Cautious NMPC with Gaussian Process Dynamics for Autonomous Miniature Race Cars

Author(s):
Hewing, Lukas; Liniger, Alexander; Zeilinger, Melanie N.

Publication date:
2018

Permanent link:
https://doi.org/10.3929/ethz-b-000265761

Originally published in:
https://doi.org/10.23919/ECC.2018.8550162

Funding acknowledgement:
157601 - Safety and Performance for Human in the Loop Control (SNF)

2018 European Control Conference (ECC)
June 12-15, 2018. Limassol, Cyprus

Cautious NMPC with Gaussian Process Dynamics for Autonomous Miniature Race Cars

Lukas Hewing[1], Alexander Liniger[2] and Melanie N. Zeilinger[1]

Abstract— **This paper presents an adaptive high performance control method for autonomous miniature race cars. Racing dynamics are notoriously hard to model from first principles, which is addressed by means of a cautious nonlinear model predictive control (NMPC) approach that learns to improve its dynamics model from data and safely increases racing performance. The approach makes use of a Gaussian Process (GP) and takes residual model uncertainty into account through a chance constrained formulation. We present a sparse GP approximation with dynamically adjusting inducing inputs, enabling a real-time implementable controller. The formulation is demonstrated in simulations, which show significant improvement with respect to both lap time and constraint satisfaction compared to an NMPC without model learning.**

I. Introduction

Control of autonomous cars is a challenging task and has attracted considerable attention in recent years [1]. One particular case of autonomous driving is autonomous racing, where the goal is to drive around a track as fast as possible, potentially to race against competitors and to avoid collisions [2]. In order to achieve high performance at these extreme conditions, racing teams today spend a significant amount of time and effort on modeling, which is challenging especially near the limits of tire adhesion [3]. Learning-based control methods have been proposed to address this challenge and show great potential towards improving racing performance [4]. They do, however, often suffer from poor model accuracy and performance during transient learning phases. This can lead to violation of critical constraints [5] related to keeping the car on track and avoiding collisions, compromising not only performance, but the success of the entire race. In addition, iteratively learning the racing task on a lap-by-lap basis, as considered e.g. in [6], suffers from poor generalization and does typically not allow for maintaining high performance for dynamic racing tasks, such as obstacle avoidance or overtaking. This paper addresses these challenges by learning the dynamics model from data and considering model uncertainty to ensure constraint satisfaction in a nonlinear model predictive control (NMPC) approach, offering a flexible framework for racing control.

Recently, a number of autonomous racing control methods were presented that rely on NMPC formulations. An NMPC racing approach for miniature race cars was proposed in [7], which uses a contouring control formulation to maximize track progress over a finite horizon and enables obstacle avoidance. It was extended to a stochastic setting in order to take model uncertainty into account in [8] and [9]. Using model learning in an MPC framework allows for generalizing from collected data and for improving performance in varying racing tasks. This was, for instance, demonstrated in [10] by using the mean estimate of a Gaussian Process (GP) as a dynamics model for an NMPC method based on [7]. Furthermore, the MPC approach recently proposed in [11] was applied to the problem of autonomous racing, where the model is improved with an iterative parameter estimation technique [12].

The method presented in this paper makes use of GP regression to improve the dynamics model from measurement data, since GPs inherently provide a measure for residual model uncertainty, which is integrated in a cautious NMPC controller. To this end we extend the approach presented in [7] with a learning module and reformulate the controller in a stochastic setting. A key element differentiating the approach from available results is the stochastic treatment of a GP model in an NMPC controller to improve both performance and constraint satisfaction properties. We derive a tractable formulation of the problem that exploits both the improved dynamics model and the uncertainty and show how chance constraints on the states can be approximated in deterministic form. The framework thereby allows for specifying a minimum probability of satisfying critical constraints, such as track boundaries, offering an intuitive and systematic way of defining a desired trade-off between aggressive driving and safety in terms of collision avoidance.

While the use of GPs in MPC offers many benefits, it poses computational challenges for use with fast sampled and larger scale systems, such as the race car problem, since the evaluation complexity of GPs is generally high and directly scales with the number of data points considered. Various approaches to address this limitation have been presented in the literature. One class of methods relies on an approximation by a finite number of basis functions, such as the sparse spectrum approximation [13], which is also used in the GP-based NMPC in [10]. We present an approach for predictive control based on a sparse GP approximation using inducing inputs [14], which are selected according to an approximate trajectory in state-action space. This enables a high-fidelity local approximation currently relevant for control at a given measured state, and facilitates real-time implementability of the presented controller.

We finally evaluate the proposed cautious NMPC con-

This work was supported by the Swiss National Science Foundation under grant no. PP00P2 157601 / 1.

[1]Institute for Dynamic Systems and Control, ETH Zurich, Zurich, Switzerland lhewing@ethz.ch, mzeilinger@ethz.ch

[2]Institute for Automatic Control, ETH Zurich, Zurich, Switzerland aliniger@ethz.ch

troller in simulations of a race. The results demonstrate that it provides safe and high performance control at sampling times of 30 ms, which is computationally on par with NMPC schemes without model learning [7], while improving racing performance and constraint satisfaction. We furthermore demonstrate robustness towards process noise, indicating fitness for hardware implementation.

II. Preliminaries

In the following we specify the notation used in the paper and briefly introduce GP regression and sparse approximations based on inducing inputs as relevant to the presented control approach.

A. Notation

For two matrices or vectors we use $[A; B] := [A^T\ B^T]^T$ for vertical matrix/vector concatenation. We use $[y]_i$ to refer to the i-th element of the vector y, and similarly $[A]_{\cdot,i}$ for the i-th column of matrix A. A normal distribution with mean μ and variance Σ is denoted $\mathcal{N}(\mu, \Sigma)$. We use $\|x\|$ for the 2-norm of vector x and $\mathrm{diag}(x)$ to express a diagonal matrix with elements given by the vector x. The gradient of a vector-valued function $f : \mathbb{R}^{n_z} \to \mathbb{R}^{n_f}$ with respect to vector $x \in \mathbb{R}^{n_x}$ is denoted $\nabla_x f : \mathbb{R}^{n_z} \to \mathbb{R}^{n_f \times n_x}$.

