2024 IEEE 63rd Conference on Decision and Control (CDC)
December 16-19, 2024. MiCo, Milan, Italy

Towards safe and tractable Gaussian process-based MPC: Efficient sampling within a sequential quadratic programming framework

Manish Prajapat*, Amon Lahr*, Johannes Köhler, Andreas Krause, Melanie N. Zeilinger

***Abstract*— Learning uncertain dynamics models using Gaussian process (GP) regression has been demonstrated to enable high-performance and safety-aware control strategies for challenging real-world applications. Yet, for computational tractability, most approaches for Gaussian process-based model predictive control (GP-MPC) are based on approximations of the reachable set that are either overly conservative or impede the controller's safety guarantees. To address these challenges, we propose a robust GP-MPC formulation that guarantees constraint satisfaction with high probability. For its tractable implementation, we propose a sampling-based GP-MPC approach that iteratively generates consistent dynamics samples from the GP within a sequential quadratic programming framework. We highlight the improved reachable set approximation compared to existing methods, as well as real-time feasible computation times, using two numerical examples.**

I. Introduction

Gaussian process (GP) regression [1] offers versatile representation capabilities and inherent uncertainty quantification for learning dynamics models. Combined with model predictive control (MPC) [2], GP models have demonstrated to be highly effective at improving the control performance while ensuring safety-awareness with respect to the remaining model uncertainty, across a diverse range of challenging real-world scenarios [3]–[8]. Yet, propagating the uncertainty within the resulting Gaussian process state-space model (GP-SSM) is non-trivial and remains a major bottleneck for safety-critical GP-MPC applications [9]. To remedy this issue, a standard approach [6]–[8], [10] is to use a linearization-based approximation of the probabilistic reachable set induced by the uncertain GP model [11]. However, in addition to the linearization error, this approximation is based on a strong independence assumption between the GP predictions at subsequent time steps [12]. Similarly, approximate uncertainty propagation methods, such as exact moment-matching [13] with a spectral GP approximation [14], or unscented filtering [3], [15], make it difficult to guarantee safety for the resulting GP-MPC formulations.

To establish constraint-satisfaction guarantees for GP-MPC, stochastic and robust MPC formulations have been proposed. In [16], a robust MPC approach has been suggested to iteratively over-approximate the uncertain prediction based on high-probability confidence regions of the GP. Yet, this approach tends to be overly conservative due to the ellipsoidal over-approximation based on global Lipschitz bounds, as well as robustification of the control input against the worst-case GP estimate at each time step *independently*, without taking into account that the uncertainty is of epistemic nature, i.e., that the true function does not change over the prediction horizon. To remedy this conservatism, [17] employs a Gaussian-process multi-step predictor; however, learning a multi-step predictor constitutes a more nonlinear and higher-dimensional regression problem, with stronger computational requirements, compared to single-step models.

For stochastic nonlinear MPC, it has been proposed to sample controlled trajectories of the GP-SSM offline to determine fixed constraint tightenings for online use, ensuring the desired probability of closed-loop chance-constraint satisfaction [18], [19]. Therein, a "forward-sampling" [20], [21] strategy, based on reconditioning the GP on previously sampled values, has been employed to overcome the difficulty of sampling continuous functions from the GP directly, which has also been used in [12], [22], [23] for trajectory predictions with GPs. However, the guarantees only hold for a fixed (distribution of) initial conditions and controller parameters.

To summarize, existing approximations of the reachable set associated with uncertain GP models either do not ensure closed-loop constraint satisfaction, or are too conservative to be practically useful.

Contributions: Toward addressing these challenges, we propose a GP-based MPC formulation that is robust against all dynamics in a confidence set, guaranteeing constraint satisfaction for the true system with high probability (Sec. III).

Additionally, we present an efficient sampling-based algorithm to approximately solve the derived robust GP-MPC problem based on sequential quadratic programming (SQP). The proposed strategy is equivalent to solving an MPC problem with multiple dynamics sampled directly from the GP posterior, leading to a more accurate uncertainty propagation compared to [7], [16]. Implementation is non-trivial since sampling the exact continuous function from GP is computationally intractable [24]. To tackle this, we jointly model the unknown dynamics and its Jacobian using a multivariate GP and construct *consistent* dynamics samples within the SQP framework by forward-sampling [12], [20]–[23] across SQP iterations. The sampling process is parallelized across all samples and GP dimensions, resulting in efficient run time on GPUs. We discuss how to apply the algorithm efficiently in receding horizon using the real-time iteration [25], ensuring a maintainable computational footprint and similarity of the sampled dynamics across MPC iterations (Section IV).

*First author, equal contribution. E-mail correspondence to: manishp@ai.ethz.ch; amlahr@ethz.ch. All authors are from ETH Zürich. Manish Prajapat is supported by ETH AI center, Amon Lahr by the European Union's Horizon 2020 research and innovation programme, Marie Skłodowska-Curie grant agreement No. 953348, ELO-X, and Johannes Köhler by the Swiss National Science Foundation under NCCR Automation, grant agreement 51NF40 180545.

We highlight the effectiveness of sampling-based uncertainty propagation in comparison with sequential over-approximation [16] and linearization-based [7] approaches on a pendulum example. Lastly, we further demonstrate our method with a more realistic car model (Section V).

II. Problem statement

We consider the task of state-feedback control for a discrete-time, nonlinear dynamical system

$$x(k+1) = f(x(k), u(k)) + B_d g^{\mathrm{tr}}(x(k), u(k)) \tag{1}$$

with state $x(k) \in \mathbb{R}^{n_x}$ and input $u(k) \in \mathbb{R}^{n_u}$. Here, $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ denotes the known part of the true dynamics, e.g., derived from first principles, and $g^{\mathrm{tr}} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_g}$, the unknown part to be estimated from data, modeled in a subspace defined by $B_d \in \mathbb{R}^{n_x \times n_g}$ with full column-rank. The data consists of $D \in \mathbb{N}$ noisy measurements of the one-step prediction error

$$y_k \doteq B_d^{\dagger}(x(k+1) - f(x(k), u(k))) + \epsilon(k), \tag{2}$$

where $\epsilon(k)$ is independent and identically distributed (i.i.d.) Gaussian measurement noise with covariance $\Lambda \doteq \lambda^2 I_{n_g}$, $\lambda > 0$, and $(\cdot)^{\dagger}$ denotes the Moore-Penrose pseudo-inverse. For notational simplicity, and without loss of generality, the results in this paper are presented for a scalar unknown dynamics component, i.e., $n_g = 1$; see Remark 2 for an extension to the multivariate case.

The main objective is the design of a model predictive control (MPC) strategy that ensures satisfaction of safety constraints

$$(x(k), u(k)) \in \mathcal{Z} \doteq \{(x, u) \mid h(x, u) \leq 0\} \tag{3}$$

at all times $k \in \mathbb{N}$, despite the epistemic uncertainty in g.

We make the following regularity assumption.

