Cautious Model Predictive Control using Gaussian Process Regression

Lukas Hewing, Juraj Kabzan, Melanie N. Zeilinger

Abstract—**Gaussian process (GP) regression has been widely used in supervised machine learning due to its flexibility and inherent ability to describe uncertainty in function estimation. In the context of control, it is seeing increasing use for modeling of nonlinear dynamical systems from data, as it allows the direct assessment of residual model uncertainty. We present a model predictive control (MPC) approach that integrates a nominal system with an additive nonlinear part of the dynamics modeled as a GP. Approximation techniques for propagating the state distribution are reviewed and we describe a principled way of formulating the chance constrained MPC problem, which takes into account residual uncertainties provided by the GP model to enable cautious control. Using additional approximations for efficient computation, we finally demonstrate the approach in a simulation example, as well as in a hardware implementation for autonomous racing of remote controlled race cars, highlighting improvements with regard to both performance and safety over a nominal controller.**

Index Terms—**Model Predictive Control, Gaussian Processes, Learning-based Control**

I. INTRODUCTION

Many modern control approaches depend on accurate model descriptions to enable safe and high performance control. Identifying these models, especially for highly nonlinear systems, is a time-consuming and complex endeavor. Often times, however, it is possible to derive an approximate system model, e.g. a linear model with adequate accuracy close to some operating point, or a simple model description from first principles. In addition, measurement data from previous experiments or during operation is often available, which can be exploited to enhance the system model and controller performance. In this paper, we present a model predictive control (MPC) approach, which improves such a nominal model description from data using Gaussian Processes (GPs) to safely enhance performance of the system.

Learning methods for automatically identifying dynamical models from data have gained significant attention in the past years [1] and more recently also for robotic applications [2], [3]. In particular, nonparametric methods, which have seen wide success in machine learning, offer significant potential for control [4]. The appeal of using Gaussian Process regression for model learning stems from the fact that it requires little prior process knowledge and directly provides a measure of residual model uncertainty. In predictive control, GPs were successfully applied to improve control performance when learning periodic time-varying disturbances [5]. The task of

learning the system dynamics as opposed to disturbances, however, imposes the challenge of propagating probability distributions of the state over the prediction horizon of the controller. Efficient methods to evaluate GPs for Gaussian inputs have been developed in [6], [7], [8]. By successively employing these approximations at each prediction time step, predictive control with GP dynamics has been presented in [9]. In [10] a piecewise linear approximate explicit solution for the MPC problem of a combustion plant was presented. Application of a one-step MPC with a GP model to a mechatronic system was demonstrated in [11] and the use for fault-tolerant MPC was presented in [12]. A constrained tracking MPC for robotic applications, which uses a GP to improve the dynamics model from measurement data, was shown in [13]. A variant of this approach was realized in [14], where uncertainty of the GP prediction is robustly taken into account using confidence bounds. Solutions of GP-based MPC problems using sequential quadratic programming schemes is discussed in [15]. A simulation study for the application of GP-based predictive control to a drinking water network was presented in [16] and data efficiency of these formulations for learning controllers was recently demonstrated in [17].

The goal of this paper is both to provide an overview of existing techniques and to propose extensions for deriving a systematic and efficiently solvable approximate formulation of the MPC problem with a GP model, which incorporates constraints and takes into account the model uncertainty for cautious control. Unlike most previous approaches, we specifically consider the combination of a nominal system description with an additive GP part which can be of different dimensionality as the nominal model. This corresponds to a common scenario in controller design, where an approximate nominal model is known and can be improved on, offering a multitude of advantages. A nominal model allows for rudimentary system operation and the collection of measurement data, as well as the design of pre-stabilizing ancillary controllers which reduce the state uncertainty in prediction. Additionally, it allows for learning only specific effects from data, potentially reducing the dimensionality of the machine learning task. For example, most dynamical models derived from physical laws include an integrator chain, which can be directly represented in the nominal dynamics and for which no nonlinear uncertainty model has to be learned from data—it would rather add unnecessary conservatism to the controller. The separation between system and GP model dimension is therefore key for the application of the GP-based predictive control scheme to higher order systems.

The paper makes the following contributions. A review and compact summary of approximation techniques for prop-

agating GP dynamics and uncertainties is provided and extended to the additive combination with nominal dynamics. The approximate propagation enables a principled formulation of chance constraints on the resulting state distributions in terms of probabilistic reachable sets [18]. In addition, the nominal system description allows for reductin the GP model learning to a subspace of states and inputs. We discuss sparse GPs and a tailored selection of inducing points for MPC to reduce the computational burden of the approach. The use of some or all of these techniques is crucial to make the computationally expensive GP approach practically feasible for control with sampling times in the millisecond range. We finally present two application examples. The first is a simulation of an autonomous underwater vehicle, illustrating key concepts and advantages in a simplified setting. Second, we present a hardware implementation for autonomous racing of remote controlled cars, showing the real-world feasibility of the approach for complex high performance control tasks. To the best of our knowledge this is the first hardware implementation of a Gaussian Process-based predictive control scheme to a system of this complexity at sampling times of $20\,\mathrm{ms}$.

II. Preliminaries

A. Notation

The i-th element of a vector x is denoted $[x]_i$. Similarly, $[M]_{ij}$ denotes element ij of a matrix M, and $[M]_{.i}$, $[M]_{i.}$ its i-th column or row, respectively. We use $\mathrm{diag}(x)$ to refer to a diagonal matrix with entries given by the vector x. The squared Euclidean norm weighted by M, i.e. $x^\mathsf{T} M x$ is denoted $\|x\|_M^2$. We use boldface to emphasize stacked quantities, e.g. a collection of vector-valued data in matrix form. $\nabla f(z)$ is the gradient of f evaluated at z. The Pontryagin set difference is denoted $\mathcal{A} \ominus \mathcal{B} = \{a \,|\, a + b \in \mathcal{A} \; \forall b \in \mathcal{B}\}$. A normally distributed vector x with mean μ and variance Σ is given by $x \sim \mathcal{N}(\mu, \Sigma)$. The expected value of x is $\mathbb{E}(x)$, $\mathrm{cov}(x, y)$ is the covariance between vectors x and y, and the variance $\mathrm{var}(x) = \mathrm{cov}(x, x)$. We use $p(x)$, $(p(x|y))$ to refer to the (conditional) probability densities of x. Similary $\mathrm{Pr}(E)$, $(\mathrm{Pr}(E \,|\, A))$ denotes the probability of an event E (given A).

Realized quantities during closed loop control are time indexed using parenthesis, i.e. $x(k)$, while quantities in prediction use subscripts, e.g. x_i is the predicted state i-steps ahead.

B. Problem formulation

We consider the control of dynamical systems that can be represented by the discrete-time model

$$x(k+1) = f(x(k), u(k)) + B_d(g(x(k), u(k)) + w(k)), \quad (1)$$

where $x(k) \in \mathbb{R}^{n_x}$ is the system state and $u(k) \in \mathbb{R}^{n_u}$ the control inputs at time k. The model is composed of a known nominal part f and additive dynamics g that describe initially unknown dynamics of the system, which are to be learned from data and are assumed to lie in the subspace spanned by B_d. We consider i.i.d. process noise $w_k \sim \mathcal{N}(0, \Sigma^w)$, which is spatially uncorrelated, i.e. has diagonal variance matrix

$\Sigma^w = \mathrm{diag}([\sigma_1^2, \ldots, \sigma_{n_d}^2])$. We assume that both f and g are differentiable functions.

The system is subject to state and input constraints $\mathcal{X} \subseteq \mathbb{R}^{n_x}$, $\mathcal{U} \subseteq \mathbb{R}^{n_u}$, respectively. The constraints are formulated as chance constraints, i.e. by enforcing

$$\mathrm{Pr}(x(k) \in \mathcal{X}) \geq p_x,$$
$$\mathrm{Pr}(u(k) \in \mathcal{U}) \geq p_u,$$

where p_x, p_u are the associated satisfaction probabilities.

