoiding nearby obstacles. In Fig. 11, we display the route taken by the system during five loops of the path. During each planning interval of $H = 0.2$ sec, the controller achieved an average of 22 iterations of trajectory distribution optimization with around

8.5 ms per iteration. The controller consistently produced PAC bounds of around 10% probability of collision, which accurately bounded the Monte Carlo estimates (Fig. 12).

VII. Discussion

In this work, we presented a novel SNMPC method capable of propagating uncertainty through arbitrary nonlinear dynamic systems and of providing statistical guarantees on expected cost and constraint violations. We demonstrated real time performance both in simulation and on-board physical hardware. Further, we show that the algorithm is capable of scaling to more complex systems, like fixed-wing UAVs. Since this algorithm can be used with "black-box" sampling of dynamics (assuming they are continuously differentiable), costs, constraints, and noise, future work can investigate its use with learned dynamics and perception-informed costs. Future work could also explore the extension of this approach to more complex control trajectory distributions.

References

[1] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza, "PAMPC: Perception-aware model predictive control for quadrotors," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 1–8.

[2] M. Basescu and J. Moore, "Direct NMPC for post-stall motion planning with fixed-wing UAVs," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 9592–9598.

[3] Y. Ding, A. Pandala, and H.-W. Park, "Real-time model predictive control for versatile dynamic motions in quadrupedal robots," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2019, pp. 8484–8490.

[4] R. Grandia, F. Farshidian, R. Ranftl, and M. Hutter, "Feedback MPC for torque-controlled legged robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 4730–4737.

[5] J. A. Paulson and A. Mesbah, "An efficient method for stochastic optimal control with joint chance constraints for nonlinear systems," *Int. J. Robust Nonlinear Control*, vol. 29, no. 15, pp. 5017–5037, 2019.

[6] Z. Manchester and S. Kuindersma, "DIRTREL: Robust trajectory optimization with ellipsoidal disturbances and LQR feedback," in *Proc. IEEE Robot. Sci. Syst.*, 2017, doi: 10.15607/RSS.2017.XIII.057.

[7] S. Singh, A. Majumdar, J.-J. Slotine, and M. Pavone, "Robust online motion planning via contraction theory and convex optimization," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 5883–5890.

[8] N. Ozaki, S. Campagnola, and R. Funase, "Tube stochastic optimal control for nonlinear constrained trajectory optimization problems," *J. Guid. Control, Dyn.*, vol. 43, pp. 645–655, 2020.

[9] J. Yin, Z. Zhang, E. Theodorou, and P. Tsiotras, "Trajectory distribution control for model predictive path integral control using covariance steering," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2022, pp. 1478–1484.

[10] M. Kobilarov, "Sample complexity bounds for iterative stochastic policy optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 3114–3122.

[11] M. Sheckells, "Probably approximately correct robust policy search with applications to mobile robotics," Ph.D. dissertation, Johns Hopkins Univ., Baltimore, MD, USA, 2020.

[12] M. E. Villanueva, R. Quirynen, M. Diehl, B. Chachuat, and B. Houska, "Robust MPC via min–max differential inequalities," *Automatica*, vol. 77, pp. 311–321, 2017.

[13] G. Garimella, M. Sheckells, J. L. Moore, and M. Kobilarov, "Robust obstacle avoidance using tube NMPC," in *Proc. Robotics: Sci. Syst.*, 2018, doi: 10.15607/RSS.2018.XIV.055.

[14] S. Lucia, R. Paulen, and S. Engell, "Multi-stage nonlinear model predictive control with verified robust constraint satisfaction," in *Proc. IEEE 53rd Conf. Decis. Control*, 2014, pp. 2816–2821.

[15] D. Q. Mayne, E. C. Kerrigan, E. Van Wyk, and P. Falugi, "Tube-based robust nonlinear model predictive control," *Int. J. Robust Nonlinear Control*, vol. 21, no. 11, pp. 1341–1353, 2011.

[16] D. L. Marruedo, T. Alamo, and E. F. Camacho, "Input-to-state stable MPC for constrained discrete-time nonlinear systems with bounded additive uncertainties," in *Proc. IEEE 41st Conf. Decis. Control*, 2002, pp. 4619–4624.