B. Gaussian Process Regression

Consider M input locations collected in the matrix $\mathbf{z} = [z_1^T; \dots; z_M^T] \in \mathbb{R}^{M \times n_z}$ and corresponding measurements $\mathbf{y} = [y_1^T; \dots; y_M^T] \in \mathbb{R}^{M \times n_d}$ arising from an unknown function $g(z) : \mathbb{R}^{n_z} \to \mathbb{R}^{n_d}$ under the following statistical model

$$y_j = g(z_j) + \omega_j \ , \tag{1}$$

where ω_j is i.i.d. Gaussian noise with zero mean and diagonal variance $\Sigma_w = \mathrm{diag}([\sigma_1^2; \dots; \sigma_{n_d}^2]$. Assuming a GP prior on g in each output dimension $a \in \{1, \dots, n_d\}$, the measurement data is normally distributed with

$$[\mathbf{y}]_{\cdot,a} \sim \mathcal{N}(0, K_{\mathbf{zz}}^a + \sigma_a^2) \ ,$$

where $K_{\mathbf{zz}}^a$ is the Gram matrix of the data points using the kernel function $k^a(\cdot,\cdot)$ on the input locations $\mathbf{z}$, i.e. $[K_{\mathbf{zz}}^a]_{ij} = k^a(z_i, z_j)$. 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 based on observed data [15]. Throughout this paper we consider the squared exponential kernel function

$$k(z, \tilde{z}) = \sigma_f^2 \exp\left(-\frac{1}{2}(z-\tilde{z})^T L^{-1} (z-\tilde{z})\right) \ ,$$

in which $L \in \mathbb{R}^{n_z \times n_z}$ is a positive diagonal length scale matrix. 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

$$p([y]_a, [\mathbf{y}]_{\cdot,a}) \sim \mathcal{N}\left(0, \begin{bmatrix} K_{\mathbf{zz}}^a & K_{\mathbf{z}z}^a \\ K_{z\mathbf{z}}^a & K_{zz}^a \end{bmatrix}\right) \ , \tag{2}$$

where $[K_{\mathbf{z}z}^a]_j = k^a(z_j, z)$, $K_{z\mathbf{z}}^a = (K_{\mathbf{z}z}^a)^T$ and similarly $K_{zz}^a = k^a(z, z)$. The resulting conditional distribution is Gaussian with $p([y]_a \,|\, [\mathbf{y}]_{\cdot,a}) \sim \mathcal{N}(\mu_a^d(z), \Sigma_a^d(z))$ and

$$\mu_a^d(z) = K_{z\mathbf{z}}^a (K_{\mathbf{zz}}^a + I\sigma_a^2)^{-1} [\mathbf{y}]_{\cdot,a} \ , \tag{3a}$$
$$\Sigma_a^d(z) = K_{zz}^a - K_{z\mathbf{z}}^a (K_{\mathbf{zz}}^a + I\sigma_a^2)^{-1} K_{\mathbf{z}z}^a \ . \tag{3b}$$

We call the resulting GP approximation of the unknown function $g(z)$

$$d(z) \sim \mathcal{N}(\mu^d(z), \Sigma^d(z)) \tag{4}$$

with $\mu^d = [\mu_1^d; \dots; \mu_{n_d}^d]$ and $\Sigma^d = \mathrm{diag}([\Sigma_1^d; \dots; \Sigma_{n_d}^d])$.

Evaluating (4) has cost $\mathcal{O}(n_d n_z M)$ and $\mathcal{O}(n_d n_z M^2)$ for mean and variance, respectively and thus scales with the number of data points. For many data points or fast real-time applications this limits the use of a GP model. To overcome these issues, various approximation techniques have been proposed, one class of which is sparse Gaussian processes using inducing inputs [16], briefly outlined in the following.

C. Sparse Gaussian Processes

Most sparse GP approximations can be understood using the concept of inducing targets $\mathbf{y}_{ind}$ at inputs $\mathbf{z}_{ind}$ and an inducing conditional distribution q to approximate the joint distribution (2) by assuming that test points and training data are conditionally independent given $\mathbf{y}_{ind}$ [14]:

$$\begin{aligned} p([y]_a, [\mathbf{y}]_{\cdot,a}) &= \int p([y]_a, [\mathbf{y}]_{\cdot,a} \,|\, \mathbf{y}_{ind}) p(\mathbf{y}_{ind}) \, \mathrm{d}\mathbf{y}_{ind} \\ &\approx \int q([y]_a \,|\, \mathbf{y}_{ind}) q([\mathbf{y}]_{\cdot,a} \,|\, \mathbf{y}_{ind}) p(\mathbf{y}_{ind}) \, \mathrm{d}\mathbf{y}_{ind} \, . \end{aligned}$$

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 [17], or letting them coincide with test points [18].

In this paper, we make use of the state-of-the-art Fully Independent Training Conditional (FITC) approximation to approximate the GP distribution and reduce computational complexity [17]. Given a selection of inducing inputs $\mathbf{z}_{ind}$ and using the shorthand notation $Q_{\zeta\tilde{\zeta}}^a := K_{\zeta \mathbf{z}_{ind}}^a (K_{\mathbf{z}_{ind}\mathbf{z}_{ind}}^a)^{-1} K_{\mathbf{z}_{ind}\tilde{\zeta}}^a$ the approximate posterior distribution is given by

$$\tilde{\mu}_a^d(z) = Q_{z\mathbf{z}}^a (Q_{\mathbf{zz}}^a + \Lambda)^{-1} [\mathbf{y}]_{\cdot,a} \ , \tag{5a}$$
$$\tilde{\Sigma}_a^d(z) = K_{zz}^a - Q_{z\mathbf{z}}^a (Q_{\mathbf{zz}}^a + \Lambda)^{-1} Q_{\mathbf{z}z} \tag{5b}$$

with $\Lambda = \mathrm{diag}(K_{\mathbf{zz}}^a - Q_{\mathbf{zz}}^a + I\sigma_a^2)$. Concatenating the output dimensions similar to (4) we arrive at the approximation

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

Several of the matrices used in (5) can be precomputed such that the evaluation complexity becomes independent of the number of original data points. Using $\tilde{M}$ inducing points, the computational complexity for evaluating the sparse GP at a test point is reduced to $\mathcal{O}(n_d n_z \tilde{M})$ and $\mathcal{O}(n_d n_z \tilde{M}^2)$ for the predictive mean and variance, respectively.