C. Gaussian Process Regression

Gaussian process regression is a nonparametric framework for nonlinear regression. A GP is a probability distribution over functions, such that every finite sample of function values is jointly Gaussian distributed. We apply Gaussian processes regression to infer the noisy vector-valued function $g(x, u)$ in (1) from previously collected measurement data of states and inputs $\{(x_j, u_j), \; j = 0, \ldots, M\}$. State-input pairs form the input data to the GP and the corresponding outputs are obtained from the deviation to the nominal system model:

$$y_j = g(x_j, u_j) + w_j = B_d^\dagger (x_{j+1} - f(x_j, u_j)),$$

where $B_d^\dagger$ is the Moore-Penrose pseudo-inverse. Note that the measurement noise on the data points w_j corresponds to the process noise in (1). With $z_j := [x_j^\mathsf{T}, u_j^\mathsf{T}]^\mathsf{T}$, the data set of the GP is therefore given by

$$\mathcal{D} = \{\mathbf{y} = [y_0, \ldots, y_M]^\mathsf{T} \in \mathbb{R}^{M \times n_d},$$
$$\mathbf{z} = [z_0, \ldots, z_M]^\mathsf{T} \in \mathbb{R}^{M \times n_z}\}.$$

Each output dimension is learned individually, meaning that we assume the components of each y_j to be independent, given the input data z_j. Specifying a GP prior on g in each output dimension $a \in \{1, \ldots, n_d\}$ with kernel $k^a(\cdot, \cdot)$ and prior mean function $m^a(\cdot)$ results in normally distributed measurement data with

$$[\mathbf{y}]_{.,a} \sim \mathcal{N}(m(\mathbf{z}), K_{\mathbf{zz}}^a + I\sigma_a^2), \quad (2)$$

where $K_{\mathbf{zz}}^a$ is the Gram matrix of the data points, i.e. $[K_{\mathbf{zz}}^a]_{ij} = k^a(z_i, z_j)$ and $m^a(\mathbf{z}) = [m^a(z_0), \ldots, m^a(z_M)]^\mathsf{T}$. The choice of kernel functions k^a and its parameterization is the determining factor for the inferred distribution of g and is typically specified using prior process knowledge and optimization [19], e.g. by optimizing the likelihood of the observed data distribution (2). Throughout this paper we consider the squared exponential kernel function

$$k^a(z_i, z_j) = \sigma_{f,a}^2 \exp\left(-\frac{1}{2}(z_i - z_j)^\mathsf{T} L_a^{-1} (z_i - z_j)\right), \quad (3)$$

in which L_a is a positive diagonal length scale matrix and $\sigma_{f,a}^2$ the signal variance. It is, however, straightforward to use any other (differentiable) kernel function.

The joint distribution of the training data and an arbitrary test point z in output dimension a is given by

$$p([y]_a, [\mathbf{y}]_{.,a}) = \mathcal{N}\left(\begin{bmatrix} m^a(\mathbf{z}) \\ m^a(z) \end{bmatrix}, \begin{bmatrix} K_{\mathbf{zz}}^a + I\sigma_a^2 & K_{\mathbf{z}z}^a \\ K_{z\mathbf{z}}^a & K_{zz}^a \end{bmatrix}\right), \quad (4)$$

where $[K^a_{\mathbf{z}z}]_j = k^a(z_j, z)$, $K^a_{z\mathbf{z}} = (K^a_{\mathbf{z}z})^\mathsf{T}$ and similarly $K^a_{zz} = k^a(z, z)$. The resulting distribution of $[y]_a$ conditioned on the observed data points is again Gaussian with $p([y]_a \mid [\mathbf{y}]_{\cdot,a}) = \mathcal{N}\left(\mu^d_a(z), \Sigma^d_a(z)\right)$ and

$$\mu^d_a(z) = K^a_{z\mathbf{z}}(K^a_{\mathbf{z}\mathbf{z}} + I\sigma^2_a)^{-1}[\mathbf{y}]_{\cdot,a}, \tag{5a}$$

$$\Sigma^d_a(z) = K^a_{zz} - K^a_{z\mathbf{z}}(K^a_{\mathbf{z}\mathbf{z}} + I\sigma^2_a)^{-1}K^a_{\mathbf{z}z}. \tag{5b}$$

The resulting multivariate GP approximation of the unknown function $g(z)$ is then simply given by stacking the individual output dimensions, i.e.

$$d(z) \sim \mathcal{N}\left(\mu^d(z), \Sigma^d(z)\right) \tag{6}$$

with $\mu^d = [\mu^d_1, \ldots, \mu^d_{n_d}]^\mathsf{T}$ and $\Sigma^d = \mathrm{diag}([\Sigma^d_1, \ldots, \Sigma^d_{n_d}]^\mathsf{T})$.

Evaluating mean and variance in (5) has cost $\mathcal{O}(n_d n_z M)$ and $\mathcal{O}(n_d n_z M^2)$, respectively and thus scales with the number of data points. For large amounts of data points or fast real-time applications this can limit the use of GP models. To overcome these issues, various approximation techniques have been proposed, one class of which is sparse Gaussian processes using inducing inputs [20], briefly outlined in the following.

D. Sparse Gaussian Processes

Many sparse GP approximations make use of *inducing* targets $\mathbf{y}_{\mathrm{ind}}$, inputs $\mathbf{z}_{\mathrm{ind}}$ and conditional distributions to approximate the joint distribution (4) [21]. Many such approximations exist, a popular of which is the Fully Independent Training Conditional (FITC) [22], which we make use of in this paper. Given a selection of inducing inputs $\mathbf{z}_{\mathrm{ind}}$ and using the shorthand notation $Q^a_{\zeta\tilde{\zeta}} := K^a_{\zeta\mathbf{z}_{\mathrm{ind}}}(K^a_{\mathbf{z}_{\mathrm{ind}}\mathbf{z}_{\mathrm{ind}}})^{-1}K^a_{\mathbf{z}_{\mathrm{ind}}\tilde{\zeta}}$ the approximate posterior distribution is given by

$$\tilde{\mu}^d_a(z) = Q^a_{z\mathbf{z}}(Q^a_{\mathbf{z}\mathbf{z}} + \Lambda)^{-1}[\mathbf{y}]_{\cdot,a}, \tag{7a}$$

$$\tilde{\Sigma}^d_a(z) = K^a_{zz} - Q^a_{z\mathbf{z}}(Q^a_{\mathbf{z}\mathbf{z}} + \Lambda)^{-1}Q^a_{\mathbf{z}z} \tag{7b}$$

with $\Lambda = \mathrm{diag}(K^a_{\mathbf{z}\mathbf{z}} - Q^a_{\mathbf{z}\mathbf{z}} + I\sigma^2_a)$. Concatenating the output dimensions similar to (5) we arrive at the approximation

$$\tilde{d}(z) \sim \mathcal{N}\left(\tilde{\mu}^d(z), \tilde{\Sigma}^d(z)\right).$$

Several of the matrices used in (7) can be precomputed and the evaluation complexity becomes independent of the number of original data points. With $\tilde{M}$ being the number of inducing points, this results in $\mathcal{O}(n_d n_z \tilde{M})$ and $\mathcal{O}(n_d n_z \tilde{M}^2)$ for the predictive mean and variance, respectively.

There are numerous options for selecting the inducing inputs, e.g. heuristically as a subset of the original data points, by treating them as hyperparameters and optimizing their location [22], or letting them coincide with test points [23], which is often referred to as *transductive* learning. In Section III-C we make use of such transductive ideas and propose a dynamic selection of inducing points, with a resulting local approximation tailored to the predictive control task.

III. MPC CONTROLLER DESIGN

We consider the design of an MPC controller for system (1) using a GP approximation d of the unknown function g:

$$x_{i+1} = f(x_i, u_i) + B_d\left(d(x_i, u_i) + w_i\right). \tag{8}$$

At each time step, the GP approximation evaluates to a stochastic distribution according to the residual model uncertainty and process noise, which is then propagated forward in time. A stochastic MPC formulation allows the principled treatment of chance constraints, i.e. by imposing a prescribed maximum probability of constraint violation. Denoting a state and input reference trajectory by $X^r = \{x^r_0, \ldots, x^r_N\}$, $U^r = \{u^r_0, \ldots, u^r_{N-1}\}$, respectively, the resulting stochastic finite time optimal control problem can be formulated as

$$\min_{\Pi(x)} \quad \mathbb{E}\left(l_f(x_N - x^r_N) + \sum_{i=0}^{N-1} l(x_i - x^r_i, u_i - u^r_i)\right) \tag{9a}$$

$$\text{s.t.} \quad x_{i+1} = f(x_i, u_i) + B_d(d(x_i, u_i) + w_i), \tag{9b}$$

$$u_i = \pi_i(x_i), \tag{9c}$$

$$\Pr(x_{i+1} \in \mathcal{X}) \geq p_x, \tag{9d}$$

$$\Pr(u_i \in \mathcal{U}) \geq p_u, \tag{9e}$$

$$x_0 = x(k), \tag{9f}$$

for all $i = 0, \ldots, N-1$ with appropriate terminal cost $l_f(x_N)$ and stage cost $l(x_i, u_i)$, where the optimization is carried out over a sequence of input policies $\Pi(x) = \{\pi_0(x), \ldots, \pi_{N-1}(x)\}$.

