Scenario-based Optimal Control for Gaussian Process State Space Models

Jonas Umlauft, Thomas Beckers, Sandra Hirche

***Abstract*— Data-driven approaches from machine learning provide powerful tools to identify dynamical systems with limited prior knowledge of the model structure. More particular, the Gaussian process state space model, a Bayesian nonparametric approach, is increasingly utilized in control. Its probabilistic nature is interpreted differently in the control literature, but so far, it is not considered as a distribution over dynamical system which allows a scenario-based control design. This paper introduces how scenarios are sampled from a Gaussian process and utilizes them in a differential dynamic programming approach to solve an optimal control problem. For the linear-quadratic case, we derive probabilistic performance guarantees using results from robust convex optimization. The proposed methods are evaluated numerically for the nonlinear and linear case.**

I. INTRODUCTION

The identification of dynamical systems using data-driven approaches gains attention, as control engineering is increasingly applied in areas where the analytic derivation of model candidates is not possible, e.g. social networks or flexible manipulators in robotics. Generally, the increased availability and improved processing speed of large datasets supports the trend away from parametric towards data-driven, nonparametric models. More specifically, Gaussian process state space models (GP-SSMs) gain attention in system identification [1] due to many favorable properties, such as the high flexibility, the intrinsic bias-variance trade-off with its Bayesian mathematical foundation and the fact that it provides a measure of the model fidelity along with the inferred output [2].

Accordingly, there are numerous applications of Gaussian processes (GPs) in control: A stabilizing control design for GP-SSMs is proposed in [3] and [4]. The authors of [5], [6] and [7] exploit the Gaussian process in robotic applications for stiffness adaptation, cooperation and computed torque control, respectively. However, the uncertainty in the model is interpreted differently in various approaches and we will provide a small overview in Sec. III-B.

One prominent technique to deal with the uncertainty is stochastic optimal control as it allows precise design of performance criteria and is solved efficiently by means of differential dynamic programming (DDP) [8]. Various approximate solutions have been proposed e.g. data-driven approximation of the value function [9], the iterative linear quadratic Gaussian regulator (iLQG) [10] or the scenario-based model predictive control (MPC) [11]. The latter avoids excessive conservatism and provides probabilistic performance guarantees. To make this work for GP-SSMs, a *scenario interpretation* is required which has not been exploited in control.

Therefore, this paper proposes a scenario-based control approach for GP-SSMs, where dynamical models are sampled from a GP to achieve robust control design. Here, probabilistic robustness as discussed in [12] is considered which uses a finite subset of all possible dynamics to optimize the control law. The novelty is to use a nonparametric model as distribution over functions from which realizations are sampled as scenarios. Exploiting principles of differential dynamic programming, a solution for the optimal control problem is approximated. For the linear quadratic (LQ) case, we utilize results from robust convex optimization to derive the required number of samples for given level of probabilistic robustness.

The remainder of this paper is structured as follows: After defining the problem setting in Sec. II, Sec. III reviews GP-SSMs and its different interpretations. Section IV focuses on the scenario-based control design, followed by considerations for the LQ case in Sec. V and simulations in Sec. VI.

II. PROBLEM FORMULATION

Consider a nonlinear discrete-time system[1]

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}\left(\boldsymbol{x}_k, \boldsymbol{u}_k\right), \tag{1}$$

with state $\boldsymbol{x} \in \mathbb{X} \subseteq \mathbb{R}^n$, input $\boldsymbol{u} \in \mathbb{U} \subseteq \mathbb{R}^r$ and $n, r \in \mathbb{N}, k \in \mathbb{N}^0$. The function $\boldsymbol{f} : \mathbb{X} \times \mathbb{U} \to \mathbb{X}$ is infinitely differentiable but unknown. The full state is measurable along with a noisy version of the consecutive state, forming the dataset states are given

$$\mathcal{D} = \left\{\boldsymbol{\xi}^{(i)}, \boldsymbol{y}^{(i)}\right\}_{i=1}^{N},$$