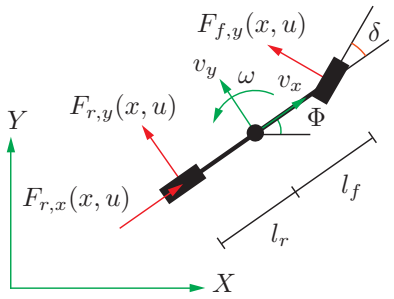

Fig. 1: Schematic of the car model.

III. Race Car Modeling

This section presents the race car setup and nominal modeling of the car dynamics, which will serve as a base model for the learning-based control approach. This is largely based on material presented in [7], which provides a more detailed exposition.

A. Car Dynamics

We consider the following model structure to describe the dynamics of the miniature race cars

$$\dot{x} = f_c(x,u) + B_d(g_c(x,u) + w) \ , \tag{6}$$

where $f_c(x,u)$ are the nominal system dynamics of the car modeled from first principles, and $g_c(x,u)$ reflects unmodeled dynamics. The considered nominal dynamics are obtained from a bicycle model with nonlinear tire forces as shown in Figure 1, resulting in

$$f_c(x,u) = \begin{bmatrix} v_x\cos(\Phi) - v_y\sin(\Phi) \\ v_x\sin(\Phi) + v_y\cos(\Phi) \\ \omega \\ \frac{1}{m}\Big(F_{r,x}(x,u) - F_{f,y}(x,u)\sin\delta + mv_y\omega\Big) \\ \frac{1}{m}\Big(F_{r,y}(x,u) + F_{f,y}(x,u)\cos\delta - mv_x\omega\Big) \\ \frac{1}{I_z}\Big(F_{f,y}(x,u)l_f\cos\delta - F_{r,y}(x,u)l_r\Big) \end{bmatrix} \ , \tag{7}$$

where $x = [X; Y; \Phi; v_x; v_y; \omega]$ is the state of the system, with position (X,Y), 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, i.e., $u = [p; \delta]$. Furthermore, m is the mass, I_z the moment of inertia and l_r and l_f are the distance of the center of gravity from the rear and front tire, respectively. The most difficult components to model are the tire forces $F_{f,y}$ and $F_{r,y}$ and the drivetrain force $F_{r,x}$. The tires are modeled by a simplified Pacejka tire model [19] and the drivetrain using a DC motor model combined with a friction model. For the exact formulations of the forces, we refer to [7].

In order to account for model mismatch due to inaccurate parameter choices and limited fidelity of this simple model, we integrate $g_c(x,u)$ 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, that is $B_d = [0; I_3]$.

For the use in a discrete-time MPC formulation, we finally discretize the system using the Euler forward scheme with a sampling time of T_s, resulting in the following description,

$$x(k+1) = f(x(k),u(k)) + B_d(g(x(k),u(k)) + w(k)), \tag{8}$$

where $w(k)$ is i.i.d. normally distributed process noise with $w(k) \sim \mathcal{N}(0, \Sigma^w)$ and $\Sigma^w = \mathrm{diag}[\sigma^2_{v_x}; \sigma^2_{v_y}; \sigma^2_{\omega}]$, which, together with the uncertain dynamics function g, will be inferred from measurement data.

B. Race Track and Constraints

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 parametrized by the path length Θ. Given a Θ, we can evaluate the corresponding centerline position $(X_c(\Theta), Y_c(\Theta))$ and orientation $\Phi_c(\Theta)$. By letting $\tilde{\Theta}$ correspond to the projection of (X,Y) on the centerline, the constraint for the car to stay within the track boundaries is expressed as

$$\mathcal{X}(\tilde{\Theta}) := \left\{ x \,\middle|\, \left\| \begin{bmatrix} X \\ Y \end{bmatrix} - \begin{bmatrix} X_c(\tilde{\Theta}) \\ Y_c(\tilde{\Theta}) \end{bmatrix} \right\| \leq r \right\} \ , \tag{9}$$

where r is half the track width.

Additionally, the system is subject to input constraints,

$$\mathcal{U} = \left\{ u \,\middle|\, \begin{bmatrix} 0 \\ -\delta_{\max} \end{bmatrix} \leq \begin{bmatrix} p \\ \delta \end{bmatrix} \leq \begin{bmatrix} 1 \\ \delta_{\max} \end{bmatrix} \right\} \ , \tag{10}$$

i.e. the steering angle is limited to a maximal angle $\delta_{\max}$ and the duty cycle has to lie between zero and one.

IV. Learning-based Controller Design

In the following, we first present the model learning module that is subsequently used in a cautious NMPC controller. We briefly state the contouring control formulation [7], serving as the basis for the controller and integrate the learning-based dynamics using a stochastic GP model. Afterwards, we introduce suitable approximations to reduce computational complexity and render the control approach real-time feasible.

A. Model Learning

We apply Gaussian process regression [15] to infer the vector-valued function g of the discrete-time system dynamics (8) from previously collected measurement data of states and inputs. Training data is generated as the deviation to the nominal system model, i.e. for a specific data point:

$$y_j = g(x(j),u(j)) + w(j) = B_d^{\dagger}\left(x(j+1) - f(x(j),u(j))\right) \ ,$$
$$z_j = [x(j); u(j)] \ ,$$

where † is the pseudoinverse. Note that this is in the form of (1) and we can directly apply (3) to derive a GP model $d(x_i,u_i)$ from the data, resulting in the stochastic model

$$x_{i+1} = f(x_i,u_i) + B_d(d(x_i,u_i) + w_i) \ . \tag{11}$$

The state x_i obtained from this model, which will be used in a predictive controller, is given in form of a stochastic distribution.

B. Contouring Control

The learning-based NMPC controller makes use of a contouring control formulation, which has been introduced in [20], [21] and was shown to provide good racing performance in [7]. The objective of the optimal contouring control formulation is to maximize progress along the race track. An approximation of the car position along the centerline is introduced as an optimization variable by including integrator dynamics $\Theta_{i+1} = \Theta_i + v_i$, where Θ_i is a position along the track at time step i and v_i is the incremental progress. The progress along the centerline over the horizon is then maximized by means of the overall incremental progress $\sum_{i=0}^{N} v_i$.