In the form (9), the optimization problem is computationally intractable. In the following sections, we present techniques for deriving an efficiently solvable approximation. Specifically, we use affine feedback policies with precomputed linear gains and show how this allows simple approximate propagation of the system uncertainties in terms of mean and variance in prediction. Using these approximate distributions, we present a framework for reformulating chance constraints deterministically and briefly discuss the evaluation of possible cost functions.

A. Ancillary Linear State Feedback Controller

Optimization over general feedback policies $\Pi(x)$ is an infinite dimensional optimization problem. A common approach that integrates well in the presented framework is to restrict the policy class to linear state feedback controllers

$$\pi_i(x_i) = K_i(\mu^x_i - x_i) + \mu^u_i,$$

where μ^x_i is the predicted mean of the state distribution. Using pre-selected gains K_i, we then optimize over μ^u_i, the mean of the applied input. Note that due to the ancillary control law the input u_i applied in prediction is a random variable, resulting from the affine transformation of x_i.

In Fig. 1, the effect of state feedback on the propagation of uncertainty is exemplified. It shows the evolution over time of a double integrator with a quadratic friction term inferred by a GP. The right plot displays mean and 2-σ variance of a number of trajectories from repeated simulations of the system with
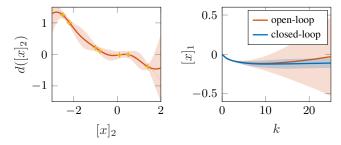
Fig. 1: Propagation of uncertainty for double integrator with GP trained on a nonlinear friction term. The left plot displays the GP with 2-σ confidence bound, while the right plot shows the mean and 2-σ variance of repeated simulation runs of the system under an open-loop control sequence and closed-loop state feedback control.

different noise realizations. Mean and variance under an open-loop control sequence derived from a linear quadratic infinite time optimal control problem are shown in red. The results with a corresponding LQR feedback law applied in closed-loop are shown in blue. As evident, open-loop input sequences can lead to rapid growth of uncertainty in the prediction, and thereby cause conservative control actions in the presence of chance constraints. Linear ancillary state feedback controllers are therefore commonly employed in stochastic and robust MPC [24].

The adequate choice of ancillary feedback gains K_i is generally a hard problem, especially if the system dynamics are highly nonlinear. A useful heuristic is to consider a linearization of the system dynamics around an approximate prediction trajectory, which in MPC applications is typically available using the solution trajectory of the previous time step. Feedback gains for the linearized system can be derived e.g. by solving a finite horizon LQR problem [25]. For mild or stabilizing nonlinearities, a fixed controller gain $K_i = K$ can be chosen to reduce computational burden, as e.g. done for the example in Fig. 1.

B. Uncertainty Propagation

1) Approximation as Normal Distributions: Because of stochastic process noise and the representation by a GP model, future predicted states result in stochastic distributions. Evaluating the posterior of a GP from an input distribution is generally intractable and the resulting distribution is not Gaussian [6]. While under certain assumptions on g, some strict over-approximations of the resulting distributions exist [26], they are typically very conservative and computationally demanding. We focus instead on computationally cheap and practical approximations at the cost of strict guarantees.

State, control input and nonlinear disturbance are approximated as jointly Gaussian distributed at every time step.

$$
\begin{bmatrix} x_i^{\mathsf{T}} & u_i^{\mathsf{T}} & (d_i + w_i)^{\mathsf{T}} \end{bmatrix}^{\mathsf{T}} \sim \mathcal{N}(\mu_i, \Sigma_i)
$$
$$
= \mathcal{N}\left(\begin{bmatrix} \mu_i^x \\ \mu_i^u \\ \mu_i^d \end{bmatrix}, \begin{bmatrix} \Sigma_i^x & \Sigma_i^{xu} & \Sigma_i^{xd} \\ \star & \Sigma_i^u & \Sigma_i^{ud} \\ \star & \star & \Sigma_i^d + \Sigma^w \end{bmatrix} \right), \quad (10)
$$

where $\Sigma_i^u = K_i \Sigma_i^x K_i^{\mathsf{T}}$, $\Sigma_i^{xu} = \Sigma_i^x K_i^{\mathsf{T}}$ and $\star$ denotes terms given by symmetry. Considering the covariances between states, inputs and GP, i.e. Σ^{xd} and Σ^{ud}, is of great importance for an accurate propagation of uncertainty when the GP model d is paired with a nominal system model f. Using a linearization of the nominal system dynamics around the mean

$$
f(x, u) \approx f(\mu^x, \mu^u) + \nabla f(\mu^x, \mu^u) \left(\begin{bmatrix} x \\ u \end{bmatrix} - \begin{bmatrix} \mu^x \\ \mu^u \end{bmatrix} \right),
$$

similar to extended Kalman filtering, this permits simple update equations for the state mean and variance based on affine transformations of the Gaussian distribution in (10)

$$
\mu_{i+1}^x = f(\mu_i^x, \mu_i^u) + B_d \mu_i^d,
$$
$$
\Sigma_{i+1}^x = [\nabla f(\mu_i^x, \mu_i^u) \ B_d] \Sigma_i [\nabla f(\mu_i^x, \mu_i^u) \ B_d]^{\mathsf{T}}.
$$

2) Gaussian Process Prediction from Uncertain Inputs: In order to define $\mu_i^d, \Sigma_i^d, \Sigma_i^{xd}$ and Σ_i^{ud} different approximations of the posterior of a GP from a Gaussian input have been proposed in the literature. In the following, we will give a brief overview of the most commonly used techniques, for details please refer to [6], [7], [8]. For comparison, we furthermore state the computational complexity of these methods. For notational convenience, we will use

$$
\mu_i^z = \begin{bmatrix} \mu_i^x \\ \mu_i^u \end{bmatrix}, \ \Sigma_i^z = \begin{bmatrix} \Sigma_i^x & \Sigma_i^{xu} \\ \star & \Sigma_i^u \end{bmatrix}, \ \Sigma_i^{zd} = \begin{bmatrix} \Sigma_i^{xd} \\ \Sigma_i^{ud} \end{bmatrix}.
$$

a) Mean Equivalent Approximation: A straightforward and computationally cheap approach is to evaluate (6) at the mean, such that

$$
\mu_i^d = \mu^d(\mu_i^z), \quad (12a)
$$
$$
\begin{bmatrix} \Sigma_i^{zd} \\ \Sigma_i^d \end{bmatrix} = \begin{bmatrix} 0 \\ \Sigma^d(\mu_i^z) \end{bmatrix}. \quad (12b)
$$

In [27] it was demonstrated that this can lead to poor approximations for increasing prediction horizons as it neglects the accumulation of uncertainty in the GP. The fact that this approach neglects covariance between d and (x, u) can in addition severely deteriorate the prediction quality when paired with a nominal system $f(x, u)$. The computationally most expensive operation is the matrix multiplication in (5) for each output dimension of the GP, such that the complexity of one prediction step is $\mathcal{O}(n_d n_z M^2)$.

b) Taylor Approximation: Using a first-order Taylor approximation of (5), the expected value, variance and covariance of the resulting distribution results in

$$
\mu_i^d = \mu^d(\mu_i^z), \quad (13a)
$$
$$
\begin{bmatrix} \Sigma_i^{zd} \\ \Sigma_i^d \end{bmatrix} = \begin{bmatrix} \Sigma_i^z (\nabla \mu^d(\mu_i^z))^{\mathsf{T}} \\ \Sigma^d(\mu_i^z) + \nabla \mu^d(\mu_i^z) \Sigma_i^z (\nabla \mu^d(\mu_i^z))^{\mathsf{T}} \end{bmatrix}. \quad (13b)
$$

Compared to (12) this leads to correction terms for the posterior variance and covariance, taking into account the gradient of the posterior mean and the variance of the input of the GP. The complexity of one prediction step amounts to $\mathcal{O}(n_d n_z^2 M^2)$. Higher order approximations are similarly possible at the expense of increased computational effort.
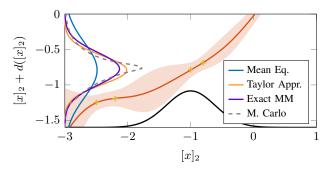
Fig. 2: Comparison of prediction methods for Gaussian input $[x]_2$. The posterior distribution $[x]_2 + d([x]_2)$ is evaluated with the different approximation methods of Section III-B2. For reference, the true distribution is approximated by Monte Carlo simulation.