Assumption 1 (Regularity [26])**.** *The uncertain dynamics g^{tr} is an element of the Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}_k$ associated with the continuous, positive definite kernel $k : \mathbb{R}^{n_z} \times \mathbb{R}^{n_z} \to \mathbb{R}$, with a bounded RKHS norm, i.e., $g^{\mathrm{tr}} \in \mathcal{H}_k$ with $\|g^{\mathrm{tr}}\|_{\mathcal{H}_k} \leq B_g < \infty$.*

To obtain a data-driven model and uncertainty quantification of the unknown dynamics g^{tr}, Gaussian process (GP) regression is employed. Denote by $\bar{\mathcal{D}} \doteq \{Z, Y\}$ the data set, where $Z = \begin{bmatrix} z_1, \dots, z_D \end{bmatrix}$, $z_k \doteq (x(k), u(k)) \in \mathbb{R}^{n_z}$, are the training inputs with $n_z = n_x + n_u$; $Y = \begin{bmatrix} y_1, \dots, y_D \end{bmatrix}$, the training targets; and $Z_\star = \begin{bmatrix} z_1^\star, \dots, z_{n_\star}^\star \end{bmatrix}$, the test inputs. Then, the posterior mean and covariance are given as

$$\mu(Z_\star) = k(Z_\star, Z)\tilde{K}_{ZZ}^{-1} Y, \tag{4}$$

$$\sigma^2(Z_\star, Z'_\star) = k(Z_\star, Z'_\star) - k(Z_\star, Z)\tilde{K}_{ZZ}^{-1} k(Z, Z'_\star), \tag{5}$$

respectively, where $[k(Z, Z')]_{ij} \doteq k(z_i, z'_j)$ for any matrices Z, Z' with respective column vectors $z_i, z'_j \in \mathbb{R}^{n_z}$, and $\tilde{K}_{ZZ} \doteq k(Z, Z) + \lambda^2 I_D$.

Assumption 1 enables the following high-probability bounds for the unknown part of the dynamics, which has also been utilized in prior work for robust GP-MPC [16].

Lemma 1. ([26, Theorem 2]) *Let Assumption 1 hold and let $\sqrt{\beta} = B_g + 4\lambda\sqrt{\gamma_D + 1 + \log(1/p)}$, where γ_D is the maximum information gain as defined in [26, Sec. 3.1]. Then, for all $z \in \mathcal{Z}$, with probability at least $1 - p$, it holds that*

$$|g^{\mathrm{tr}}(z) - \mu(z)| \leq \sqrt{\beta}\sigma(z).$$

Using Lemma 1, we can construct high-probability upper and lower confidence bounds $\underline{g}(z) \doteq \mu(z) - \sqrt{\beta}\sigma(z)$, $\overline{g}(z) \doteq \mu(z) + \sqrt{\beta}\sigma(z)$, respectively, such that we can guarantee that the unknown function g^{tr} is contained in the set

$$\mathcal{G} \coloneqq \left\{ g \mid \underline{g}(z) \leq g(z) \leq \overline{g}(z)\ \forall z \in \mathcal{Z} \right\} \tag{6}$$

with probability at least $1 - p$, i.e., $\Pr(g^{\mathrm{tr}} \in \mathcal{G}) \geq 1 - p$.

To this end, we denote by $\mathcal{GP}_{[\underline{g}, \overline{g}]}(0, k; \bar{\mathcal{D}})$ the truncated distribution of functions $g \sim \mathcal{GP}(0, k; \bar{\mathcal{D}})$ that are contained in the set $g \in \mathcal{G}$. For a finite number of inputs $Z_\star \in \mathbb{R}^{n_z \times n_\star}$, the values $G_\star$ of a function $g \sim \mathcal{GP}_{[\underline{g}, \overline{g}]}(0, k; \bar{\mathcal{D}})$ are thus distributed according to the $n_\star$-dimensional *truncated* Gaussian distribution $\mathcal{N}_{[\underline{g}, \overline{g}]}(G_\star; \mu(Z_\star), \Sigma(Z_\star))$ on the hyper-rectangle $[\underline{g}(z_1^\star), \overline{g}(z_1^\star)] \times \ldots \times [\underline{g}(z_{n_\star}^\star), \overline{g}(z_{n_\star}^\star)]$.

Remark 1. *The measurement noise assumption can be relaxed to conditionally σ-sub-Gaussian noise, see, e.g., [26].*

Remark 2. *Note that Assumption 1 and Lemma 1 may easily be transferred to the multidimensional case when each component of g is modeled as an independent, single-dimensional GP.*

Remark 3. *In this work, we focus on epistemic uncertainty introduced by the missing knowledge about g. As such, we consider noise-free true dynamics* (1) *with exact state measurements, while allowing for measurement noise in the data set $\bar{\mathcal{D}}$ collected offline. For the special case of noise-free measurements, tighter bounds than Lemma 1 for the unknown dynamics g may be employed, see, e.g., [27, Sec. 14.1].*

III. Robust GP-MPC problem

In this section, we formulate a robust GP-MPC problem using the derived dynamics distribution, which provides safety guarantees for the unknown system. The proposed MPC problem at time $k \in \mathbb{N}$ is given by

$$\min_{\boldsymbol{u}} \quad \mathbb{E}_{g \sim \mathcal{GP}_{[\underline{g}, \overline{g}]}(0, k; \bar{\mathcal{D}})} \left[\sum_{i=0}^{H-1} l_i(x_i^g, u_i) \right] \tag{7a}$$

$$\text{s.t.} \quad \forall i \in \{0, \dots, H-1\}, \forall g \sim \mathcal{GP}_{[\underline{g}, \overline{g}]}(0, k; \bar{\mathcal{D}}), \tag{7b}$$

$$x_0^g = x(k), \tag{7c}$$

$$x_{i+1}^g = f(x_i^g, u_i) + B_d g(x_i^g, u_i), \tag{7d}$$

$$(x_i^g, u_i) \in \mathcal{Z}, \tag{7e}$$

$$x_H^g \in \mathcal{Z}_{\mathrm{f}}. \tag{7f}$$

Here, Eq. (7d) yields a predicted state sequence $\boldsymbol{x}^g = (x_0^g, \dots, x_H^g)$ for any uncertain dynamics instance $g \sim \mathcal{GP}_{[\underline{g}, \overline{g}]}(0, k; \bar{\mathcal{D}})$ under the same, shared control sequence $\boldsymbol{u} = (u_0, \dots, u_{H-1})$ along the prediction horizon H. Eq. (7e) ensures that all the predicted sequences satisfy the joint state and input constraints (3).

The constraint in Eq. (7c) initializes all the trajectories to the system state at time k, denoted by $x(k)$, and Eq. (7f) ensures that all the predicted trajectories reach the terminal set

$$\mathcal{Z}_{\mathrm{f}} \doteq \{x \mid h_H(x) \leq 0\} \tag{8}$$

at the end of the horizon. The cost in Eq. (7a) reflects the expected finite-horizon cost over functions drawn from $g \sim \mathcal{GP}_{[\underline{g},\overline{g}]}(0, k; \bar{\mathcal{D}})$, with stage cost $l_i : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$. Note that a terminal penalty can also be included in the cost.

Next, we investigate the theoretical guarantees of the proposed robust GP-MPC problem (7). In order to address constraint satisfaction for infinite time we make the following standard assumption on the terminal invariant set.

Assumption 2 (Robust positive invariant set)**.** *The terminal set $\mathcal{Z}_{\mathrm{f}}$ is a robust positive invariant set for system* (1) *under the input $u_{\mathrm{f}} \in \mathbb{R}^m$, i.e., $\forall g \in \mathcal{G}$, $x \in \mathcal{Z}_{\mathrm{f}}$: $(x, u_{\mathrm{f}}) \in \mathcal{Z}$ and $f(x, u_{\mathrm{f}}) + B_d g(x, u_{\mathrm{f}}) \in \mathcal{Z}_{\mathrm{f}}$.*

Similar to standard MPC designs [2, Sec. 2.5.5], this condition can, e.g., be satisfied by constructing $\mathcal{Z}_{\mathrm{f}}$ based on a local Lyapunov function around the origin.[1] The following theorem guarantees constraint satisfaction by applying the optimal control sequence obtained by solving problem (7).