In order to connect the progress variable to the race car's position, Θ_i is linked to the projection of the car on the centerline. This is achieved by minimizing the so-called lag error $\hat{e}^l$ and contouring error $\hat{e}^c$, defined as

$$\begin{aligned}\hat{e}^l(x_i,\Theta_i) = &-\cos(\Phi(\Theta_i))(X_i - X_c(\Theta_i))\\ &-\sin(\Phi(\Theta_i))(Y_i - Y_c(\Theta_i))\,,\\ \hat{e}^c(x_i,\Theta_i) = &\ \ \sin(\Phi(\Theta_i))(X_i - X_c(\Theta_i))\\ &-\cos(\Phi(\Theta_i))(Y_i - Y_c(\Theta_i))\,.\end{aligned}$$

For small contouring error $\hat{e}^c$, the lag error $\hat{e}^l$ approximates the distance between the projection of the car's position and $(X_c(\Theta_i), Y_c(\Theta_i))$, such that a small lag error ensures a good approximate projection. The stage cost function is then formulated as

$$\begin{aligned}l(x_i,u_i,\Theta_i,v_i) =&\|\hat{e}^c(x_i,\Theta_i)\|^2_{q_c} + \|\hat{e}^l(x_i,\Theta_i)\|^2_{q_l}\\ &-\gamma v_i + l_{reg}(\Delta u_i, \Delta v_i)\ . \end{aligned} \tag{12}$$

The term $-\gamma v_i$ encourages the progress along the track, using the relative weighting parameter γ. The parameters q_c and q_l are weights on contouring and lag error, respectively, and $l_{reg}(\Delta u_i, \Delta v_i)$ is a regularization term penalizing large changes in the control input and incremental progress $l_{reg}(\Delta u_i, \Delta v_i) = \|u_i - u_{i-1}\|^2_{R_u} + \|v_i - v_{i-1}\|^2_{R_v}$, with the corresponding weights R_u and R_v.

Based on this contouring formulation, we define a stochastic MPC problem that integrates the learned GP-model (11) and minimizes the expected value of the cost function (12) over a finite horizon of length N:

$$\min_{U,V} \quad \mathbb{E}\left(\sum_{i=0}^{N-1} l(x_i,u_i,\Theta_i,v_i)\right) \tag{13a}$$

$$\text{s.t.} \quad x_{i+1} = f(x_i,u_i) + B_d(d(x_i,u_i) + w_i), \tag{13b}$$

$$\Theta_{i+1} = \Theta_i + v_i, \tag{13c}$$

$$P(x_{i+1} \in \mathcal{X}(\Theta_{i+1})) > 1-\epsilon, \tag{13d}$$

$$u_i \in \mathcal{U}, \tag{13e}$$

$$x_0 = x(k),\ \Theta_0 = \Theta(k)\ , \tag{13f}$$

where $i = 0,\dots,N-1$ and $x(k)$ and $\Theta(k)$ are the current system state and the corresponding position on the centerline. The state constraints are formulated w.r.t. the centerline position at Θ_i as an approximation of the projection of the car position, and are in the form of chance constraints which guarantee that the track constraint (9) is violated with a probability less than $1-\epsilon$.

Solving problem (13) is computationally demanding, especially since the distribution of the state is generally not Gaussian after the first prediction time step. In addition, fast sampling times – in the considered race car setting of about 30 ms – pose a significant challenge for real-time computation. In the following subsections, we present a sequence of approximations to reduce the computational complexity of the GP-based NMPC problem for autonomous racing in (13) and eventually provide a real-time feasible approximate controller that can still leverage the key benefits of learning.

C. Approximate Uncertainty Propagation

At each time step, the GP $d(x_i,u_i)$ evaluates to a stochastic distribution according to the residual model uncertainty, which is then propagated forward in time, rendering the state distributions non-Gaussian. In order to solve (13), we therefore approximate the distributions of the state at each prediction step as a Gaussian, i.e. $x_i \sim \mathcal{N}(\mu_i^x, \Sigma_i^x)$ [22], [23], [24]. The dynamics equations for the Gaussian distributions can be found e.g. through a sigma point transform [25] or a first order Taylor expansion detailed in Appendix I. We make use of the Taylor approximation offering a computationally cheap procedure of sufficient accuracy, resulting in the following dynamics for the mean and variance

$$\mu_{i+1}^x = f(\mu_i^x,u_i) + B_d\mu^d(\mu_i^x,u_i)\ , \tag{14a}$$

$$\Sigma_{i+1}^x = \tilde{A}_i \begin{bmatrix} \Sigma_i^x & \star \\ \nabla_x\mu^d(\mu_i^x,u_i)\Sigma_i^x & \Sigma^d(\mu_i^x,u_i)\end{bmatrix} \tilde{A}_i^T\ , \tag{14b}$$

where $\tilde{A}_i = \begin{bmatrix}\nabla_x f(\mu_i^x,u_i) & B_d\end{bmatrix}$ and the star denotes the corresponding element of the symmetric matrix.

D. Simplified Chance Constraints

The Gaussian approximation of the state distribution allows for a simplified treatment of the chance constraints (13d). They can be approximated as deterministic constraints on mean and variance of the state using the following Lemma.

Lemma 1. *Let n-dimensional random vector $x \sim \mathcal{N}(\mu,\Sigma)$ and the set $\mathcal{B}^{x_c}(r) = \{x \,|\, \|x - x_c\| \le r\}$. Then*

$$\|\mu - x_c\| \le r - \sqrt{\chi_n^2(p)\lambda_{max}(\Sigma)} \Rightarrow \Pr(x \in \mathcal{B}^{x_c}(r)) \ge p,$$

where $\chi_n^2(p)$ is the quantile function of the chi-squared distribution with n degrees of freedom and $\lambda_{max}(\Sigma)$ the maximum eigenvalue of Σ.