where $\boldsymbol{\xi} \in \mathcal{X} \subseteq \mathbb{R}^p,\ p = n + r,\ \boldsymbol{y} \in \mathbb{R}^n$,

$$\boldsymbol{\xi}^{(i)} = \begin{bmatrix} \boldsymbol{x}_k^{(i)} \\ \boldsymbol{u}_k^{(i)} \end{bmatrix} \in \mathcal{X} \quad \text{and} \quad \boldsymbol{y}^{(i)} = \boldsymbol{f}\left(\boldsymbol{x}_k^{(i)}, \boldsymbol{u}_k^{(i)}\right) + \boldsymbol{\epsilon}$$

with $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma_n^2 \boldsymbol{I}_n)$. Given a stage cost $c : \mathbb{X} \times \mathbb{U} \to \mathbb{R}$ and a final cost $c_f : \mathbb{X} \to \mathbb{R}$, with $c, c_f \in \mathcal{C}^2$,

All authors are members of the Chair of Information-oriented Control, Department of Electrical and Computer Engineering, Technical University of Munich, D-80333 Munich, Germany [jonas.umlauft, t.beckers, hirche]@tum.de

[1]**Notation:** Lower/upper case bold symbols denote vectors/matrices, $\mathbb{R}_+^0$, $\mathbb{R}_+$ all real positive numbers with and without the zero, $\mathbb{N}^0$, $\mathbb{N}$ all natural numbers with and without the zero, and $\mathbb{E}[\cdot]$, $\mathbb{V}[\cdot]$ the expected value and variance of a random variable, respectively. $\boldsymbol{I}_n$ denotes the $n \times n$ identity matrix, $\boldsymbol{A} \succ 0$ positive definiteness of matrix $\boldsymbol{A}$, $\mathcal{N}(\mu, \sigma^2)$ a normal distribution, $\boldsymbol{a}_{(1:n)}$ the first n elements of $\boldsymbol{a}$, $\mathcal{C}^i$ the set of i-times differentiable functions and $\mathrm{diag}(\cdot)$ constructs from a set of scalars/matrices a (block-)diagonal matrix.

the goal is to find a series of control inputs $\boldsymbol{u}_{0:H\text{-}1} = \begin{bmatrix}\boldsymbol{u}_0^\intercal & \cdots & \boldsymbol{u}_{H-1}^\intercal\end{bmatrix}^\intercal \in \mathbb{U}^H \subseteq \mathbb{R}^{rH}$, which minimizes the accumulated cost over the horizon $H \in \mathbb{N}$

$$\min_{\boldsymbol{u}_{0:H\text{-}1}} \sum_{k=0}^{H-1} c(\boldsymbol{x}_k, \boldsymbol{u}_k) + c_f(\boldsymbol{x}_H). \tag{2}$$

We aim to solve this optimal control problem by means of DDP which requires a model of the unknown dynamics. We utilize the Gaussian process framework for the dynamic model as described in the following.

III. SCENARIO SAMPLING FOR GAUSSIAN PROCESSES

This sections first reviews GP-SSMs before discussing its different interpretations in control. Then, we introduce how dynamic models are sampled from a GP to obtain different scenarios for the control design.

A. Gaussian process state space models

The Gaussian process is a stochastic process, which assigns to any finite subset $\{\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_M\} \subset \mathcal{X}$ from a continuous input space $\mathcal{X} \subseteq \mathbb{R}^p$ a joint Gaussian distribution. A common interpretation of the Gaussian process is the "distribution over functions" [2] and written by

$$f_{\boldsymbol{\psi}}(\boldsymbol{\xi}) \sim \mathcal{GP}\left(m_{\boldsymbol{\psi}}(\boldsymbol{\xi}), k_{\boldsymbol{\psi}}(\boldsymbol{\xi}, \boldsymbol{\xi}')\right).$$