Theorem 2. *Let Assumptions 1 and 2 hold. Let Problem* (7) *be feasible at $k = 0$ and $\boldsymbol{u}^\star = \{u_0^\star, \dots, u_{H-1}^\star\}$ be the corresponding optimal control sequence. Then, with probability at least $1 - p$, applying the control sequence*

$$u(k) = \begin{cases} u_k^\star, & k \in \{0, \dots, H-1\} \\ u_{\mathrm{f}}, & k \geq H \end{cases}$$

to the system (1) *leads to constraint satisfaction for all times, i.e., $(x(k), u(k)) \in \mathcal{Z} \quad \forall k \in \mathbb{N}$.*

Proof. First, note that $g^{\mathrm{tr}} \in \mathcal{G}$ with probability at least $1 - p$ due to Lemma 1. Thus, for $g^{\mathrm{tr}} \in \mathcal{G}$ and $k \in \{0, \dots, H-1\}$, constraint satisfaction directly follows from feasibility of Problem (7), for $k \geq H$, from Assumption 2. □

While Problem (7) is in general not recursively feasible, closed-loop constraint satisfaction can be ensured using an implementation strategy similar to [29, Alg. 1]. In the remainder of this paper, we focus on a computationally efficient, sampling-based approximation of (7); further closed-loop considerations are left for future work.

IV. Efficient sampling-based GP-MPC

Solving problem (7) is intractable due to the infinite number of dynamics functions in the set. In this section, we obtain a tractable approximation by solving it using a finite number of functions. To solve the nonlinear program iteratively, we employ SQP (Section IV-A). During optimization, the required *function and derivative* values are thereby provided by modeling the unknown function g^{tr} and its Jacobian *jointly*, using a multivariate GP with a derivative kernel (Section IV-B). As it is generally not possible to obtain analytical expressions for the exact infinite-dimensional GP sample paths, we utilize a sequential GP sampling approach [20], [21], which is based on conditioning the GP on previously sampled values, and integrate it into the SQP framework (Section IV-C). Finally, we discuss the properties of the proposed SQP algorithm, as well as its efficient closed-loop implementation (Section IV-D).

A. Sequential Quadratic Programming

In the following, we describe how to solve (7) for a finite number of N samples using SQP [30, Sec. 18]. Therefore, let the functions f, g^{tr}, h, h_H, l_i and k be twice continuously differentiable. A local optimizer can be obtained by iteratively approximating the nonlinear program (NLP) with a quadratic program (QP)

$$\begin{aligned}
\min_{\Delta \boldsymbol{u}, \Delta \boldsymbol{x}} \quad & \frac{1}{N} \sum_{n=1}^{N} \sum_{i=0}^{H-1} \begin{bmatrix} \Delta x_i^n \\ \Delta u_i \end{bmatrix}^\top H_i^n \begin{bmatrix} \Delta x_i^n \\ \Delta u_i \end{bmatrix} + (q_i^n)^\top \begin{bmatrix} \Delta x_i^n \\ \Delta u_i \end{bmatrix} \\
\text{s.t.} \quad & \forall i \in \{0, \dots, H-1\}, \forall n \in \{1, \dots, N\}, && \text{(9a)} \\
& \Delta x_0^n = 0, && \text{(9b)} \\
& \Delta x_{i+1}^n = f(\hat{x}_i^n, \hat{u}_i) + B_d g^n(\hat{x}_i^n, \hat{u}_i) - \hat{x}_{i+1}^n \\
& \qquad + \hat{A}_i^n \Delta x_i^n + \hat{B}_i^n \Delta u_i, && \text{(9c)} \\
& 0 \geq h(\hat{x}_i^n, \hat{u}_i) + \hat{H}_{x,i}^x \Delta x_i^n + \hat{H}_{u,i}^u \Delta u_i^n, && \text{(9d)} \\
& 0 \geq h_H(\hat{x}_H^n) + \hat{H}_{x,H}^x \Delta x_H^n. && \text{(9e)}
\end{aligned}$$

Thereby, H_i^n denotes a positive definite approximation of the Hessian of the Lagrangian, q_i^n, the cost gradient, and

$$\hat{A}_i^n = \frac{\partial (f + B_d g^n)}{\partial x}, \quad \hat{B}_i^n = \frac{\partial (f + B_d g^n)}{\partial u} \tag{10}$$

$$\hat{H}_{x,i}^n = \frac{\partial h}{\partial x}, \quad \hat{H}_{u,i}^n = \frac{\partial h}{\partial u}, \quad \hat{H}_{x,H}^n = \frac{\partial h_H}{\partial x}, \tag{11}$$

the Jacobians of the dynamics and other (in-)equality constraints with respect to the states and inputs, respectively, each evaluated at $(\hat{x}_i^n, \hat{u}_i)$. The solution of (9) in terms of inputs $\Delta \boldsymbol{u} = (\Delta u_0, \dots, \Delta u_{H-1})$ and states $\Delta \boldsymbol{x} = \{\Delta \boldsymbol{x}^n\}_{n=1}^N$, with $\Delta \boldsymbol{x}^n = (\Delta x_0^n, \dots, \Delta x_H^n)$, is then used to update the current solution estimate $\hat{\boldsymbol{x}}^n = (\hat{x}_0^n, \dots, \hat{x}_H^n)$, $\hat{\boldsymbol{u}} = (\hat{u}_0, \dots, \hat{u}_{H-1})$, for the next iteration, $\hat{\boldsymbol{x}}^n \leftarrow \hat{\boldsymbol{x}}^n + \Delta \boldsymbol{x}^n$, $\hat{\boldsymbol{u}} \leftarrow \hat{\boldsymbol{u}} + \Delta \boldsymbol{u}$, until convergence.

Evaluation of (10) requires obtaining the values of the sampled functions g^n and their respective Jacobians, i.e.,

$$g^n(\hat{x}_i^n, \hat{u}_i), \qquad \frac{\partial g^n}{\partial x}(\hat{x}_i^n, \hat{u}_i), \qquad \frac{\partial g^n}{\partial u}(\hat{x}_i^n, \hat{u}_i), \tag{12}$$

at the current solution estimate $(\hat{x}_i^n, \hat{u}_i)$. In the following, we discuss how to efficiently draw these function and Jacobian values jointly from a Gaussian process.

B. Gaussian processes with derivatives

To sample function values and derivatives jointly that are consistent with sample paths of $\mathcal{GP}_{[\underline{g},\overline{g}]}(0, k; \bar{\mathcal{D}})$, we model the function g^{tr} and its spatial derivatives as a

[1] In particular, this requires that the origin is an equilibrium point $\forall g \in \mathcal{G}$, i.e., $f(0,0) + B_d g(0,0) = 0$, and that the linearizations at the origin are open-loop stable with a common Lyapunov function. The assumption of open-loop stability can be relaxed by using a feedback $u(x) = \kappa(x) + v$ in the predictions in (7) instead of open-loop predictions (cf., e.g., [28]).

multi-output GP, with the matrix-valued kernel function $k_{\mathrm{d}} : \mathbb{R}^{n_z} \times \mathbb{R}^{n_z} \to \mathbb{R}^{(n_x+1)\times(n_x+1)}$,

$$k_{\mathrm{d}}(z_a, z_b) = \begin{bmatrix} k(z_a, z_b) & \frac{\partial k(z_a,z_b)}{\partial z_a} \\ \frac{\partial k(z_a,z_b)}{\partial z_b}^\top & \frac{\partial^2 k(z_a,z_b)}{\partial z_a \partial z_b} \end{bmatrix}, \tag{13}$$

see, e.g., [1, Sec. 9.4]. To simplify notation, note that we have dropped the sample index $(\cdot)^n$ in both input and output arguments.