Proof. Let $\mathcal{E}_p^x := \{x \,|\, (x-\mu)^T\Sigma^{-1}(x-\mu) \le \chi_n^2(p)\}$ be the confidence region of x at level p, such that $\Pr(x \in \mathcal{E}_p^x) \ge p$. We have $\mathcal{E}_p^x \subseteq \mathcal{E}_p^{\tilde{x}}$ with $\tilde{x} \sim \mathcal{N}(\mu, \lambda_{max}(\Sigma)\, I)$, i.e. $\mathcal{E}_p^{\tilde{x}}$ is an

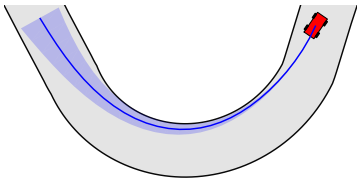

Fig. 2: Planned trajectory with active chance constraints. Shown is the mean trajectory of the car with 1-σ confidence level perpendicular to the car's mean orientation.

outer approximation of the confidence region using the direction of largest variance. Now $\mu \in \mathcal{B}^{x_c}(r - \sqrt{\chi_n^2(p)\lambda_{max}(\Sigma)})$ implies $\mathcal{E}_p^{\tilde{x}} \subseteq \mathcal{B}_c^x(r)$, which means $\Pr(x \in \mathcal{B}_c^x(r)) \geq \Pr(x \in \mathcal{E}_p^{\tilde{x}}) \geq \Pr(x \in \mathcal{E}_p^x) = p$. □

Using Lemma 1, we can formulate a bound on the probability of track constraint violation by enforcing

$$\left\| \begin{bmatrix} \mu_i^X \\ \mu_i^Y \end{bmatrix} - \begin{bmatrix} X_c(\Theta_i) \\ Y_c(\Theta_i) \end{bmatrix} \right\| \leq r - \sqrt{\chi_2^2(p)\lambda_{max}(\Sigma_i^{XY})}, \quad (15)$$

where $\Sigma_i^{XY} \in \mathbb{R}^{2\times 2}$ is the marginal variance of the joint distribution of X_i and Y_i. This procedure is similar to constraint tightening in robust control. Here the amount of tightening is related to an approximate confidence region for the deviation from the mean system state.

Constraint (15) as well as the cost (12) require the variance dynamics. The next section proposes a further simplification to reduce computational cost by considering an approximate evolution of the state variance.

E. Time-Varying Approximation of Variance Dynamics

The variance dynamics in (14b) require $\frac{N}{2}(n^2+n)$ additional variables in the optimization problem and can increase computation time drastically. We trade off accuracy in the system description with computational complexity by evaluating the system variance around an approximate evolution of the state and input. This state-action trajectory can typically be chosen as a reference to be tracked or by shifting a solution of the MPC optimization problem at an earlier time step. Denoting a point on the approximate state-action trajectory with $(\bar{\mu}_i^x, \bar{u}_i)$, the approximate variance dynamics are given by

$$\bar{\Sigma}_{i+1}^x = \bar{A}_i \begin{bmatrix} \bar{\Sigma}_i^x & \star \\ \nabla_x \mu^d(\bar{\mu}_i^x, \bar{u}_i)\bar{\Sigma}_i^x & \Sigma^d(\bar{\mu}_i^x, \bar{u}_i) \end{bmatrix} \bar{A}_i^T$$

with $\bar{A}_i = [\nabla_x f(\bar{\mu}_i^x, \bar{u}_i)\ B_d]$. The variance along the trajectory thus does not depend on any optimization variable and can be computed before the state measurement becomes available at each sampling time. The precomputed variance is then used to satisfy the chance constraints approximately, by replacing Σ^{XY} with $\bar{\Sigma}^{XY}$ in (15). The resulting set is denoted $\bar{\mathcal{X}}(\bar{\Sigma}_i^x, \Theta_i)$. Figure 2 shows an example of a planned trajectory with active chance constraints according to this formulation with $\chi_2^2(p) = 1$.

In the following, we use similar ideas to reduce the computational complexity of the required GP evaluations by dynamically choosing inducing inputs in a sparse GP approximation.

F. Dynamic Sparse GP

Sparse approximations as outlined in Section II-C can considerably speed up evaluation of a GP, with little deterioration of prediction quality. For fast applications with high-dimensional state-input spaces, however, the computational burden can still be prohibitive.

We therefore propose to select inducing inputs locally at each sampling time, which relies on the idea that in MPC the area of interest at each sampling time typically lies close to a known trajectory in the state-action space. Similar to the approximation presented in the previous subsection, inducing inputs can then be selected along the approximate trajectory, e.g. according to a solution computed at a previous time step.

We illustrate the procedure using a two-dimensional example in Figure 3 showing the dynamic approximation for a simple double integrator. Shown is the contour plot of the posterior variance of a GP with two input dimensions x_1 and x_2. Additionally, two trajectories generated from an MPC are shown. The solid red line corresponds to a current prediction trajectory, while the dashed line shows the previous prediction, which is used for local approximation of the GP. As the figure illustrates, full GP and sparse approximation are in close correspondence along the predicted trajectory of the system.

The dynamic selection of local inducing points in a receding horizon fashion allows for an additional speed-up by computing successive approximations adding or removing single inducing points by means of rank 1 updates [26]. These are applied to a reformulation of (5), which offers better numerical properties [14] and avoids inversion of the large matrix $Q_{\mathbf{zz}}^a + \Lambda$,

$$\begin{aligned} \tilde{\mu}_d^a(z) &= K_{z\mathbf{z}}^a \Sigma K_{\mathbf{z}_{ind},\mathbf{z}}^a \Lambda^{-1}[\mathbf{y}]_{\cdot,a}\,, \\ \tilde{\Sigma}_d^a(z) &= K_{zz}^a - Q_{zz}^a + K_{z\mathbf{z}_{ind}}^a \Sigma K_{\mathbf{z}_{ind}z}^a\,, \end{aligned}$$

with $\Sigma = \left(K_{\mathbf{z}_{ind}\mathbf{z}_{ind}}^a + K_{\mathbf{z}_{ind}\mathbf{z}}^a \Lambda^{-1} K_{\mathbf{z}\mathbf{z}_{ind}}^a\right)^{-1}$. Substitution of single inducing points corresponds to a single line and column changing in Σ^{-1}. The corresponding Cholesky factorizations can thus efficiently be updated [27].