It is fully characterized by a mean $m_{\boldsymbol{\psi}}(\boldsymbol{\xi})\colon \mathcal{X} \to \mathbb{R}$ and a covariance $k_{\boldsymbol{\psi}}(\boldsymbol{\xi}, \boldsymbol{\xi}')\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ function. The subscript $\boldsymbol{\psi}$ indicates the dependency of the functions on hyperparameters. Common practice is to set the mean function $m_{\boldsymbol{\psi}}$ to zero due to lack of prior knowledge. The covariance function characterizes the functions over which the GP describes the distribution. Thus, all functions drawn from a GP follow these properties induced by the kernel. For a linear kernel

$$k_{\boldsymbol{\psi}}^{\text{lin}}(\boldsymbol{\xi}, \boldsymbol{\xi}') = \sum_{i=1}^{p} \frac{\xi_i \xi_i'}{l_i^2}, \tag{3}$$

where $\boldsymbol{\psi} = \begin{bmatrix} l_1 & \cdots & l_p \end{bmatrix}^\intercal \in \mathbb{R}_+^p$ are the hyperparameters, all drawn functions are a sum of linear functions, thus linear. For the squared exponential (SE) kernel

$$k_{\boldsymbol{\psi}}^{\text{SE}}(\boldsymbol{\xi}, \boldsymbol{\xi}') = \sigma_f^2 \exp\left(\sum_{i=1}^{p} \frac{(\xi_i - \xi_i')^2}{-2l_i^2}\right), \tag{4}$$

all resulting functions are a sum of Gaussians and the hyperparameters are $\boldsymbol{\psi} = \begin{bmatrix} l_1 & \cdots & l_p & \sigma_f \end{bmatrix}^\intercal \in \mathbb{R}_+^{p+1}$. The SE kernel is *universal*, thus every continuous function can be approximated arbitrarily exact with the GP. The resulting model is considered nonparametric, because the data points itself are the parameters (hyperparameters of the kernel only assume a specific correlation among these points).

To model functions with multidimensional outputs, such as the dynamics in (1), n independent GPs are concatenated

$$\boldsymbol{f}_{\boldsymbol{\Psi}}(\boldsymbol{\xi}) = \begin{cases} f_{\boldsymbol{\psi}_1}(\boldsymbol{\xi}) \sim \mathcal{GP}(0, k_{\boldsymbol{\psi}_1}(\boldsymbol{\xi}, \boldsymbol{\xi}')) \\ \quad\vdots \qquad\qquad \vdots \\ f_{\boldsymbol{\psi}_n}(\boldsymbol{\xi}) \sim \mathcal{GP}(0, k_{\boldsymbol{\psi}_n}(\boldsymbol{\xi}, \boldsymbol{\xi}')), \end{cases} \tag{5}$$

where the prior mean functions $m_{\boldsymbol{\psi}_i}(\boldsymbol{\xi})$ are set to zero for simplicity and $\boldsymbol{\Psi} = \begin{bmatrix}\boldsymbol{\psi}_1^\intercal & \cdots & \boldsymbol{\psi}_n^\intercal\end{bmatrix}^\intercal$ is the concatenation of all parameter vectors. It is denoted as

$$\boldsymbol{f}_{\boldsymbol{\Psi}}(\boldsymbol{\xi}) \sim \mathcal{GP}\left(\boldsymbol{0}, \boldsymbol{k}_{\boldsymbol{\Psi}}(\boldsymbol{\xi}, \boldsymbol{\xi}')\right), \tag{6}$$

where $\boldsymbol{k}_{\boldsymbol{\Psi}}(\cdot,\cdot) = \begin{bmatrix} k_{\boldsymbol{\psi}_1}(\cdot,\cdot) & \cdots & k_{\boldsymbol{\psi}_n}(\cdot,\cdot)\end{bmatrix}^\intercal$ concatenates the kernel functions.