Given the initial data set $\bar{\mathcal{D}}$ in terms of only the function values and no derivative measurements, the GP with derivatives is conditioned on partial measurements. The inference formulas are easily derived starting from the standard GP regression case of full function-value and derivative measurements, see, e.g. [1, Sec. 2]: Let the partial data set be obtained by a linear projection with a data selection matrix $P \in \mathbb{N}^{(n_x+1)D\times D}$, such that $Y = P^\top Y_{\mathrm{d}}$, where $Y_{\mathrm{d}} \in \mathbb{R}^{(n_x+1)D}$ is the (unavailable) data set covering all output dimensions (including derivative data for g^{tr}). Then, the posterior mean and covariance of $\mathcal{GP}(0, k_{\mathrm{d}})$ conditioned on Y, evaluated at $n_\star$ test inputs $Z_\star \in \mathbb{R}^{n_z\times n_\star}$, $Z_\star = \begin{bmatrix} z_1, \dots, z_{n_\star} \end{bmatrix}$, are given as

$$\mu_{\mathrm{d}}(Z_\star) = k_{\mathrm{d}}(Z_\star, Z) P \tilde{K}_{ZZ}^{-1} Y, \tag{14}$$

$$\begin{aligned} \Sigma_{\mathrm{d}}(Z_\star, Z'_\star) = \dots \\ k_{\mathrm{d}}(Z_\star, Z'_\star) - k_{\mathrm{d}}(Z_\star, Z) P \tilde{K}_{ZZ}^{-1} P^\top k_{\mathrm{d}}(Z, Z'_\star), \end{aligned} \tag{15}$$

see, e.g., [1, Sec. A.2], where $[k_{\mathrm{d}}(Z, Z')]_{ij} \doteq k_{\mathrm{d}}(z_i, z'_j)$ for any matrices Z, Z' with respective column vectors $z_i, z'_j \in \mathbb{R}^{n_z}$, and $\tilde{K}_{ZZ} = P^\top k_{\mathrm{d}}(Z, Z) P + \lambda^2 I_D$ is the same covariance matrix as in Eq. (5). Note that the posterior mean $\mu(Z_\star)$ and covariance $\Sigma(Z_\star)$ of the original GP are recovered by projecting $\mu(Z_\star) = P_\star^\top \mu_{\mathrm{d}}(Z_\star)$, $\Sigma(Z_\star) = P_\star^\top \Sigma_{\mathrm{d}}(Z_\star) P_\star$, respectively, with an output selection matrix $P_\star \in \mathbb{N}^{n_\star(n_x+1)\times n_\star}$; hence, it is easily verified that the predictions of the GP function *values* with $\mathcal{GP}(0, k_{\mathrm{d}})$ and $\mathcal{GP}(0, k)$, given only function-value measurements Y, are identical.

C. Sequential GP sampling

Using the derivative kernel, we can jointly draw consistent function and derivative samples from the GP posterior by computing [1, Sec. A.2]

$$\begin{aligned} G_\star &\doteq \left[\dots, g(z_i), \frac{\partial g}{\partial z}(z_i), \dots \right]^\top \\ &= \mu_{\mathrm{d}}(Z_\star) + \sqrt{\Sigma_{\mathrm{d}}(Z_\star, Z_\star)}\eta, \end{aligned} \tag{16}$$

where $\eta \sim \mathcal{N}(0, I_{n_\star})$ and $\sqrt{\Sigma(Z_\star, Z_\star)}$ denotes the square-root of the GP posterior covariance matrix. To ensure that the component corresponding to the function values is sampled from the truncated Gaussian distribution $\mathcal{N}_{[\underline{g},\overline{g}]}(\mu(Z_\star), \sigma^2(Z_\star); \bar{\mathcal{D}})$, we thereby discard all samples whose value-component violates the confidence bounds in Lemma 1 at any input location[2].

[2]We analogously denote by $\mathcal{N}_{[\underline{g},\overline{g}]}(\mu_{\mathrm{d}}(Z_\star), \Sigma_{\mathrm{d}}(Z_\star))$ the truncated Gaussian distribution for the function value and Jacobian, where only the value-component is truncated.

Using (16), the values and derivatives of a function $g \sim \mathcal{GP}_{[\underline{g},\overline{g}]}(0, k, \bar{\mathcal{D}})$ can be obtained at a known set of input locations. However, during the SQP algorithm, the same function sample is to be evaluated at different input locations, whose value is only known at each respective iteration of the algorithm. To ensure consistency of the sampled function values across all SQP iterations, we thus employ the iterative sampling strategy proposed by [20, Sec. 3]. The sampling strategy is based on a simple equivalence relation, allowing to sample $G_\star = \begin{bmatrix} \dots, G_j, \dots \end{bmatrix}$ from $p(G_\star|\bar{\mathcal{D}}, Z_\star)$ at all input locations $Z_\star = \begin{bmatrix} \dots, Z_j, \dots \end{bmatrix}$ by iteratively sampling G_j from the conditional distributions $p(G_j|\mathcal{D}_{0:j-1}, Z_j)$, where $\mathcal{D}_0 \doteq \bar{\mathcal{D}}$ is the initial data set, and the sets $\mathcal{D}_j = \{G_j, Z_j\}$, $j = 1, \dots, L$, contain the previously sampled values at every iteration; see the illustration in Fig. 1. While this idea has also been applied in, e.g., [21, Sec. III.C], [22, Sec. IV.B], for completeness, we state this equivalence again in the following lemma.

Lemma 3. (cf. [23, Property 1]) *Let $p(G_\star|\bar{\mathcal{D}}, Z_\star) = \mathcal{N}_{[\underline{g},\overline{g}]}(G_\star; \mu(Z_\star), \Sigma(Z_\star))$ be the truncated Gaussian posterior distribution of a Gaussian process $\mathcal{GP}(0, k; \mathcal{D}_0)$, conditioned on the data set $\mathcal{D}_0 \doteq \bar{\mathcal{D}}$, and evaluated at L sets of input locations Z_j, $j = 1, \dots, L$, with $Z_\star = \begin{bmatrix} Z_1, \dots, Z_L \end{bmatrix} \in \mathbb{R}^{n_z\times HL}$. It holds that*

$$p(G_\star|\bar{\mathcal{D}}, Z_\star) = \prod_{j=1}^{L} p(G_j|\mathcal{D}_{0:j-1}, Z_j). \tag{17}$$

Proof. By the definition of the conditional probability distribution, it follows that the joint likelihood can be factorized, i.e.,

$$\begin{aligned} p(G_\star|\mathcal{D}_0, Z_\star) &= p(G_{2:L}|\mathcal{D}_0, G_1, Z_\star) p(G_1|\mathcal{D}_0, Z_\star) \\ &= p(G_{2:L}|\mathcal{D}_{0:1}, Z_{2:L}) p(G_1|\mathcal{D}_0, Z_1) \\ &= \dots = \prod_{j=1}^{L} p(G_j|\mathcal{D}_{0:j-1}, Z_j). \end{aligned}$$

□

D. Discussion of Sampling-based GP-MPC

The sampling-based SQP algorithm is summarized in Algorithm 1. In the following, we comment on the algorithm's convergence and its properties at convergence. In particular, we show that, at convergence, the true reachable set is recovered as the number of samples goes to infinity. Last, we discuss its efficient receding-horizon implementation.

1) Convergence of the SQP algorithm: Applied to SQP, Lemma 3 allows us to obtain the required values and Jacobians of each sample in (12) while running the algorithm, *as if* we had sampled the infinite-dimensional function $g^n \sim \mathcal{GP}_{[\underline{g},\overline{g}]}(0, k, \bar{\mathcal{D}})$ beforehand to evaluate and differentiate it at the SQP iterates. This implies that the convergence of the SQP iterates towards a local minimizer of the NLP (7) is not affected, but rather follows standard arguments for Newton-type methods, see, e.g. [30, Sec. 18].