[17] Y. Gao, A. Gray, H. E. Tseng, and F. Borrelli, "A tube-based robust nonlinear predictive control approach to semiautonomous ground vehicles," *Veh. Syst. Dyn.*, vol. 52, no. 6, pp. 802–823, 2014.

[18] F. Bayer, M. Bürger, and F. Allgöwer, "Discrete-time incremental ISS: A framework for robust NMPC," in *Proc. Eur. Control Conf.*, 2013, pp. 2068–2073.

[19] G. Chou, N. Ozay, and D. Berenson, "Model error propagation via learned contraction metrics for safe feedback motion planning of unknown systems," in *Proc. IEEE 60th Conf. Decis. Control*, 2021, pp. 3576–3583.

[20] J. Köhler, R. Soloperto, M. A. Müller, and F. Allgöwer, "A computationally efficient robust model predictive control framework for uncertain nonlinear systems," *IEEE Trans. Autom. Control*, vol. 66, no. 2, pp. 794–801, Feb. 2021.

[21] T. Lew and M. Pavone, "Sampling-based reachability analysis: A random set theory approach with adversarial sampling," in *Proc. Conf. Robot Learn.*, 2021, pp. 2055–2070.

[22] A. Wu, T. Lew, K. Solovey, E. Schmerling, and M. Pavone, "Robust-RRT: Probabilistically-complete motion planning for uncertain nonlinear systems," in *Proc. Int. Symp. Robot. Res.*, 2023, pp. 538–554.

[23] E. Todorov and W. Li, "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems," in *Proc. Amer. Control Conf.*, 2005, pp. 300–306.

[24] E. Todorov and Y. Tassa, "Iterative local dynamic programming," in *Proc. IEEE Symp. Adaptive Dyn. Program. Reinforcement Learn.*, 2009, pp. 90–95.

[25] E. Theodorou, Y. Tassa, and E. Todorov, "Stochastic differential dynamic programming," in *Proc. IEEE Amer. Control Conf.*, 2010, pp. 1125–1132.

[26] T. A. Howell, C. Fu, and Z. Manchester, "Direct policy optimization using deterministic sampling and collocation," *IEEE Robot. Automat. Lett.*, vol. 6, no. 3, pp. 5324–5331, Jul. 2021.

[27] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 1433–1440.

[28] G. Williams, B. Goldfain, P. Drews, K. Saigol, J. M. Rehg, and E. A. Theodorou, "Robust sampling based model predictive control with sparse objective information," in *Proc. Robot. Sci. Syst.*, 2018, doi: 10.15607/RSS.2018.XIV.042.

[29] M. S. Gandhi, B. Vlahov, J. Gibson, G. Williams, and E. A. Theodorou, "Robust model predictive path integral control: Analysis and performance guarantees," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 1423–1430, Apr. 2021.

[30] J. Yin, Z. Zhang, and P. Tsiotras, "Risk-aware model predictive path integral control using conditional value-at-risk," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 7937–7943.

[31] M. P. Deisenroth et al., "A survey on policy search for robotics," *Found. Trends Robot.*, vol. 2, no. 1/2, pp. 1–142, 2013.

[32] A. Majumdar, A. Farid, and A. Sonar, "PAC-Bayes control: Learning policies that provably generalize to novel environments," *Int. J. Robot. Res.*, vol. 40, no. 2/3, pp. 574–593, 2021.

[33] S. Veer and A. Majumdar, "Probably approximately correct vision-based planning using motion primitives," in *Proc. Conf. Robot Learn.*, 2021, pp. 1001–1014.

[34] J. Peters, K. Mulling, and Y. Altun, "Relative entropy policy search," in *Proc. AAAI Conf. Artif. Intell.*, 2010, pp. 1607–1612.

[35] O. Catoni, "Challenging the empirical mean and empirical variance: A deviation study," *Annales de l'IHP Probabilité s et statistiques*, vol. 48, no. 4, pp. 1148–1185, 2012.

[36] Y. Fei, G. Rong, B. Wang, and W. Wang, "Parallel l-BFGS-B algorithm on GPU," *Comput. Graph.*, vol. 40, pp. 1–9, 2014.

[37] R. Tedrake, "Underactuated robotics," 2022. [Online]. Available: http://underactuated.mit.edu