c) Exact Moment Matching: It has been shown that for certain mean and kernel functions, the first and second moments of the posterior distribution can be analytically computed [6]. Under the assumption that $z_i = [x_i^\mathsf{T}, u_i^\mathsf{T}]^\mathsf{T}$ is normally distributed, we can define

$$\mu_i^d = \mathbb{E}\left(d(z_i)\right), \tag{14a}$$

$$\begin{bmatrix} \Sigma_i^{zd} \\ \Sigma_i^d \end{bmatrix} = \begin{bmatrix} \mathrm{cov}\left(z_i, d(z_i)\right) \\ \mathrm{var}\left(d(z_i)\right) . \end{bmatrix} \tag{14b}$$

In particular, this is possible for a zero prior mean function and the squared exponential kernel. As the procedure exactly matches first and second moments of the posterior distribution, it can be interpreted as an optimal fit of a Gaussian to the true distribution. Note that the model formulation directly allows for the inclusion of linear prior mean functions, as they can be expressed as a nominal linear system in (8), while preserving exact computations of mean and variance. The computational complexity of this approach is $\mathcal{O}(n_d^2 n_z M^2)$ [8]. In Fig. 2 the different approximation methods are compared for a one-step prediction of the GP shown in Fig. 1 with normally distributed input. As evident, the predictions with *Taylor Approximation* and *Moment Matching* are qualitatively similar, which will typically be the case if the second derivative of the posterior mean and variance of the GP is small over the input distribution. It is furthermore apparent that the posterior distribution is not Gaussian and that the approximation as a Gaussian distribution leads to prediction error, even if the first two moments are matched exactly. This can lead to the effect that locally the *Taylor Approximation* provides a closer fit to the underlying distribution. The *Mean Equivalent Approximation* is very conservative by neglecting the covariance between $[x]_2$ and $d([x]_2)$. Note that depending on the sign of the covariance it can also be overly confident.

All presented approximations scale directly with the input and output dimensions, as well as the number of data points and thus become expensive to evaluate in high dimensional spaces. This presents a challenge for predictive control approaches with GP models, which in the past have mainly focused on relatively small and slow systems [9], [12]. Using a nominal model, however, it is possible to consider GPs that depend on only a subset of states and inputs, such that the computational burden can be significantly reduced. This is due to a reduction in the effective input dimension n_z, and more importantly due to a reduction the in necessary training points M for learning in a lower dimensional space.

Remark 1. *With only slight notational changes, the presented approximation methods similarly apply to prior mean and kernel functions that are functions of only a subset of states and inputs.*

Another significant reduction in the computational complexity can be achieved by employing sparse GP approaches, for which in the following we present a modification tailored to predictive control.

C. Dynamic Sparse GPs for MPC

To reduce the computational burden one can make use of sparse GP approximations as outlined in Section II-D, which often comes with little deterioration of the prediction quality. A property of the task of predictive control is that it lends itself to *transductive* approximation schemes, meaning that we adjust the approximation based on the test points, i.e. the (future) evaluation locations in order to achieve a high fidelity local approximation. In MPC, some knowledge of the test points is typically available in terms of an approximate trajectory through the state-action space. This trajectory can, e.g., be given by the reference signal or a previous solution trajectory which will typically lie close to the current one. We therefore propose to select inducing inputs locally at each sampling time according to the prediction quality on these approximate trajectories. Ideally, the local inducing inputs would be optimized to maximize predictive quality, which is, however, not
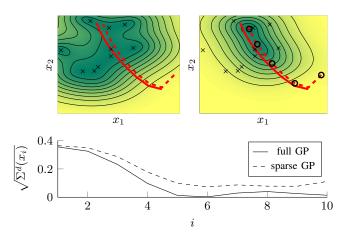

Fig. 3: Illustration of dynamic sparse approximation [28]. Countor plot of the posterior variance of the full GP (top left) and dynamic sparse approximation (top right) with corresponding data points as black crosses. Trajectories planned by an MPC are shown as solid red lines, while the dashed lines show the prediction of the previous time step and used in the approximation, with the chosen inducing points indicated by black circles. The bottom plot shows the respective variances along the planned trajectory.

computationally feasible in the targeted millisecond sampling times. Instead, inducing inputs are selected heuristically along the approximate state input trajectory, specifically we focus here on the MPC solution computed at a previous time step.

To illustrate the procedure, we consider a simple double integrator system controlled by an MPC. Fig. 3 shows the variance $\Sigma^d(x)$ of a GP trained on the systems' states. Additionally, two successive trajectories from an MPC algorithm are displayed, in which the solid red line is the current prediction, while the dashed line is the prediction trajectory from the previous iteration. The plot on the left displays the original GP, with data points marked as crosses, whereas on the right we have the sparse approximation resulting from placing inducing points along the previous solution trajectory, indicated by circles. The figure illustrates how full GP and sparse approximation match closely along the predicted trajectory of the system, while approximation quality far away from the trajectory deteriorates. Since current and previous trajectory are similar, however, local information is sufficient for computation of the MPC controller.

D. Chance Constraint Formulation

The tractable Gaussian approximation of the state and input distribution over the prediction horizon in (10) can be used to approximate the chance constraints in (9d) and (9e).

We reformulate the chance constraints on state and input w.r.t. their means μ^x, μ^u using constraint tightening based on the respective errors $e_i^x = \mu_i^x - x_i$ and $e_i^u = K_i(\mu_i^x - x_i)$. For this we make use of probabilistic reachable sets [18], an extension of the concept of reachable sets to stochastic systems, which is related to probabilistic set invariance [29], [30].

Definition 1 (Probabilistic i-step Reachable Set). *A set $\mathcal{R}$ is said to be a probabilistic i-step reachable set (i-step PRS) of probability level p if*

$$\Pr(e_i \in \mathcal{R} \mid e_0 = 0) \geq p.$$

Given i-step PRS $\mathcal{R}^x$ of probability level p_x for the state error e_i^x and similarly $\mathcal{R}^u$ for the input e_i^u, we can define tightened constraints on μ_i^x and μ_i^u as

$$\mu_i^x \in \mathcal{Z} = \mathcal{X} \ominus \mathcal{R}^x, \tag{15a}$$
$$\mu_i^u \in \mathcal{V} = \mathcal{U} \ominus \mathcal{R}^u, \tag{15b}$$

where $\ominus$ denotes the Pontryagin set difference. Satisfaction of the tightened constraints (15) for the mean thereby implies satisfaction of the original constraints (9d) and (9e), i.e. when $\mu_i^x \in \mathcal{Z}$ we have $\Pr(x_i = \mu_i^x + e_i^x \in \mathcal{X}) \geq \Pr(e_i^x \in \mathcal{R}^x) \geq p_x$.

Under the approximation of a normal distribution of x_i, the uncertainty in each time step is fully specified by the variance matrix Σ_i^x and Σ_i^u. The sets can then be computed as functions of these variances, i.e. $\mathcal{R}^x(\Sigma_i^x)$ and $\mathcal{R}^u(\Sigma_i^x)$, for instance as ellipsoidal confidence regions. The online computation and respective tightening in (15), however, is often computationally prohibitive for general convex constraint sets.

In the following, we present important cases for which a computationally cheap tightening is possible, such that it

can be performed online. For brevity, we concentrate on state constraints, as input constraints can be treated analogously.

1) Half-space constraints: Consider the constraint set $\mathcal{X}$ given by a single half-space constraint $\mathcal{X}^{hs} := \{x \mid h^\mathsf{T} x \leq b\}$, $h \in \mathbb{R}^n$, $b \in \mathbb{R}_+$. Considering the marginal distribution of the error in the direction of the half-space $h^\mathsf{T} e_i^x \sim \mathcal{N}(0, h^\mathsf{T} \Sigma_i^x h)$ enables us to use the quantile function of a standard Gaussian random variable $\phi^{-1}(p_x)$ at the needed probability of constraint satisfaction p_x to see that

$$\mathcal{R}^x(\Sigma_i^x) := \left\{ e \,\middle|\, h^\mathsf{T} e \leq \phi^{-1}(p_x)\sqrt{h^\mathsf{T}\Sigma_i^x h} \right\}$$

is an i-step PRS of probability level p_x. In this case, evaluating the Pontryagin difference in (15) is straightforward and we can directly define the tightened constraint on the state mean

$$\mathcal{Z}^{hs}(\Sigma_i^x) := \left\{ z \,\middle|\, h^\mathsf{T} z \leq b - \phi^{-1}(p_x)\sqrt{h^\mathsf{T}\Sigma_i^x h} \right\}.$$

Remark 2. *A tightening for slab constraints $\mathcal{X}^{sl} = \{x \mid |h^\mathsf{T} x| \leq b\}$ can be similarly derived as*

$$\mathcal{Z}^{sl}(\Sigma_i^x) := \left\{ z \,\middle|\, |h^\mathsf{T} z| \leq b - \phi^{-1}\left(\frac{p_x+1}{2}\right)\sqrt{h^\mathsf{T}\Sigma_i^x h} \right\}.$$