2) Simultaneous GP sampling and optimization: A key challenge for simulations with GP-SSMs using forward-sampling is that the GP needs to be sampled (and re-

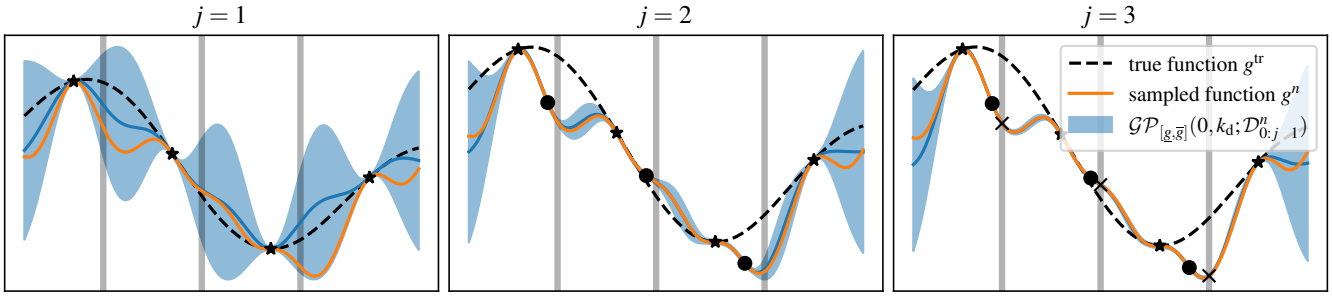


Fig. 1: Conditioning on the past SQP iterations to construct a sample from GP. In each SQP iteration j in Algorithm 1 (or forward-sampling iteration in Lemma 3), consistent function and derivative values of g^n are obtained by sampling from $\mathcal{GP}_{[\underline{g},\overline{g}]}(0, k_{\text{d}}; \mathcal{D}^n_{0:j-1})$. Initial training points are denoted by black stars; sampled values at iteration $j \in \{1, 2\}$, by {circles, crosses}, respectively; sampling locations, by gray vertical bars.

Algorithm 1: Sampling-based GP-MPC-SQP

input: initial guess $\hat{x}^n_i, \hat{u}_i$, data set $\mathcal{D}^n_0 \supseteq \bar{\mathcal{D}}$
for $j = 1, \dots L$ *SQP iterations* **do**
  `/* Preparation phase */`
  Set $\hat{Z}^n \leftarrow [\dots, (\hat{x}^n_i, \hat{u}_i), \dots]$;
  Sample $\hat{G}^n \sim \mathcal{N}_{[\underline{g},\overline{g}]}(\mu_{\text{d}}(\hat{Z}^n), \Sigma_{\text{d}}(\hat{Z}^n); \mathcal{D}^n_{0:j-1})$;
  Compute $\hat{A}^n_i, \hat{B}^n_i, \hat{H}^n_{x,i}, \hat{H}^n_{x,H}, \hat{H}^n_{u,i}, H^n_i, q^n_i$;
  Set $\mathcal{D}^n_j \leftarrow \left\{\hat{Z}^n, \hat{G}^n\right\}$;
  `/* Feedback phase */`
  Solve QP (9) for $\Delta \boldsymbol{x}, \Delta \boldsymbol{u}$;
  Set $\{\hat{\boldsymbol{x}}, \hat{\boldsymbol{u}}\} \leftarrow \{\hat{\boldsymbol{x}} + \Delta \boldsymbol{x}, \hat{\boldsymbol{u}} + \Delta \boldsymbol{u}\}$;
end
return $\hat{x}^n_i, \hat{u}_i, \mathcal{D}^n_{0:L}$

conditioned) *in sequence* along the simulation horizon. However, due to the proposed *simultaneous* optimization over state and input variables in the SQP algorithm, at each iteration, function and derivative values can efficiently be drawn jointly and in parallel based on the state and input values $(\hat{x}^n_i, \hat{u}_i)$ at the i-th prediction step. The functions $g^n \sim \mathcal{GP}_{[\underline{g},\overline{g}]}(0, k; \bar{\mathcal{D}})$ are thus efficiently sampled while the SQP algorithm iterates towards a feasible and locally optimal solution of the finite-sample problem (7). At convergence of the SQP algorithm, i.e., when $\Delta \boldsymbol{x} = 0$ and $\Delta \boldsymbol{u} = 0$ in (9), Lemma 3 ensures that the generated trajectories correspond to exact simulations of the GP-SSM dynamics.

Corollary 4. *Let $\hat{\boldsymbol{u}} = (\hat{u}_0, \dots, \hat{u}_{H-1})$, $\hat{\boldsymbol{x}}^n = (\hat{x}^n_0, \dots, \hat{x}^n_H)$ be an input and state sequence obtained by running Algorithm 1 until convergence. Then, $\hat{\boldsymbol{x}}^n$ is equal to the state sequence generated by $x^n_0 = \bar{x}_0$, $x^n_{i+1} = f(x^n_i, \hat{u}_i) + B_d g^n(x^n_i, \hat{u}_i)$, for $i = 0, \dots, H-1$, where $g^n \sim \mathcal{GP}_{[\underline{g},\overline{g}]}(0, k; \bar{\mathcal{D}})$.*

In addition to the joint sampling of the multivariate GP along the horizon, the sampling procedure can also be efficiently parallelized across the independent output dimensions and samples. By evaluating the $(N \times n_g)$ samples and output dimensions as a *batch* in the `GPyTorch` [31] framework, drastic speedups are achieved on supported parallel computing architectures.

3) Reachable set approximation: The next proposition establishes that, as the number of samples goes to infinity, at least one of the sampled function values $g^n(z)$ will be arbitrarily close to the true dynamics $g^{\text{tr}}(z)$, given that the true dynamics is contained within the uncertainty set $\mathcal{G}$.

Proposition 5. *Let Assumption 1 hold. Let $z \in \mathcal{Z}$ and $g^n(z) \sim \mathcal{N}_{[\underline{g},\overline{g}]}(G_\star; \mu(z), \sigma(z))$ where $n \in [1, N]$. For any $\epsilon > 0$, it holds that*

$$\lim_{N \to \infty} \Pr\left(\min_{n \in [1,N]} \left| g^{\text{tr}}(z) - g^n(z) \right| < \epsilon \right) \geq 1 - p. \quad (18)$$

Proof. Lemma 1 implies that $\Pr(g^{\text{tr}} \in \mathcal{G}) \geq 1 - p$. Now, assume that $g^{\text{tr}} \in \mathcal{G}$. Since $\mathcal{N}_{[\underline{g},\overline{g}]}(G_\star; \mu(z), \sigma(z))$ has full support on the domain $[\underline{g}(z), \overline{g}(z)]$, it holds that $p_{\text{in}} \doteq \Pr(|g^{\text{tr}}(z) - g^n(z)| < \epsilon) > 0$ for all $n \in [1, N]$. Therefore, for $N \to \infty$, the probability that no sample is ϵ-close to $g^{\text{tr}}(z)$ is $\lim_{N \to \infty}(1 - p_{\text{in}})^N = 0$. This implies that the joint probability of both events is greater or equal $1 - p$, which concludes the proof. □

Proposition 5 can be trivially extended to hold for any finite set of input locations. Combined with Corollary 4, this ensures that the sampling-based approximation convergences to the true reachable set as the number of samples increases (given a finite number of SQP steps).

4) Efficient receding-horizon implementation: Next, we discuss how to embed the proposed sampling strategy into a real-time SQP algorithm for a receding-horizon controller that solves (9) at each time step $k \in \mathbb{N}$, applying the first element of the optimized input sequence to the true system. Thereby, we focus on efficient real-time optimization rather than closed-loop guarantees, which are left for future work.