Gaussian processes are often employed for regression: For the unknown function $\tilde{\boldsymbol{f}}: \mathcal{X} \to \mathbb{R}^n$, noisy measurements

$$\boldsymbol{y}^{(i)} = \tilde{\boldsymbol{f}}\left(\boldsymbol{\xi}^{(i)}\right) + \boldsymbol{\epsilon}, \quad i = 1, \ldots, N \tag{7}$$

with $\boldsymbol{\epsilon} \sim \mathcal{N}\left(0, \sigma_n^2 \boldsymbol{I}_n\right)$ are available. Considering a test input $\boldsymbol{\xi}^*$ and the j-th component ($j = 1, \ldots, n$) of the corresponding predicted output $\boldsymbol{y}^*$, the joint distribution is

$$\begin{bmatrix} y_j^* \\ \boldsymbol{Y}_{(j,:)} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}0\\ \boldsymbol{0}\end{bmatrix}, \begin{bmatrix} k_j^* & \boldsymbol{k}_j^\intercal \\ \boldsymbol{k}_j & \boldsymbol{K}_j + \sigma_n^2 \boldsymbol{I}_N \end{bmatrix}\right), \tag{8}$$

where $\boldsymbol{Y} = \begin{bmatrix}\boldsymbol{y}^{(1)} & \cdots & \boldsymbol{y}^{(N)}\end{bmatrix}$ concatenates the measured outputs, $\boldsymbol{Y}_{(j,:)}$ denotes j-th row of $\boldsymbol{Y}$, $k_j^* = k_{\boldsymbol{\psi}_j}(\boldsymbol{\xi}^*, \boldsymbol{\xi}^*)$ and

$$\boldsymbol{K}_j = \begin{bmatrix} k_{\boldsymbol{\psi}_j}\left(\boldsymbol{\xi}^{(1)}, \boldsymbol{\xi}^{(1)}\right) & \cdots & k_{\boldsymbol{\psi}_j}\left(\boldsymbol{\xi}^{(1)}, \boldsymbol{\xi}^{(N)}\right) \\ \vdots & \ddots & \vdots \\ k_{\boldsymbol{\psi}_j}\left(\boldsymbol{\xi}^{(N)}, \boldsymbol{\xi}^{(1)}\right) & \cdots & k_{\boldsymbol{\psi}_j}\left(\boldsymbol{\xi}^{(N)}, \boldsymbol{\xi}^{(N)}\right) \end{bmatrix} \in \mathbb{R}^{N \times N}$$

$$\boldsymbol{k}_j = \begin{bmatrix} k_{\boldsymbol{\psi}_j}\left(\boldsymbol{\xi}^{(1)}, \boldsymbol{\xi}^*\right) & \cdots & k_{\boldsymbol{\psi}_j}\left(\boldsymbol{\xi}^{(N)}, \boldsymbol{\xi}^*\right)\end{bmatrix}^\intercal \in \mathbb{R}^N.$$

For further notations, we write the concatenation of kernel evaluations as $\boldsymbol{K}_j = k_j(\boldsymbol{\Xi}, \boldsymbol{\Xi})$ and $\boldsymbol{k}_j = \boldsymbol{k}_j(\boldsymbol{\Xi}, \boldsymbol{\xi}^*)$, where $\boldsymbol{\Xi} = \begin{bmatrix}\boldsymbol{\xi}^{(1)} & \cdots & \boldsymbol{\xi}^{(N)}\end{bmatrix} \in \mathbb{R}^{p \times N}$.

Conditioning the joint distribution (8) on the test input $\boldsymbol{\xi}^*$ and the observations $\mathcal{D}$ results in a normal distribution for $p(y_j^* | \boldsymbol{\xi}^*, \mathcal{D})$ with mean and variance given by

$$\mathbb{E}[y_j^* | \boldsymbol{\xi}^*, \mathcal{D}, \boldsymbol{\psi}_j] = \boldsymbol{k}_j^\intercal \left(\boldsymbol{K}_j + \sigma_n^2 \boldsymbol{I}_N\right)^{-1} \boldsymbol{Y}_{(j,:)} =: \mu_j(\boldsymbol{\xi}^*), \tag{9}$$

$$\mathbb{V}[y_j^* | \boldsymbol{\xi}^*, \mathcal{D}, \boldsymbol{\psi}_j] = k_j^* - \boldsymbol{k}_j^\intercal \left(\boldsymbol{K}_j + \sigma_n^2 \boldsymbol{I}_N\right)^{-1} \boldsymbol{k}_j =: \sigma_j^2(\boldsymbol{\xi}^*) \tag{10}$$

and concatenated to

$$\boldsymbol{\mu}(\boldsymbol{\xi}^*) \coloneqq \begin{bmatrix}\mu_1(\boldsymbol{\xi}^*) & \cdots & \mu_n(\boldsymbol{\xi}^*)\end{bmatrix}^\intercal,$$
$$\boldsymbol{\sigma}^2(\boldsymbol{\xi}^*) \coloneqq \begin{bmatrix}\sigma_1^2(\boldsymbol{\xi}^*) & \cdots & \sigma_n^2(\boldsymbol{\xi}^*)\end{bmatrix}^\intercal.$$