2) Polytopic Constraints: Consider polytopic constraints, given as the intersection of n_j half-spaces

$$\mathcal{X}^p = \left\{ x \,\middle|\, h_j^\mathsf{T} x \leq b_j \,\, \forall j = 1, \ldots, n_j \right\}.$$

Making use of ellipsoidal PRS for Gaussian random variables one can formulate a semidefinite program to tighten constraints [31], [32], which is computationally demanding to solve online. Computationally cheaper approaches typically rely on more conservative bounds. One such bound is given by Boole's inequality

$$\Pr(E_1 \wedge E_2) \leq \Pr(E_1) + \Pr(E_2), \tag{16}$$

which allows for the definition of polytopic PRS based on individual half-spaces. We present two possibilities, the first of which is based on bounding the violation probability of the individual polytope faces, and a second which considers the marginal distributions of e_i^x.

a) PRS based on polytope faces: Similar to the treatment of half space constraints, we can define a PRS on the state error e^x which is aligned with the considered constraints. Using Boole's inequality (16), we have that

$$\mathcal{R}^x(\Sigma_i^x) = \left\{ e \,\middle|\, h_j^\mathsf{T} e \leq \phi^{-1}\left(\frac{p_x}{n_j}\right)\sqrt{h_j^\mathsf{T}\Sigma_i^x h_j} \,\, \forall j = 1, \ldots, n_j \right\}$$

is a PRS of probability level p_x. The tightening results in

$$\mathcal{Z}^p(\Sigma_i^x) = \left\{ z \,\middle|\, h_j^\mathsf{T} z \leq b_j - \phi^{-1}\left(\frac{p_x}{n_j}\right)\sqrt{h_j^\mathsf{T}\Sigma_i^x h_j} \,\, \forall j = 1, \ldots, n_j \right\},$$

which scales with the number of faces of the polytope and can therefore be quite conservative if the polytope has many (similar) faces.

b) PRS based on marginal distributions: Alternatively, one can define a PRS based on the marginal distribution of the error e^x in each dimension, which therefore scales with the state dimension. We use Boole's inequality (16) with Remark 2 directly on marginal distributions in each dimension of the state to define a box-shaped PRS of probability level p_x

$$\mathcal{R}^x(\Sigma_i^x) =$$
$$\left\{ e \left| |[e]_j| \le \phi^{-1}\left(\frac{p_x+1}{2n_x}\right) \sqrt{[\Sigma_i^x]_{j,j}} \; \forall j = 1, \ldots, n_x \right. \right\}.$$

To compute the Pontryagin difference (15) we make use of the following Lemma.

Lemma 1. *Let* $\mathcal{A} = \{x \,|\, Hx \le b\}$ *be a polytope and* $\mathcal{B} = \{e \,|\, -r \le e \le r, \; r \in \mathbb{R}_+^n\}$, *a box in* $\mathbb{R}^n$.
Then $\mathcal{A} \ominus \mathcal{B} = \{x \,|\, Hx \le b - |H|r\}$, *where* $|H|$ *is the element-wise absolute value.*

Proof. From definition of the Pontryagin difference we have

$$
\begin{aligned}
\mathcal{A} \ominus \mathcal{B} &= \{x \,|\, x + e \in \mathcal{A} \; \forall e \in \mathcal{B}\} \\
&= \{x \,|\, H(x+e) \le b \; \forall e \in \mathcal{B}\} \\
&= \left\{ x \left| [H]_{j,.}x \le [b]_j - \max_{e \in \mathcal{B}} [H]_{j,.}e \; \forall j = 1, \ldots, n \right. \right\} \\
&= \{x \,|\, [H]_{j,.}x \le [b]_j - |[H]_{j,.}|r \; \forall j = 1, \ldots, n\} \\
&= \{x \,|\, Hx \le b - |H|r\} ,
\end{aligned}
$$

proving the result. $\qquad\square$

The resulting tightened set is therefore

$$\mathcal{Z}^p(\Sigma_i^x) = \left\{ z \left| Hz \le \tilde{b}(\Sigma_i^x) \right. \right\}$$

with $\tilde{b}(\Sigma_i^x) = b - |H|\phi^{-1}\left(\frac{p_x+1}{2n_x}\right)\sqrt{\mathrm{diag}(\Sigma_i^x)}$, where $\mathrm{diag}(\cdot)$ is the vector of diagonal elements and the square root of the vector is taken element wise.

Remark 3. *Treating polytopic constraints through* (16) *can lead to undesired and conservative individual constraints [33]. Practically, it can therefore be beneficial to constrain the probability of violating each individual half-space constraint.*

E. Cost Function

Given the approximate joint normal distribution of state and input, the cost function (9a) can be evaluated using standard stochastic formulations. For simplicity, we focus here on costs in form of expected values, which encompass most stochastic objectives typically considered. While the expected value for general cost functions needs to be computed numerically, there exist a number of functions for which evaluation based on mean and variance information can be analytically expressed and is computationally cheap. The most prominent example for tracking tasks is a quadratic cost on states and inputs

$$l(x_i - x_i^r, u_i - u_i^r) = \|x_i - x_i^r\|_Q^2 + \|u_i - u_i^r\|_R^2 \quad (17)$$

with appropriate weight matrices Q and R, typically satisfying $Q \succeq 0$ and $R \succ 0$, resulting in

$$
\begin{aligned}
\mathbb{E}\left(l(x_i - x_i^r, u_i - u_i^r)\right) = \\
\|\mu_i^x - x_i^r\|_Q^2 + \mathrm{tr}(Q\Sigma_i^x) + \|\mu_i^u - u_i^r\|_R^2 + \mathrm{tr}(R\Sigma_i^u).
\end{aligned}
$$

Further examples include a saturating cost [8], risk sensitive costs [34] or radial basis function networks [17].

We refer to the evaluation of the expected value (9a) in terms of mean and variance as

$$\mathbb{E}\left(l(x_i - x_i^r, u_i - u_i^r)\right) = c(\mu_i^x - x_i^r, \mu_i^u - u_i^r, \Sigma_i^x)$$

and similarly $c_f(\mu_N^x - x_N^r, \Sigma_N^x)$ for the terminal cost.

Remark 4. *While many performance indices can be expressed as expected values, there exist various stochastic cost measures, such as conditional value-at-risk. Given the approximate state distributions, many can be similarly considered.*

F. Tractable MPC Formulation with GP Model

By bringing together the approximations of the previous sections, the following tractable approximation of the MPC problem (9) can be derived:

$$
\begin{aligned}
\min_{\{\mu_i^u\}} \quad & c_f(\mu_N^x - x_N^r, \Sigma_N^x) + \sum_{i=0}^{N-1} c(\mu_i^x - x_i^r, \mu_i^u - u_i^r, \Sigma_i^x) \\
\text{s.t.} \quad & \mu_{i+1}^x = f(\mu_i^x, \mu_i^u) + B_d\mu_i^d, \\
& \Sigma_{i+1}^x = [\nabla f(\mu_i^x, \mu_i^u) \; B_d]\Sigma_i[\nabla f(\mu_i^x, \mu_i^u) \; B_d]^\mathsf{T}, \\
& \mu_{i+1}^x \in \mathcal{Z}(\Sigma_{i+1}^x), \\
& \mu_i^u \in \mathcal{V}(\Sigma_i^x), \\
& \mu_i^d, \Sigma_i \text{ according to (10) and (12), (13) or (14),} \\
& \mu_0^x = x(k), \Sigma_0^x = 0
\end{aligned}
$$
$$(18)$$

for $i = 0, \ldots, N-1$. The resulting optimal control law is obtained in a receding horizon fashion as $\kappa(x(k)) = \mu_0^{u*}$, where μ_0^{u*} is the first element of the optimal control sequence $\{\mu_0^{u*}, \ldots, \mu_N^{u*}\}$ obtained from solving (18) at state $x(k)$.

The presented formulation of the MPC problem with a GP-based model results in a non-convex optimization problem. Assuming twice differentiability of kernel and prior mean function, second-order derivative information of all quantities is available. Problems of this form can typically be solved to local optima using Sequential Quadratic Programming (SQP) or nonlinear interior-point methods [35]. There exist a number of general-purpose solvers that can be applied to solve this class of optimization problems, e.g. the openly available IPOPT [36]. In addition, there are specialized solvers that exploit the structure of predictive control problems, such as ACADO [37] or FORCES Pro [38], [39], which implements a nonlinear interior point method. Derivative information can be automatically generated using automated differentiation tools, such as CASADI [40].