For real-time MPC implementations, running a fixed number of SQP iterations achieves an efficient trade-off between optimality of the current iterate and fast control feedback with respect to the currently processed initial condition [32, Sec. 4.1.2]. A particularly efficient trade-off is given by the Real-Time Iteration (RTI) [25], which runs a single SQP iteration per time step that is divided into a preparation phase, for computing the linearization around the current iterate, and a feedback phase, for solving the QP. With a sufficiently high sampling rate and regularity of the problem, this allows to continuously track the solution manifold of the finite-sample

Algorithm 2: Receding-horizon implementation

input: $x(0), \bar{\mathcal{D}}, \hat{x}_i^n, \hat{u}_i$
Initialize $\hat{x}_0^n \leftarrow x(0), \mathcal{D}^n \leftarrow \bar{\mathcal{D}}$;
for $k = 0, \dots$ **do**
  `/* Measure next state */`
  Set $x_0^n \leftarrow f(x(k), \hat{u}_0) + B_g g(x(k), \hat{u}_0)$;
  `/* Run SQP */`
  Set $(\hat{\boldsymbol{x}}^n, \hat{\boldsymbol{u}}, \mathcal{D}^n) \leftarrow$ **Algorithm 1**$(\hat{\boldsymbol{x}}^n, \hat{\boldsymbol{u}}, \mathcal{D}^n)$;
  $\rightarrow$ Apply $\hat{u}_0$ to (1) after `Feedback phase`;
  $\rightarrow$ Start `Preparation phase` for time $k+1$;
  `/* Keep most recent` $\tilde{L}$ `samples */`
  Set $\mathcal{D}^n \leftarrow \left\{\bar{\mathcal{D}}, \mathcal{D}^n_{L-\tilde{L}:L}\right\}$;
end

problem (7b) as it is evolving based on the changing initial condition, see, e.g. [33].

Still, even when running only a fixed number of SQP iterations per time step during closed-loop operation, storing all previously sampled values of the GP would lead to a steadily increasing solve time. Thus, to retain a fixed computational workload associated with GP sampling at each time step, we propose to discard a subset of the sampled data points at each MPC iteration, keeping only the most recent $\tilde{L} \leq L$ sampled values[3] generated at each call of Algorithm 1. The complete receding-horizon control algorithm is shown in Algorithm 2.

By conditioning the GP on the $\tilde{L}$ most recently sampled values and Jacobians at the previous time step during closed-loop operation, smoothness of the GP sample provides that the value and Jacobian in a local neighborhood of the current time step will be close to the previously sampled ones. We expect that standard RTI arguments (see, e.g., [33]) carry over to the proposed implementation using only a fixed number of SQP steps.

V. Simulation results

In this section, we demonstrate our method with two numerical examples. First, we show the effectiveness of the uncertainty propagation resulting from the proposed sampling-based GP-MPC method in comparison to existing methods [7], [16] using a pendulum example. Second, we demonstrate closed-loop constraint satisfaction under challenging constraints for a more realistic car dynamics model.

In both numerical examples, we model the unknown dynamics using a separate GP for each output dimension (cf. Remark 2) and set $\sqrt{\beta} = 2.5$. A squared-exponential kernel is employed, whose hyper-parameters are determined using maximum-likelihood estimation. In both examples, the measurement noise in Eq. (2) is set to $\lambda^2 = 10^{-6}$. For numerical stability, the same noise value is used for the iterative sampling strategy (Section IV-C). To simplify comparisons, the simulations are carried out without a terminal set. Algorithm 1 is implemented[4] using the `Python` interfaces of `acados` [34] and `CasADi` [35]; for efficient GP sampling, we employ `GPyTorch` [31].

A. Comparison of uncertainty propagation using a pendulum

We consider a pendulum whose dynamics are given by

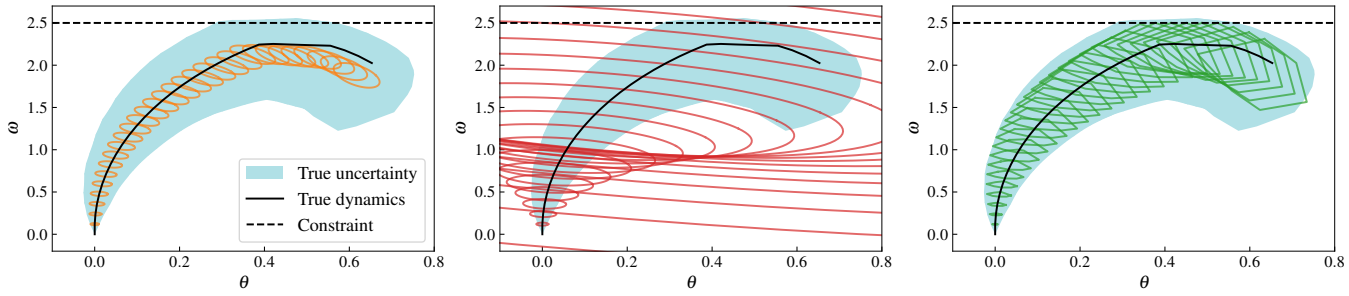
$$\begin{bmatrix}\theta(k+1)\\ \omega(k+1)\end{bmatrix} = \begin{bmatrix}\theta(k) + \omega(k)\Delta\\ \omega(k) - g_a \sin(\theta(k))\Delta/l + \alpha(k)\Delta\end{bmatrix}.$$

The pendulum state is $x = [\theta, \omega]^\top$, where $\theta[\mathrm{rad}]$ denotes the angular position, $\omega[\mathrm{rad/s}]$, the angular velocity, and the control input is the angular acceleration $\alpha[\mathrm{rad/s^2}]$. The constant parameter $l = 1\mathrm{m}$ denotes the length of the pendulum, $\Delta = 0.015\mathrm{s}$, the discretization time and $g_a = 10\mathrm{m/s^2}$, the acceleration due to gravity. We consider the case of completely unknown dynamics, i.e., $f(x,u) = 0$, and residual nonlinear dynamics $g : \mathbb{R}^3 \to \mathbb{R}^2$, $g(x,u) = g(\theta, \omega, \alpha)$, and $B_d = \mathbb{I}_{2\times 2}$. The GP is trained using $|\bar{\mathcal{D}}| = 45$ data points (including derivatives) on an equally-spaced $3 \times 3 \times 5$ mesh grid in the constraint set $\mathcal{Z} = \{(\theta, \omega, \alpha) \in [-2.14, 2.14] \times [-2.5, 2.5] \times [-8, 8]\}$. The task is to control the pendulum from $x = (0, 0)$ to a desired state of $x_{\mathrm{des}} = (2.5, 0)$. The cost of the MPC Problem (7) is given by $l_i(\theta_i, \alpha_i) = 50(x_i - x_{\mathrm{des}})^2 + 0.1\alpha_i^2$, $\forall i = 0, \dots, H-1$, where $H = 31$.

We compare the uncertainty propagation of three methods in Fig. 2: a) Cautious GP-MPC [7], which uses linearization and independence approximations, b) Robust tube-based GP-MPC [16], which uses ellipsoidal tube propagation and c) Robust sampling-based GP-MPC (Algorithm 1), which approximates the reachable set using sampled dynamics. For a better comparison, the uncertainty propagation is shown for the same open-loop control sequence $\boldsymbol{u}$ determined using Algorithm 1 with $N = 20$ dynamics samples. In Fig. 2a, the orange ellipsoids show the confidence sets corresponding to $\sqrt{\beta}$ standard deviations for the state using the Cautious GP-MPC method [7]. The confidence sets significantly under-approximate the true uncertainty (blue shaded area), failing to capture the true system uncertainty. Fig. 2b demonstrates the uncertainty propagation computed using the Robust tube-based GP-MPC [16] method, where the red ellipsoids show safe sets obtained for the same high-probability confidence set $\mathcal{G}$. Due to the exponential growth of the sets along the prediction horizon, it significantly over-approximates the true uncertainty set, which would lead to an overly conservative controller. Fig. 2c shows convex hulls at each horizon of the 20 simulated trajectories, representing the uncertainty propagation of our method. In contrast to the other methods, the sampling-based approximation provides a close approximation of the true reachable set (which can be further improved by using more samples), leading to practically meaningful constraint tightenings.