G. Resulting Control Formulation for Autonomous Racing

We integrate the approximations presented in the previous sections in the learning-based MPC problem in (13) resulting in the following approximate optimization problem

$$\begin{aligned} \min_{U,V} \quad & \mathbb{E}\left(\sum_{i=0}^{N-1} l(\mu_i^x, u_i, \Theta_i, v_i)\right) && (17a) \\ \text{s.t.} \quad & \mu_{i+1}^x = f(\mu_i^x, u_i) + B_d \tilde{\mu}^d(\mu_i^x, u_i), && (17b) \\ & \Theta_{i+1} = \Theta_i + v_i, && (17c) \\ & \mu_{i+1}^x \in \bar{\mathcal{X}}(\bar{\Sigma}_{i+1}^x, \Theta_{i+1}), && (17d) \\ & u_i \in \mathcal{U}, && (17e) \\ & \mu_0^x = x(k),\ \Theta_0 = \Theta(k)\,, && (17f) \end{aligned}$$

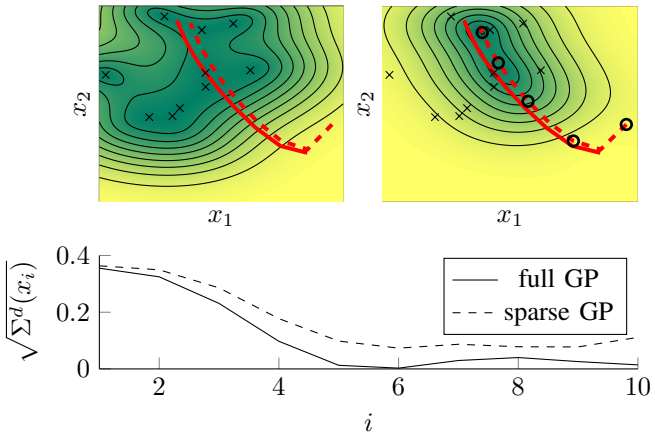

Fig. 3: Contour plots of the posterior variance of a GP for the full GP (top left) and dynamic sparse approximation (top right). The solid red line is the trajectory planned by an MPC, the dashed red line the trajectory of the previous time step used for the approximation, with inducing points indicated by black circles. The bottom plot shows the respective variances along the planned trajectory.

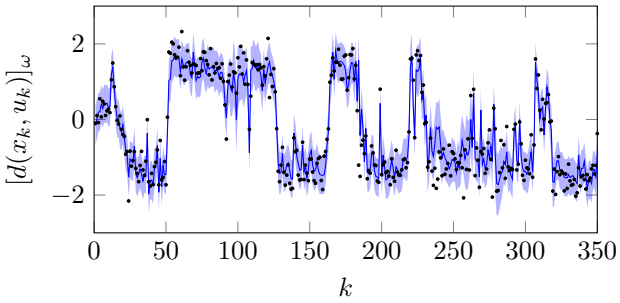

Fig. 4: Prediction of the dynamic sparse GP with 10 inducing inputs during a race lap. Shown as black dots are the error on the yaw rate under process noise as encountered at each time step. The blue line shows the dynamics error predicted by the GP. The shaded region indicates the 2-σ confidence interval, including noise.

where $i = 0, ..., N-1$. By reducing the learned model to the mean GP dynamics and considering approximate variance dynamics and simplified chance constraints, the problem is reduced to a deterministic nonlinear program of moderate dimension.

In the presented form, the approximate optimization problem (17) still requires an optimization over a large spline polynomial corresponding to the entire track. Since evaluation of this polynomial and its derivative is computationally expensive, one can apply an additional approximation step and quadratically approximate the cost function around the shifted solution trajectory from the previous sampling time, for which the expected value is equivalent to the cost at the mean. Similarly, Θ_i can be fixed using the previous solution when evaluating the state constraints (17d), such that the spline can be evaluated separately from the optimization procedure, as done in [7].

V. Simulation

We finally evaluate the proposed control approach in simulations of a race. The race car is simulated using system (6) with g_c resulting from a random perturbation of all parameters of the nominal dynamics f_c by up to $\pm 15\%$ of their original value. We compare two GP-based approaches, one using the full GP $d(x_i, u_i)$ with all available data points and one a dynamic sparse approximation $\tilde{d}(x_i, u_i)$, against a baseline NMPC controller, which makes use of only the nominal part of the model f_c, as well as against a reference controller using the true system model, i.e. with knowledge of g_c.

A. Simulation Setup

We generate controllers using formulation (17), both for the full GP and the dynamic sparse approximation with 10 inducing inputs along the previous solution trajectory of the MPC problem. The inducing points are placed with exponentially decaying density along the previous solution trajectory, putting additional emphasis on the current and near future states of the car. The prediction horizon is chosen as $N = 30$ and we formulate the chance constraints (17d) with $\chi^2_2(p) = 1$. To guarantee feasibility of the optimization problem, we implement the chance constraint using a linear quadratic soft constraint formulation. Specifically, we use slack variables $s_i \geq 0$, which incur additional costs $l_s(s_i) = \|s_i\|^2_{q_s} + c_s s_i$. For sufficiently large c_s the soft constrained formulation is exact, if feasible [28]. To reduce conservatism of the controllers, constraints are only tightened for the first 15 prediction steps and are applied to the mean for the remainder of the prediction horizon, similar to the method used in [8].

The system is simulated for one lap of a race, starting with zero initial velocity from a point on the center-line under white noise of power spectral density $Q_w = \frac{1}{T_s}\,\mathrm{diag}([0.001; 0.001; 0.1])$. The resulting measurements from one lap with the baseline controller are used to generate 350 data-points for both GP-based controllers. Hyperparameters and process noise level were found through likelihood optimization, see e.g. [15].

To exemplify the learned deviations from the nominal system, Figure 4 shows the encountered dynamics error in the yaw-rate and the predicted error during a lap with the sparse GP-based controller. Overall, the learned dynamics are in good correspondence with the true model and the uncertainty predicted by the GP matches the residual model uncertainty and process noise well. Note that the apparent volatility in the plot does not correspond to overfitting, but instead is due to fast changes in the input and matches the validation data.

Solvers were generated using FORCES Pro [29] with a sampling time of $T_s = 30$ ms and the number of maximum solver iterations were limited to 75, which is sufficient to guarantee a solution of required accuracy. All simulations were carried out on a laptop computer with a 2.6 GHz i7-5600 CPU and 12GB RAM.