In the following, we demonstrate the proposed algorithm and its properties using one simulation and one experimental example. The first is an illustrative example of an autonomous underwater vehicle (AUV) which, around a trim point, is well described by nominal linear dynamics but is subject to nonlinear friction effects for larger deviations. The second example is a hardware implementation demonstrating the approach for autonomous racing of miniature race cars which are described by a nonlinear nominal model. While model complexity and small sampling times require additional approximations in this

second example, we show that we can leverage key benefits of the proposed approach in a highly demanding real-world application.

IV. ONLINE LEARNING FOR AUTONOMOUS UNDERWATER VEHICLE

We consider the depth control of an autonomous underwater vehicle (AUV). The vehicle is described by a nonlinear continuous-time model of the vehicle at constant surge velocity as ground truth and for simulation purposes [41]. Around a trim point of purely horizontal movement, the states and input of the system are

$[x]_1$ the pitch angle relative to trim point [rad],
$[x]_2$ the heave velocity relative to trim point [m/s],
$[x]_3$ the pitch velocity [rad/s],
u the stern rudder deflection around trim point [rad].

We assume that an approximate linear system model is given, which in practice can be established using methods of linear system identification around the trim point of the system. Using a zero-order hold discretization of the linear part of the model with $T_s = 100$ ms we obtain

$$x(k+1) = Ax(k) + Bu(k) + B_d\left(g(x(k)) + w(k)\right),$$

which is in the form of (1). The nonlinearity results from friction effects when moving in the water, which is therefore modeled as only affecting the (continuous time) velocity states,

$$B_d = \begin{bmatrix} 0 & \frac{T_s^2}{2} \\ T_s & 0 \\ 0 & T_s \end{bmatrix}, \ g(x(k)) = g([x(k)]_2, [x(k)]_3) : \mathbb{R}^2 \to \mathbb{R}^2.$$

Note that the eigenvalues of A are given by $\lambda = \{1.1, 1.03, 1, 0.727\}$, i.e. the linear system has one integrating and two unstable modes.

In the considered scenario, the goal is to track reference changes from the original zero set point to a pitch angle of $30°$, back to zero and then to $45°$. We furthermore consider a safety state constraint on the pitch angle of at least $10°$ below the reference, as well as input constraints corresponding to $\pm 20°$ rudder deflection. In this example, we treat the case of *online learning*, that is we start without any data about the nonlinearity g, collect measurement data during operation and enhance performance online.

A. GP-based Reference Tracking Controller

GP data $\mathcal{D} = \{\mathbf{y}, \mathbf{z}\}$ is generated by calculating the deviation of the linear model from the measured states, as described in Section II-C, where the input data is reduced to the velocity states $z_j = [[x_j]_2, [x_j]_3]^\mathsf{T}$. Data is continuously updated during the closed-loop run. Specifically, we consider a new data point every 5 time steps, and keep track of 30 data points by discarding the oldest when adding a new point. The training data is initialized with 30 data points of zero input and zero output. We employ a squared exponential kernel (3) for both output dimensions with fixed hyperparameters $L_1 = L_2 = \text{diag}([0.35, 0.15]^\mathsf{T})$, and variances $\sigma_{f,1}^2 = 0.04$ and $\sigma_{f,2}^2 = 0.25$.
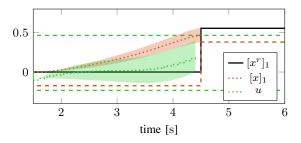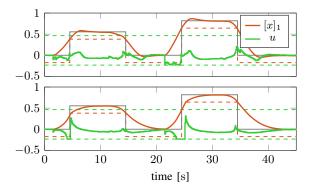

Fig. 4: Predicted pitch angle $[x]_1$ and rudder deflection u. Dotted lines are the mean prediction and shaded the 2-σ confidence region. The dashed lines show the corresponding state and input constraints, while the black line is the pitch angle reference.
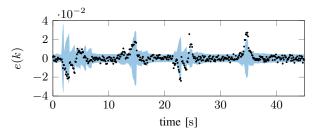
We use a quadratic stage cost as in (17), with weight matrices $Q = \text{diag}([1, 0, 10, 0.5^\mathsf{T}])$, $R = 20$ and a prediction horizon $N = 35$ to track a pitch angle reference. The terminal cost P is chosen according to the solution of the associated discrete-time algebraic Riccati equation, i.e. the LQR cost. As ancillary linear controller K_i, an infinite horizon LQR control law based on the linear nominal model is designed using the same weights as in the MPC and used in all prediction steps i. This stabilizes the linear system and reduces uncertainty growth over the prediction horizon. To propagate the uncertainties associated with the GP we make use of the *Taylor Approximation* outlined in Section III-B2. Constraints are introduced based on Remark 3 considering a maximum probability of individual constraint violation of 2.28%, corresponding to a 2-σ confidence bound. The MPC optimization problem (18) is solved using FORCES Pro [38], [39].

B. Results

Fig. 4 shows the prediction of the GP-based MPC for the first reference change from $0°$ to $30°$. Since no data was collected on the state trajectory necessary for this change, the predicted state and input trajectories are uncertain and a safety margin to the state constraint at the end of the horizon is enforced. The resulting closed-loop trajectory, during which the system learns from online data, is displayed in the top plot of Fig. 5a. For comparison, we run the same simulation with a soft constrained linear MPC formulation with the same cost function, which does not consider nonlinearities or uncertainties in the dynamics, shown in the bottom plot. The results demonstrate improved performance of the GP-based MPC over the linear MPC, especially with regard to constraint satisfaction. The constraint on minimum pitch angle is violated under the linear MPC control law during both reference changes, even though the soft constraint is chosen as an exact penalty function such that constraints are always satisfied, if possible. Fig. 5b exemplifies the residual model error during the closed-loop simulation applying the GP-based controller, as well as the predicted 2-σ residual error bound of the GP. We observe the largest error during reference changes, which are anticipated by the GP-uncertainty, overall matching the resulting residual errors well.

(a) GP-based MPC (top) and linear MPC (bottom). The solid gray line shows the reference value of the pitch angle, dashed in the respective color the state and input constraints.

(b) Residual model error with GP-based MPC. Measured model error as black dots, 2-σ residual model uncertainty from GP shaded blue.

Fig. 5: Simulation results of GP-based MPC for autonomous underwater vehicle in an online learning scenario.

V. AUTONOMOUS RACING

As a second example we consider an autonomous racing scenario, in which the goal is to drive a car around a track as quickly as possible, while keeping the vehicle safe, i.e. while avoiding collision with the track boundaries. The controller is based on a model predictive contouring control formulation [42], [43] which has been applied to the problem of autonomous racing in [44]. Preliminary simulations results were published in [28].

A. Car Dynamics

The race cars are modeled by continuous-time nominal dynamics $\dot{x} = f^c(x, u)$ obtained from a bicycle model with nonlinear tire forces given by a simplified Pacejka tire model [45], which results in the following states and inputs

$$x = [X, Y, \Phi, v_x, v_y, \omega]^\mathsf{T}, \ u = [p, \delta]^\mathsf{T},$$

with position $x^{XY} = [X, Y]^\mathsf{T}$, orientation Φ, longitudinal and lateral velocities v_x and v_y, and yaw rate ω. The inputs to the system are the motor duty cycle p and the steering angle δ. For details on the system modeling please refer to [44], [28].

For use in the MPC formulation, we discretize the system using a Runge-Kutta method with a sampling time of $T_s = 20\,\mathrm{ms}$. In order to account for model mismatch due to inaccurate parameter choices and limited fidelity of this simple model, we add $g(x, u)$ capturing unmodeled dynamics, as well as additive Gaussian white noise w. Due to the structure of the nominal model, i.e. since the dynamics of the first three states

are given purely by kinematic relationships, we assume that the model uncertainty, as well as the process noise w, only affect the velocity states v_x, v_y and ω of the system. From physical considerations, we can also infer that the unmodeled dynamics do not depend on the position states, i.e. we assume

$$B_d = [0 \ I]^\mathsf{T}, \ g(x, u) = g(v_x, v_y, \omega, p, \delta) : \mathbb{R}^5 \to \mathbb{R}^3,$$

resulting in

$$x(k+1) = f(x(k), u(k)) + B_d(g(x(k), u(k)) + w(k)).$$

The system is subject to input constraints $\mathcal{U}$, i.e. the steering angle is limited to lie in $\pm\delta_{\max}$ and the duty cycle has to lie $[-0.1, 1]$, where the negative values correspond to negative applied torques, i.e. breaking. Additionally, operation requires the vehicle to stay on track, which is expressed as a state constraint $\mathcal{X}$ on the car's position.