[3]While this approach may seem related to the limited-memory GP-SSMs introduced in [23], note that, in this case, the memory limitation is expressed across MPC iterations rather than across time. In each MPC iteration, the functions $g^n \sim \mathcal{GP}_{[\underline{g},\overline{g}]}(0, k, \bar{\mathcal{D}})$ implicitly generating the sampled values and Jacobians may thus change; however, they are fixed during the runtime of the SQP algorithm, leading to consistent trajectories at convergence.

[4]The source code is available at `https://github.com/manish-pra/sampling-gpmpc`, doi:10.3929/ethz-b-000693678



(a) Cautious GP-MPC [7] (b) Robust tube-based GP-MPC [16] (c) Robust sampling-based GP-MPC (ours)

Fig. 2: Comparison of uncertainty propagation for a given input sequence $\boldsymbol{u}$ by different GP-MPC methods in the pendulum example. The blue-shaded area represents the true uncertainty propagation for 2×10^5 dynamics sampled from $\mathcal{GP}_{[\underline{g},\overline{g}]}(0, k; \widetilde{\mathcal{D}})$, the solid black line represents the true system, and the dashed line shows the angular velocity constraint. Cautious GP-MPC significantly under-approximates (orange, Fig. 2a), Robust tube-based GP-MPC significantly over-approximates (red, Fig. 2b), whereas Robust sampling-based (proposed) GP-MPC provides a close approximation (green, Fig. 2c) to the true uncertainty set, even with only 20 dynamics samples.

B. Safe closed-loop control with car dynamics

In the following, we illustrate the performance of the proposed receding-horizon implementation for an overtaking maneuver of an autonomous car, with particular emphasis on real-time applicability and closed-loop constraint satisfaction. We model the nonlinear car dynamics using a kinematic bicycle model as follows:

$$\begin{bmatrix} x_p(k+1) \\ y_p(k+1) \\ \theta(k+1) \\ v(k+1) \end{bmatrix} = \underbrace{\begin{bmatrix} x_p(k) \\ y_p(k) \\ \theta(k) \\ v(k)+a(k)\Delta \end{bmatrix}}_{=:f(x,u)} + \underbrace{\begin{bmatrix} \mathbb{I}_{3\times 3} \\ 0_{1\times 3} \end{bmatrix}}_{=:B_d} \underbrace{\begin{bmatrix} v(k)\cos(\theta(k)+\zeta_k)\Delta \\ v(k)\sin(\theta(k)+\zeta_k)\Delta \\ v(k)\sin(\zeta_k) l_r^{-1}\Delta \end{bmatrix}}_{=:g(x,u)}$$

where $\zeta_k = \tan^{-1}\left(\frac{l_r}{l_f+l_r}\tan(\delta(k))\right)$ is the slip angle [rad]. The states $x = [x_p, y_p, \theta, v]^\top$ consist of the position [m] of the car in 2D-Cartesian coordinates $[x_p, y_p]^\top$, its absolute heading angle θ[rad] and longitudinal velocity v[m/s]; the control input $u = [\delta, a]^\top$, of the steering angle δ[rad] and linear acceleration a[m/s^2]. The distances between the car's center of gravity to the front and the rear wheel are denoted by $l_f = 1.105$[m] and $l_r = 1.738$[m], respectively. The known part of the dynamics, $f(x, u)$, assumes simple integrator dynamics. The unknown part, $g : \mathbb{R}^3 \to \mathbb{R}^3$, $g(x, u) = g(\theta, v, \delta)$, is a nonlinear function of heading angle, velocity and steering angle. The GP is trained with 45 prior data points (excluding derivatives) on an equally spaced $3 \times 3 \times 5$ mesh grid along (θ, v, δ) in the set $\{(x, u) \in [-2.14, 70] \times [0, 6] \times [-1.14, 1.14] \times [-1, 15] \times [-0.6, 0.6] \times [-2, 2]\}$, given by the intersection of the track, velocity and input constraints. In addition, the full constraint set includes ellipsoidal obstacle avoidance constraints accounting for the size of the ego and other vehicles, $(x_{p_i} - x_e)^2/9 + (x_{p_i} - y_e)^2 \geq 5.67$, respectively centered at the other vehicle's position (x_e, y_e).

The task is to drive the car in the right lane at $y_{\text{ref}} = 1.95$ to the target destination at $x_{\text{ref}} = 70$ while avoiding obstacles as shown in Fig. 3. Problem (7) is solved with stage cost $l_i(x, u) = \|x_i\|_Q^2 + \|u_i\|_R^2$, where $Q = \text{diag}(2, 36, 0.07, 0.005)$ and $R = \text{diag}(2, 2)$; the prediction horizon is set to $H = 16$ with sampling time $\Delta = 0.06$s. Algorithm 2 is run with $N = 20$ dynamics samples for $L = 2$ SQP iterations. As shown in Fig. 3, the sampling-based GP-MPC plans a safe trajectory with respect to each of the dynamics samples, driving the car to the target destination while avoiding obstacles despite the uncertain dynamics. In Table I, we report the execution time per MPC solve, averaged over 50 time steps, when running Algorithm 2 on an AMD EPYC 7543P Processor at 2.80GHz, with different numbers of SQP iterations L and dynamics samples N. The GP sampling is parallelized on an RTX 4090 GPU, carrying over $\tilde{L} = 1$ GP samples between MPC iterations. In particular, for $N = 20$ dynamics and $L = 1$ SQP iterations real-time feasible execution times of 32.16 ms (≈ 31 Hz) are achieved.

VI. Conclusion

Tractable and accurate propagation of the epistemic uncertainty poses a major obstacle for Gaussian process-based MPC. This paper has taken a step at addressing this challenge by presenting a robust formulation for Gaussian process-based MPC, which employs GP confidence bounds to guarantee constraint satisfaction with high probability. For a tractable solution of the resulting GP-MPC problem, a sampling-based approach has been proposed. Its approximation quality, as well as computational tractability are illustrated using numerical examples. On the theoretical side, a promising future research direction is the establishment of recursive feasibility and closed-loop constraint satisfaction guarantees for a finite number of samples. On the computational side, further speedups may still be achieved by exploiting the structure of the sampling-based dynamics in the QP solver, as well as by employing computationally efficient GP approximations and tailored sampling strategies for GP models with derivatives.

Acknowledgments

The authors would like to thank Elena Arcari and Anna Scampicchio for literature recommendations and valuable discussions. Amon Lahr thanks Katrin Baumgärtner and Andrea Ghezzi for an inspiring hackathon and discussions.

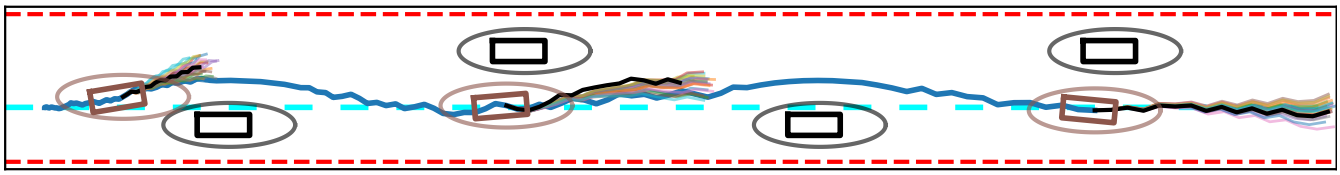


Fig. 3: Demonstration of safe closed-loop control using the proposed sampling-based GP-MPC with car dynamics. The ego car (brown) successfully overtakes other vehicles (black) from left to right. The solid blue line represents the resulting closed-loop trajectory. The dashed cyan line is a reference along the y-axis, while multiple thin lines show trajectory prediction with various trajectory samples used in sampling-based GP-MPC (Algorithm 2).