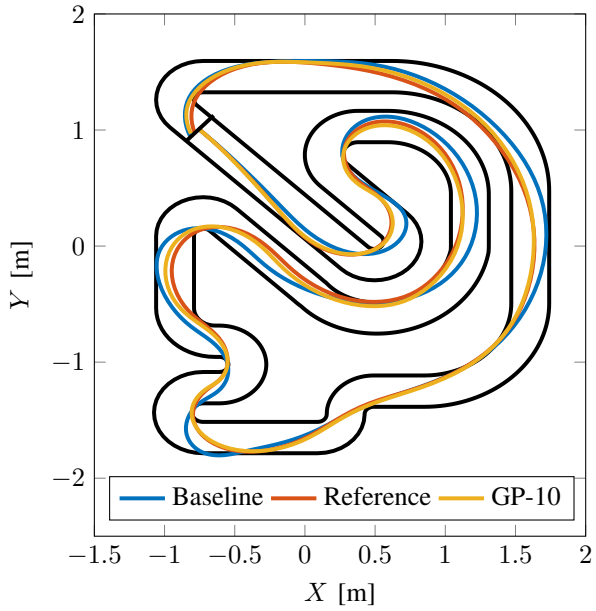

Fig. 5: Resulting trajectories on the race track for simulations without process noise with baseline, reference and sparse GP-based controller.

B. Results

To quantify performance of the proposed controllers we compare the lap time $\overline{T_l}$ and the average squared slack of the realized states $\overline{s_0^2}$ corresponding to state-constraint violations. We furthermore state average solve times $\overline{T_c}$ of the NMPC problem and its 99.9th percentile $T_c^{99.9}$ over the simulation run. To demonstrate the learning performance we also evaluate the average 2-norm error in the system dynamics $\overline{\|e\|}$, i.e. the difference between the mean state after one prediction step and the realized state, $e(k+1) = \mu_1^x - x(k+1)$.

For direct comparison, we first evaluate controller performance in simulations without process noise. As evident in Figure 5, the baseline controller performs visually suboptimally and is unable to guarantee constraint satisfaction, even in the absence of process noise. The reference controller and sparse GP-based controller (GP-10) perform similarly. Table I(a) summarizes the results of the simulations without process noise. We can see that the full GP controller (GP-Full) matches the performance of the reference controller. It also displays only small constraint violations, while the reference controller exhibits some corner cutting behavior leading to constraint violations. This is due to unmodeled discretization error, also evident in the dynamics error of the reference controller. The discretization error is partly learned by the GPs, leading to lower error than even the reference controller. Overall the sparse GP controller demonstrates a performance close to that of the full GP controller, both in terms of lap time and constraint satisfaction and is able to significantly outperform the baseline controller.

Table I(b) shows the averaged simulation for different process noise realizations. The values are averaged over 200 runs, except for $T_c^{99.9}$, which is the 99.9th percentile of all solve times. Qualitatively, the observations for the noise-free case carry over to the simulations in the presence of process noise. Most strikingly, the baseline NMPC controller displays severe constraint violations under noise. In eight cases this even causes the car to completely lose track. The runs were subsequently removed as outliers in Table I(b). All other formulations tolerate the process noise well and achieve similar performance as in the noise-free case. The reference controller achieves slightly faster lap times than the GP-based formulations. These, however, come at the expense of higher constraint violations. Through shaping the allowed probability of violation in the chance constraints (17d), the GP-based formulations allow for a trade-off between aggressive racing and safety.

TABLE I: Simulation results

(a) without process noise

Controller	T_l [s]	$\overline{s_0^2}$ [10^{-3}]	$\overline{\|e\|}$ [-]	$\overline{T_c}$ [ms]	$T_c^{99.9}$ [ms]
Reference	8.64	4.50	0.18	9.4	19.1
Baseline	9.45	4.77	1.20	10.8	20.6
GP-Full	8.67	0.95	0.09	105.2	199.23
GP-10[a]	8.76	1.77	0.16	12.3	26.9

(b) with process noise

Controller	T_l [s]	$\overline{s_0^2}$ [10^{-3}]	$\overline{\|e\|}$ [-]	$\overline{T_c}$ [ms]	$T_c^{99.9}$ [ms]
Reference	8.76	2.88	0.33	9.7	20.8
Baseline[b]	9.55	65.11	1.20	10.1	23.9
GP-Full	8.80	0.68	0.23	102.0	199.4
GP-10[a]	8.90	1.20	0.28	12.1	25.6

[a] Requires an additional ≈ 2.5 ms for sparse approximation.
[b] Eight outliers removed.

The simulations underline the real-time capabilities of the sparse GP-based controller. While the full GP formulation has excessive computational requirements relative to the sampling time of $T_s = 30$ ms, the dynamic sparse formulation is solved in similar time as the baseline formulation. It does, however, require the successive update of the sparse GP formulation, which in our implementation took an additional 2.5 ms on average. Note that this computation can be done directly after the previous MPC solution, whereas the MPC problem is solved after receiving a state measurement at each sample time step. The computation for the sparse approximation thus does not affect the time until an input is applied to the system, which is why we state both times separately. With 99.9% of solve times below 25.6 ms, a computed input can be applied within the sampling time of $T_s = 30$ ms, leaving enough time for the subsequent precomputation of the sparse approximation.

The results demonstrate that the presented GP-based controller can significantly improve performance while maintaining safety, approaching the performance of the reference controller using the true model. They furthermore demonstrate that the controller is real-time implementable and able to tolerate process noise much better than the initial baseline controller. Overall, this indicates fitness for a hardware implementation.

VI. Conclusion

In this paper we addressed the challenge of automatically controlling miniature race cars with an MPC approach under model inaccuracies, which can lead to dramatic failures, especially in a high performance racing environment. The proposed GP-based control approach is able to learn from model mismatch, adapt the dynamics model used for control and subsequently improve controller performance. By considering the residual model uncertainty, we can furthermore enhance constraint satisfaction and thereby safety of the vehicle. Using a dynamic sparse approximation of the GP we demonstrated the real-time capability of the resulting controller and finally showed in simulations that the GP-based approaches can significantly improve lap time and safety after learning from just one example lap.