B. GP-based Racing Controller

We consider a race track given by its centerline and a fixed track width. The centerline is described by a piecewise cubic spline polynomial, which is parameterized by the path length Θ. Progress along the track is characterized by Θ_i, which is introduced as an additional state and enters the considered cost function linearly, encouraging a maximization of progress along the track. Given a Θ_i, we can evaluate the corresponding centerline position $C(\Theta) = [X_c(\Theta), Y_c(\Theta)]^\mathsf{T}$, such that the constraint for the car to stay within the track boundaries can be expressed as

$$\mathcal{X}(\Theta_i) := \{x^{XY} \mid \|x^{XY} - C(\Theta_i)\| \le r\} \subset \mathbb{R}^2,$$

where r is half the track width.

Based on approximate Gaussian distributions of the state in prediction we have for the position error $e_i^{XY} = x_i^{XY} - \mu_i^{XY} \sim \mathcal{N}(0, \Sigma_i^{XY})$ and define an ellipsoidal PRS as

$$\mathcal{R}^{ell}(\Sigma_i^{XY}) = \left\{ e^{XY} \left| (e^{XY})^\mathsf{T}(\Sigma_i^{XY})^{-1} e^{XY} \le \chi_2^2(p_x) \right. \right\},$$

where $\chi_2^2(p_x)$ is the quantile function of the chi-squared distribution with two degrees of freedom. In order to simplify constraint tightening of $\mathcal{X}(\Theta_i)$ we find an outer approximation by a ball $\mathcal{R}^b \subseteq \mathcal{R}^{ell}$ as

$$\mathcal{R}^b(\Sigma_i^{XY}) = \left\{ e^{XY} \left| \|e^{XY}\| \le \sqrt{\lambda_{\max}(\Sigma_i^{XY}) \chi_2^2(p_x)} \right. \right\},$$
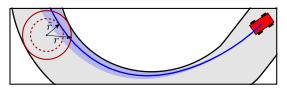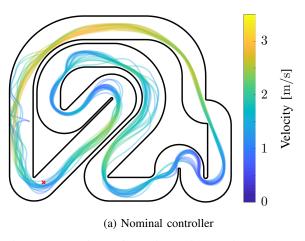

Fig. 6: Illustration of the constraint tightening procedure. The effective track radius is adjusted based on the predicted position uncertainty.

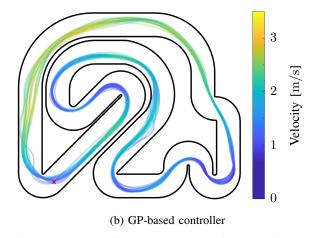(a) Nominal controller

(b) GP-based controller

Fig. 7: Comparison of racelines with nominal and GP-based controller. The color indicates the 2-norm of the velocity, the red cross is the starting point of the race car.

where $\lambda_{\max}(\cdot)$ is the maximum eigenvalue which can be readily computed since Σ_i^{XY} is a 2 by 2 matrix. The necessary constraint tightening can therefore be expressed as

$$\mathcal{Z}(\Theta_i, \Sigma_i^{XY}) = \mathcal{X}(\Theta_i) \ominus \mathcal{R}^b(\Sigma_i^{XY}) \quad (19)$$
$$= \left\{ z^{XY} \,\middle|\, \left\| z^{XY} - C(\Theta_i) \right\| \leq \tilde{r}\left(\Sigma_i^{XY}\right) \right\},$$

where $\tilde{r}\left(\Sigma_i^{XY}\right) = r - \sqrt{\chi_2^2(p_x)\lambda_{\max}\left(\Sigma_i^{XY}\right)}$. Fig. 6 exemplifies the predicted evolution of the cars position and the resulting constraint tightening.

The state is extended by previously applied inputs and large input changes are additionally penalized. We make use of the Taylor approximation (13) to propagate uncertainties without the use of an ancillary state feedback controller, i.e. $K = 0$. The prediction horizon is chosen as $N = 30$ and we formulate the chance constraints (19) with $\chi_2^2(p_x) = 1$. To reduce conservatism of the controller, constraints are instead only tightened for the first 20 prediction steps and are applied to the mean for the remainder of the prediction horizon, similar to the method used in [46]. We reduce computation times by making use of the dynamic sparse approximations with 10 inducing points as outlined in Section III-C, placing the inducing inputs regularly along the previous solution trajectory.

To ensure real-time feasibility of the approach for the sampling time of 20ms two additional approximations are applied. The variance dynamics are pre-evaluated based on the previous MPC solution, which enables the pre-computation of state constraints (19) such that they remain fixed during optimization. Additionally, we neglect the mean prediction of the lateral velocity error v_y since the state is difficult to estimate reliably and the error generally small, such that we observed no improvement when including it in the control formulation.

C. Results

We start out racing the car using the nominal controller, i.e. the controller without an added GP term, which therefore does not consider uncertainties for constraint tightening. Since the nominal model is not well tuned, driving behavior is somewhat erratic and there are a number of small collisions with the track boundaries. This can be observed in the racelines plotted in Fig. 7a showing 20 laps run with the nominal controller. Using the collected data, we train the GP error model d using 325 data points. We infer the hyperparameters in (3) as well as the noise level Σ_w using maximum likelihood optimization. The resulting racelines of 20 laps with the GP-based controller are displayed in Figure 7b, generally showing a much more consistent and safe racing behavior. In particular, it can be seen that almost all of the systematic and persistent problems in the raceline of the nominal controller can be alleviated.

Fig. 8 shows the encountered dynamics error in the yaw-rate and the predicted error during the first lap with the sparse GP-based controller. Mean and residual uncertainty predicted by the GP matches the encountered errors well. It is important to note that the apparent volatility in the plot is not due to overfitting, but is instead due to fast changes in the input and matches the validation data, i.e. the measured errors. To quantify performance of the proposed controllers we compare in Table I average lap time $\overline{T}_l$, minimum lap time $T_{l,\min}$ as well as the average 2-norm error in the system dynamics $\overline{\|e\|}$, i.e. the difference between the mean state after one prediction
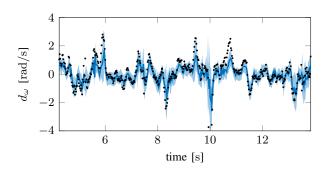

Fig. 8: Dynamic sparse GP compensation of the yaw-rate error with 10 inducing inputs during the first race lap. The black dots show the measured error on the yaw rate at each time step, while the blue line shows the error predicted by the GP. The shaded region is the 2-σ confidence interval.

TABLE I: Experimental results

Controller	$\overline{T}_l$ [s]	$T_{l,\min}$ [s]	$\overline{\|e\|}$ [−]	$\overline{T}_c$ [ms]	$T_c < 20\,\text{ms}$
Nominal	10.32	9.65	0.73	18.2	76.3%
GP-based	9.61	9.27	0.33	17.2	99.8%

step and the realized state, $e(k+1) = \mu_1^x - x(k+1)$. We see that the GP-based controller is able to improve significantly on all these quantities, with an average lap time improvement of 0.71 s, or almost 7%, which constitutes a large improvement in the considered racing task. This is in part due to the improved system model, as evident in the average dynamics error $\overline{\|e\|}$, but also due to the cautious nature of the controller, which helps to further reduce collisions and large problems in the raceline. Due to the cautious nature, the minimum lap time gains are slightly less pronounced. In fact, the nominal controller consistently displays higher top speeds, which often times, however, lead to significant problems at the breakpoint to a slow corner. Computation times are reported as average solve times $\overline{T}_c$ and the percentage of solutions in under 20 ms, $T_c < 20$ ms. Average solution times of nominal and GP-based controller are similar. The percentiles of solutions in under 20 ms, however, differ significantly. This is mainly due to frequent large re-planning actions occurring with the nominal controller.

The results therefore demonstrate that the presented GP-based controller can significantly improve performance while maintaining safety in a hardware implementation of a complex system with small sampling times.

VI. CONCLUSION

The paper discussed the use of Gaussian process regression to learn nonlinearities for improved performance in model predictive control. Combining GP dynamics with a nominal system allows for learning only parts of the dynamics, which is key for keeping the required number of data points and computational complexity of the GP regression feasible for online control. Approximation methods for the propagation of the state distributions over the prediction horizon were reviewed and it was shown how this enables a principled treatment of chance constraints on both states and inputs.

A simulation example as well as a hardware implementation have shown how the proposed formulations provide cautious control with improved performance for medium-sized systems with low sampling times. In particular, we have demonstrated in experiment that both performance and safety in an autonomous racing setting can be significantly improved by using cautious data-driven techniques.