TABLE I: Average run time and standard deviation (in ms) of Algorithm 1 for $H = 16$ in the car example

N	$L = 1$	$L = 2$	$L = 3$	$L = 4$	$L = 5$
5	20.72 ± 2.07	42.90 ± 3.20	51.58 ± 18.95	78.13 ± 20.70	109.45 ± 16.78
10	24.33 ± 4.23	49.75 ± 5.78	75.76 ± 6.20	103.18 ± 15.04	124.48 ± 21.61
20	32.16 ± 8.77	63.94 ± 5.56	102.22 ± 12.77	138.82 ± 21.00	164.68 ± 26.96
40	51.00 ± 16.71	100.23 ± 17.28	160.88 ± 18.27	237.50 ± 30.94	294.71 ± 38.53
80	111.24 ± 24.72	234.91 ± 38.30	373.92 ± 42.02	487.02 ± 81.19	668.24 ± 55.64

REFERENCES

[1] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*, ser. Adaptive Computation and Machine Learning. Cambridge, Massachusetts: MIT Press, 2006.

[2] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model Predictive Control: Theory, Computation and Design*, 2nd ed. Santa Barbara: Nob Hill Publishing, LLC, 2020.

[3] C. J. Ostafew, A. P. Schoellig, and T. D. Barfoot, "Robust Constrained Learning-based NMPC enabling reliable mobile robot path tracking," *The International Journal of Robotics Research*, vol. 35, no. 13, 2016.

[4] G. Cao, E. M.-K. Lai, and F. Alam, "Gaussian Process Model Predictive Control of an Unmanned Quadrotor," *J Intell Robot Syst*, vol. 88, no. 1, 2017.

[5] A. Carron, E. Arcari, M. Wermelinger, L. Hewing, M. Hutter, and M. N. Zeilinger, "Data-Driven Model Predictive Control for Trajectory Tracking With a Robotic Arm," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, 2019.

[6] J. Kabzan, L. Hewing, A. Liniger, and M. N. Zeilinger, "Learning-Based Model Predictive Control for Autonomous Racing," *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, 2019.

[7] L. Hewing, J. Kabzan, and M. N. Zeilinger, "Cautious Model Predictive Control Using Gaussian Process Regression," *IEEE Trans Control Syst Technol*, vol. 28, no. 6, 2020.

[8] S. Vaskov, R. Quirynen, M. Menner *et al.*, "Friction-Adaptive Stochastic Predictive Control for Trajectory Tracking of Autonomous Vehicles," in *Proc. American Control Conference (ACC)*, 2022.

[9] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-Based Model Predictive Control: Toward Safe Learning in Control," *Annu. Rev. Control Robot. Auton. Syst.*, vol. 3, no. 1, 2020.

[10] A. Lahr, A. Zanelli, A. Carron, and M. N. Zeilinger, "Zero-order optimization for Gaussian process-based model predictive control," *European Journal of Control*, vol. 74, p. 100862, 2023.

[11] A. Girard, C. Rasmussen, J. Q. Candela, and R. Murray-Smith, "Gaussian Process Priors with Uncertain Inputs Application to Multiple-Step Ahead Time Series Forecasting," in *Advances in Neural Information Processing Systems*, vol. 15. MIT Press, 2002.

[12] L. Hewing, E. Arcari, L. P. Fröhlich, and M. N. Zeilinger, "On Simulation and Trajectory Prediction with Gaussian Process Dynamics," in *Learning for Dynamics and Control*. PMLR, 2020.

[13] J. Quinonero-Candela, A. Girard, J. Larsen, and C. Rasmussen, "Propagation of uncertainty in Bayesian kernel models - application to multiple-step ahead forecasting," in *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03).*, vol. 2. Hong Kong, China: IEEE, 2003.

[14] Y. Pan, X. Yan, E. A. Theodorou, and B. Boots, "Prediction under Uncertainty in Sparse Spectrum Gaussian Processes with Applications to Filtering and Control," in *Proceedings of the 34th International Conference on Machine Learning*. PMLR, 2017.

[15] R. Quirynen and K. Berntorp, "Uncertainty Propagation by Linear Regression Kalman Filters for Stochastic NMPC," *IFAC-PapersOnLine*, vol. 54, no. 6, 2021.

[16] T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause, "Learning-Based Model Predictive Control for Safe Exploration," in *IEEE Conference on Decision and Control (CDC)*, 2018.

[17] E. T. Maddalena, P. Scharnhorst, Y. Jiang, and C. N. Jones, "KPC: Learning-based model predictive control with deterministic guarantees," in *Learning for Dynamics and Control*. PMLR, 2021.

[18] E. Bradford, L. Imsland, D. Zhang, and E. A. del Rio Chanona, "Stochastic data-driven model predictive control using gaussian processes," *Computers & Chemical Engineering*, vol. 139, 2020.

[19] E. Bradford, L. Imsland, M. Reble, and E. A. del Rio-Chanona, "Hybrid Gaussian Process Modeling Applied to Economic Stochastic Model Predictive Control of Batch Processes," in *Recent Advances in Model Predictive Control: Theory, Algorithms, and Applications*. Cham: Springer International Publishing, 2021.

[20] S. Conti, J. P. Gosling, J. E. Oakley, and A. O'Hagan, "Gaussian process emulation of dynamic computer codes," *Biometrika*, vol. 96, no. 3, 2009.

[21] J. Umlauft, T. Beckers, and S. Hirche, "Scenario-based optimal control for gaussian process state space models," in *European Control Conference (ECC)*. IEEE, 2018, pp. 1386–1392.

[22] A. Lederer, Q. Hao, and S. Hirche, "Confidence Regions for Simulations with Learned Probabilistic Models," in *American Control Conference (ACC)*, 2020.

[23] T. Beckers and S. Hirche, "Prediction with Approximated Gaussian Process Dynamical Models," *IEEE Transactions on Automatic Control*, 2021.

[24] J. A. Lin, J. Antorán, S. Padhy, D. Janz, J. M. Hernández-Lobato, and A. Terenin, "Sampling from gaussian process posteriors using stochastic gradient descent," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[25] M. Diehl, H. G. Bock, and J. P. Schlöder, "A Real-Time Iteration Scheme for Nonlinear Optimization in Optimal Feedback Control," *SIAM J. Control Optim.*, vol. 43, no. 5, 2005.

[26] S. R. Chowdhury and A. Gopalan, "On kernelized multi-armed bandits," in *International Conference on Machine Learning*. PMLR, 2017.

[27] G. Fasshauer and M. McCourt, *Kernel-Based Approximation Methods Using MATLAB*. WORLD SCIENTIFIC, 2015.

[28] L. Chisci, J. A. Rossiter, and G. Zappa, "Systems with persistent disturbances: predictive control with restricted constraints," *Automatica*, vol. 37, no. 7, pp. 1019–1028, 2001.

[29] K. P. Wabersich and M. N. Zeilinger, "Linear Model Predictive Safety Certification for Learning-Based Control," in *IEEE Conference on Decision and Control (CDC)*, 2018.

[30] J. Nocedal and S. J. Wright, *Numerical Optimization*, second edition ed., ser. Springer Series in Operation Research and Financial Engineering. New York: Springer, 2006.

[31] J. Gardner, G. Pleiss, K. Q. Weinberger, D. Bindel, and A. G. Wilson, "GPyTorch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration," in *Advances in Neural Information Processing Systems*, vol. 31. Curran Associates, Inc., 2018.

[32] M. Diehl, "Real-Time Optimization for Large Scale Nonlinear Processes," Ph.D. dissertation, University of Heidelberg, 2001.

[33] A. Zanelli, Q. Tran-Dinh, and M. Diehl, "Contraction Estimates for Abstract Real-Time Algorithms for NMPC," in *IEEE 58th Conference on Decision and Control (CDC)*, 2019.

[34] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. van Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, "Acados—a modular open-source framework for fast embedded optimal control," *Math. Prog. Comp.*, vol. 14, no. 1, 2022.

[35] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: A software framework for nonlinear optimization and optimal control," *Math. Prog. Comp.*, vol. 11, no. 1, 2019.