Appendix I
Uncertainty propagation for nonlinear systems

Let μ_i^x and Σ_i^x denote the mean and variance of x_i, respectively. Using the law of iterated expectation and the law of total variance we have

$$\begin{aligned}
\mu_{i+1}^x &= \mathbb{E}_{x_i}\left(\mathbb{E}_{d|x_i}\left(x_{i+1}\right)\right) \\
&= \mathbb{E}_{x_i}\left(f(x_i,u_i) + B_d\mu^d(x_i,u_i)\right) \\
\Sigma_{i+1}^x &= \mathbb{E}_{x_i}\left(\mathrm{var}_{d|x_i}\left(x_{i+1}\right)\right) + \mathrm{var}_{x_i}\left(\mathbb{E}_{d|x_i}\left(x_{i+1}\right)\right) \\
&= \mathbb{E}_{x_i}\left(B_d\Sigma^d(x_i,u_i)B_d^T\right) \\
&\quad + \mathrm{var}_{x_i}\left(f(x_i,u_i) + B_d\mu^d(x_i,u_i)\right)
\end{aligned}$$

With a first order expansions of f, μ^d and Σ^d around $x_i = \mu_i^x$ these can be approximated as [22]

$$\begin{aligned}
\mu_{i+1}^x &\approx f(\mu_i^x,u_i) + B_d\mu^d(\mu_i^x,u_i)\ , \\
\Sigma_{i+1}^x &\approx B_d\Sigma^d(\mu_i^x,u_i)B_d^T \\
&\quad + \nabla_x\tilde{f}(\mu_i^x,u_i)\Sigma_i^x\left(\nabla_x\tilde{f}(\mu_i^x,u_i)\right)^T
\end{aligned}$$

with $\tilde{f}(\mu_i^x,u_i) = f(\mu_i^x,u_i) + B_d\mu^d(\mu_i^x,u_i)$.

References

[1] M. Buehler, K. Iagnemma, and S. Singh, *The DARPA urban challenge: Autonomous vehicles in city traffic*. Springer, 2009, vol. 56.

[2] K. Kritayakirana and C. Gerdes, "Using the centre of percussion to design a steering controller for an autonomous race car," *Vehicle System Dynamics*, vol. 15, pp. 33–51, 2012.

[3] M. Guiggiani, *The Science of Vehicle Dynamics: Handling, Braking, and Ride of Road and Race Cars*. Springer, 2014.

[4] J. Z. Kolter, C. Plagemann, D. T. Jackson, A. Y. Ng, and S. Thrun, "A probabilistic approach to mixed open-loop and closed-loop control, with application to extreme autonomous driving," in *Int. Conf. Robotics and Automation*, 2010, pp. 839–845.

[5] A. K. Akametalu, J. F. Fisac, J. H. Gillula, S. Kaynama, M. N. Zeilinger, and C. J. Tomlin, "Reachability-based safe learning with Gaussian processes," in *53rd Conf. Decis. Control*, 2014, pp. 1424–1431.

[6] N. R. Kapania and J. C. Gerdes, "Path tracking of highly dynamic autonomous vehicle trajectories via iterative learning control," *American Control Conf.*, pp. 2753–2758, 2015.

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

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

[9] A. Liniger, X. Zhang, P. Aeschbach, A. Georghiou, and J. Lygeros, "Racing miniature cars: Enhancing performance using stochastic MPC and disturbance feedback," in *American Control Conf.*, 2017, pp. 5642–5647.

[10] B. V. Niekerk, A. Damianou, and B. Rosman, "Online constrained model-based reinforcement learning," *Conf. Uncertainty in Artificial Intelligence*, 2017.

[11] U. Rosolia and F. Borrelli, "Learning Model Predictive Control for Iterative Tasks. A Data-Driven Control Framework." *IEEE Trans. Automat. Contr.*, vol. PP, no. 99, p. 1, 2017.

[12] U. Rosolia, A. Carvalho, and F. Borrelli, "Autonomous racing using learning model predictive control," in *American Control Conf.*, 2017, pp. 5115–5120.

[13] M. Lázaro-Gredilla, J. Quiñonero-Candela, C. E. Rasmussen, and A. R. Figueiras-Vidal, "Sparse Spectrum Gaussian Process Regression," *J. Mach. Learning Res.*, vol. 11, pp. 1865–1881, 2010.

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

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

[16] J. Quiñonero-Candela, C. E. Rasmussen, and C. K. Williams, "Approximation methods for Gaussian process regression," *Large-Scale Kernel Machines*, pp. 203–224, 2007.

[17] E. Snelson and Z. Ghahramani, "Sparse Gaussian processes using pseudo-inputs," *Adv. Neural Information Process. Syst.*, pp. 1257–1264, 2006.

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

[19] H. B. Pacejka and E. Bakker, "The magic formula tyre model," *Vehicle System Dynamics*, vol. 21, no. sup1, pp. 1–18, 1992.

[20] T. Faulwasser, B. Kern, and R. Findeisen, "Model predictive path-following for constrained nonlinear systems," in *48th Conf. Decis. Control*, 2009, pp. 8642–8647.

[21] D. Lam, C. Manzie, and M. Good, "Model predictive contouring control," in *49th Conf. Decis. Control*, 2010, pp. 6137–6142.

[22] J. Quiñonero-Candela, A. Girard, and C. E. Rasmussen, "Prediction at an uncertain input for Gaussian processes and relevance vector machines application to multiple-step ahead time-series forecasting," Danish Tech. Univ., Technical Report IMM-2003-18, Oct. 2003.

[23] M. Deisenroth, "Efficient reinforcement learning using Gaussian processes," Ph.D. dissertation, KIT, Karlsruhe, Nov. 2010.

[24] L. Hewing and M. N. Zeilinger, "Cautious model predictive control using Gaussian process regression," *arXiv:1705.10702*, 2017.

[25] C. J. Ostafew, A. P. Schoellig, and T. D. Barfoot, "Robust Constrained Learning-based NMPC enabling reliable mobile robot path tracking," *Int. J. Robotics Research*, vol. 35, no. 13, pp. 1547 – 1563, 2016.

[26] M. Seeger, "Low rank updates for the Cholesky decomposition," EPFL, Technical Report: EPFL-REPORT-161468, 2004.

[27] D. Nguyen-Tuong, M. Seeger, and J. Peters, "Model learning with local Gaussian process regression," *Adv. Robotics*, vol. 23, no. 15, pp. 2015–2034, 2009.

[28] E. C. Kerrigan and J. M. Maciejowski, "Soft constraints and exact penalty functions in model predictive control," *UKACC Int. Conf. Control*, 2000.

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