REFERENCES

[1] K. J. Hunt, D. Sbarbaro, R. Zbikowski, and P. J. Gawthrop, "Neural networks for control systems-A survey," *Automatica*, vol. 28, no. 6, pp. 1083–1112, 1992.

[2] D. Nguyen-Tuong and J. Peters, "Model learning for robot control: a survey," *Cognitive Processing*, vol. 12, no. 4, pp. 319–340, Nov. 2011.

[3] O. Sigaud, C. Salan, and V. Padois, "On-line regression algorithms for learning mechanical models of robots: A survey," *Robotics and Autonomous Systems*, vol. 59, no. 12, pp. 1115–1129, 2011.

[4] G. Pillonetto, F. Dinuzzo, T. Chen, G. De Nicolao, and L. Ljung, "Kernel methods in system identification, machine learning and function estimation: A survey," *Automatica*, vol. 50, no. 3, pp. 657–682, 2014.

[5] E. D. Klenske, M. N. Zeilinger, B. Schölkopf, and P. Hennig, "Gaussian process-based predictive control for periodic error correction," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 1, pp. 110–121, 2016.

[6] J. Quionero-Candela, A. Girard, and C. Rasmussen, "Prediction at an uncertain input for Gaussian processes and relevance vector machines - application to multiple-step ahead time-series forecasting," Max Planck Institute for Biological Cybernetics, Tübingen, Germany, Tech. Rep. IMM-2003-18, 2003.

[7] M. Kuß, "Gaussian process models for robust regression, classification, and reinforcement learning," Ph.D. dissertation, Technische Universität Darmstadt, Darmstadt, 2006.

[8] M. Deisenroth, "Efficient reinforcement learning using Gaussian processes," Ph.D. dissertation, Karlsruhe Institute for Technology, Karlsruhe, 2010.

[9] J. Kocijan, R. Murray-Smith, C. E. Rasmussen, and A. Girard, "Gaussian process model based predictive control," in *American Control Conference*, Jun. 2004, pp. 2214–2219.

[10] A. Grancharova, J. Kocijan, and T. A. Johansen, "Explicit stochastic predictive control of combustion plants based on Gaussian process models," *Automatica*, vol. 44, no. 6, pp. 1621–1631, 2008.

[11] G. Cao, E. M.-k. Lai, and F. Alam, "Gaussian process model predictive control of unmanned quadrotors," in *International Conference on Control, Automation and Robotics*, 2016.

[12] X. Yang and J. M. Maciejowski, "Fault tolerant control using Gaussian processes and model predictive control," *International Journal of Applied Mathematics and Computer Science*, vol. 25, no. 1, pp. 133–148, 2015.

[13] C. J. Ostafew, A. P. Schoellig, T. D. Barfoot, and J. Collier, "Learning-based nonlinear model predictive control to improve vision-based mobile robot path tracking," *Journal of Field Robotics*, vol. 33, no. 1, pp. 133–152, 2016.

[14] 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, pp. 1547–1563, 2016.

[15] G. Cao, E. M.-K. Lai, and F. Alam, "Gaussian process model predictive control of unknown non-linear systems," *IET Control Theory & Applications*, vol. 11, pp. 703–713(10), 2017.

[16] Y. Wang, C. Ocampo-Martinez, and V. Puig, "Stochastic model predictive control based on Gaussian processes applied to drinking water networks," *IET Control Theory Applications*, vol. 10, no. 8, pp. 947–955, 2016.

[17] S. Kamthe and M. P. Deisenroth, "Data-efficient reinforcement learning with probabilistic model predictive control," in *International Conference on Artificial Intelligence and Statistics*, 2018.

[18] L. Hewing and M. N. Zeilinger, "Stochastic model predictive control for linear systems using probabilistic reachable sets," *Conference on Decision and Control*, 2018, *accepted*.

[19] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*. The MIT Press, 2006.

[20] J. Quiñonero-Candela, C. E. Rasmussen, and C. K. Williams, "Approximation methods for Gaussian process regression," Microsoft Research, Tech. Rep. MSR-TR-2007-124, 2007.

[21] J. Quiñonero-Candela, C. E. Rasmussen, and R. Herbrich, "A unifying view of sparse approximate Gaussian process regression," *Journal of Machine Learning Research*, vol. 6, pp. 1935–1959, 2005.

[22] E. Snelson and Z. Ghahramani, "Sparse gaussian processes using pseudo-inputs," in *Advances in Neural Information Processing Systems*, Y. Weiss, B. Schölkopf, and J. C. Platt, Eds., 2006, pp. 1257–1264.

[23] V. Tresp, "A Bayesian committee machine," *Neural Computation*, vol. 12, no. 11, pp. 2719–2741, Nov. 2000.

[24] A. Bemporad and M. Morari, "Robust model predictive control: A survey," *Robustness in identification and control*, vol. 245, pp. 207–226, 1999.

[25] J. Rawlings and D. Mayne, *Model Predictive Control: Theory and Design*. Nob Hill Pub., 2009.

[26] T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause, "Learning-based model predictive control for safe exploration and reinforcement learning," *arXiv:0707.3168*, 2018.

[27] A. Girard, C. E. Rasmussen, and R. Murray-Smith, "Gaussian process priors with uncertain inputs : Multiple-step-ahead prediction," Department of Computing Science, University of Glasgow, Glasgow, Tech. Rep., 2002.

[28] L. Hewing, A. Liniger, and M. N. Zeilinger, "Cautious NMPC with Gaussian process dynamics for miniature race cars," *European Control Conference*, 2018.

[29] E. Kofman, J. A. De Don, and M. M. Seron, "Probabilistic set invariance and ultimate boundedness," *Automatica*, vol. 48, no. 10, pp. 2670–2676, 2012.

[30] L. Hewing, A. Carron, K. Wabersich, and M. N. Zeilinger, "On a correspondence between probabilistic and robust invariant sets for linear systems," *European Control Conference*, 2018.

[31] Z. Zhou and R. Cogill, "Reliable approximations of probability-constrained stochastic linear-quadratic control," *Automatica*, vol. 49, no. 8, pp. 2435 – 2439, 2013.

[32] G. Schildbach, P. Goulart, and M. Morari, "Linear controller design for chance constrained systems," *Automatica*, vol. 51, pp. 278 – 284, 2015.

[33] M. Lorenzen, F. Dabbene, R. Tempo, and F. Allgwer, "Constraint-tightening and stability in stochastic model predictive control," *IEEE Transactions on Automatic Control*, vol. 62, no. 7, pp. 3165–3177, 2017.

[34] P. Whittle, "Risk-sensitive linear/quadratic/gaussian control," *Advances in Applied Probability*, vol. 13, no. 4, p. 764777, 1981.

[35] M. Diehl, H. J. Ferreau, and N. Haverbeke, "Efficient numerical methods for nonlinear MPC and moving horizon estimation problem formulation," *Nonlinear model predictive control*, pp. 391–417, 2009.

[36] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[37] B. Houska, H. J. Ferreau, and M. Diehl, "ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization," *Optimal Control Applications and Methods*, vol. 32, no. 3, pp. 298–312, 2011.

[38] A. Domahidi and J. Jerez, "FORCES Professional," embotech GmbH (http://embotech.com/FORCES-Pro), Jul. 2014.

[39] A. Zanelli, A. Domahidi, J. Jerez, and M. Morari, "FORCES NLP: an efficient implementation of interior-point methods for multistage nonlinear nonconvex programs," *International Journal of Control*, 2017.

[40] J. Andersson, "A general-purpose software framework for dynamic optimization," Ph.D. dissertation, KU Leuven, 2013.

[41] M. S. Naik and S. N. Singh, "State-dependent Riccati equation-based robust dive plane control of AUV with control constraints," *Ocean Engineering*, vol. 34, no. 11-12, pp. 1711–1723, 2007.

[42] T. Faulwasser, B. Kern, and R. Findeisen, "Model predictive path-following for constrained nonlinear systems," *Conference on Decision and Control and Chinese Control Conference*, pp. 8642–8647, 2009.

[43] D. Lam, C. Manzie, and M. Good, "Model predictive contouring control," in *Conference on Decision and Control*, 2010, pp. 6137–6142.

[44] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1:43 scale RC cars," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 628–647, 2015.

[45] H. B. Pacejka and E. Bakker, "The magic formula tyre model," *Vehicle System Dynamics*, vol. 21, no. sup001, pp. 1–18, 1992.

[46] J. V. Carrau, A. Liniger, X. Zhang, and J. Lygeros, "Efficient implementation of randomized MPC for miniature race cars," *European Control Conference*, pp. 957–962, 2016.