From Bayesian inference principle, the hyperparameters are obtained through optimization of the marginal likelihood for each component, i.e. for every $j = 1, \ldots, n$.

B. Interpretations of Gaussian process dynamic models

Gaussian processes are widely applied in system identification and control to model unknown dynamics. However, the interpretations of the stochastic process in analysis and design are different. A visualization is provided in Fig. 1. For notational simplicity, we consider in this section scalar uncontrolled, discrete-time state space models of the form $x_{k+1} = f(x_k)$.

First, in the *deterministic* interpretation, the mean function (9) of the GP is taken as estimate for the true function

$$x_{k+1} = \mu(x_k).$$

The variance function (10) is either ignored or used for additional task, e.g. robotic cooperation [6]. The approaches neglect the present uncertainty and do not make use of the full potential of the probabilistic model.

Alternatively, the work in [13] derives (under certain assumptions on $f(x)$) an upper bound for the error $|f(x) - \mu(x)|$, which is proportional to the variance $\sigma(x)$ and holds with high probability. Thus, the GP model is interpreted as a high probability *robust* model of the form

$$x_{k+1} = \mu(x_k) \pm \beta\sigma(x_k),$$

where $\beta > 0$ is a constant. In control, this has been employed e.g. in [4] for feedback linearization. However, this view requires knowledge regarding the true function, e.g. the correct kernel, which is hard to guarantee.

Third, the uncertainty in the prediction of the GP is interpreted as process noise and results in drawing from the normal distribution predicted by the GP in each step, thus

$$x_{k+1} \sim \mathcal{N}\left(\mu(x_k), \sigma^2(x_k)\right). \tag{11}$$

It is utilized for control design with corresponding stability considerations in [3] and [14]. However, assuming deterministic dynamics, the interpretation is inconsistent, since the model predicts two different outputs for two queries at the same state x_k.

Finally, the *belief space* view is a hybrid concept of the before mentioned techniques. An approximation $\boldsymbol{f}_{\text{BS}}$ is utilized to map from the current state μ_{x_k} with covariance σ_{x_k} to the next state distribution given by

$$\begin{bmatrix}\mu_{x_{k+1}} \\ \sigma_{x_{k+1}}\end{bmatrix} = \boldsymbol{f}_{\text{BS}}\left(\begin{bmatrix}\mu_{x_k} \\ \sigma_{x_k}\end{bmatrix}\right).$$

It is applied to control design in [15], but the applied approximation refrains the approach from providing performance or convergence guarantees.

The summary shows that none of these methods uses an interpretation which considers the GP as a distribution over functions which allows to draw deterministic functions as samples of the stochastic process. We will refer to this interpretation as the *scenario* view and explain it in detail in the following section.

C. Iterative sampling from Gaussian processes

As stated in Sec. II, the control goal is to minimize the cost function over a time horizon H. This requires to perform model-based predictions of the state $\boldsymbol{x}_k$, $k = 1, \ldots, H$, starting from an initial state $\boldsymbol{x}_0$ for a given input sequence $\boldsymbol{u}_{0:H\text{-}1}$. This section proposes the scenario view of GPs to perform these predictions which differs from previously used techniques. So other than in the deterministic interpretation, we sample from the GP. But we do not sample in the output space (as in the stochastic interpretation) but from the function space. Technically, this requires sampling from an infinite dimensional space, however, we are not interested in the entire function, but only at the locations which are reached during the prediction horizon. So for the first draw, we start similar to the stochastic interpretation

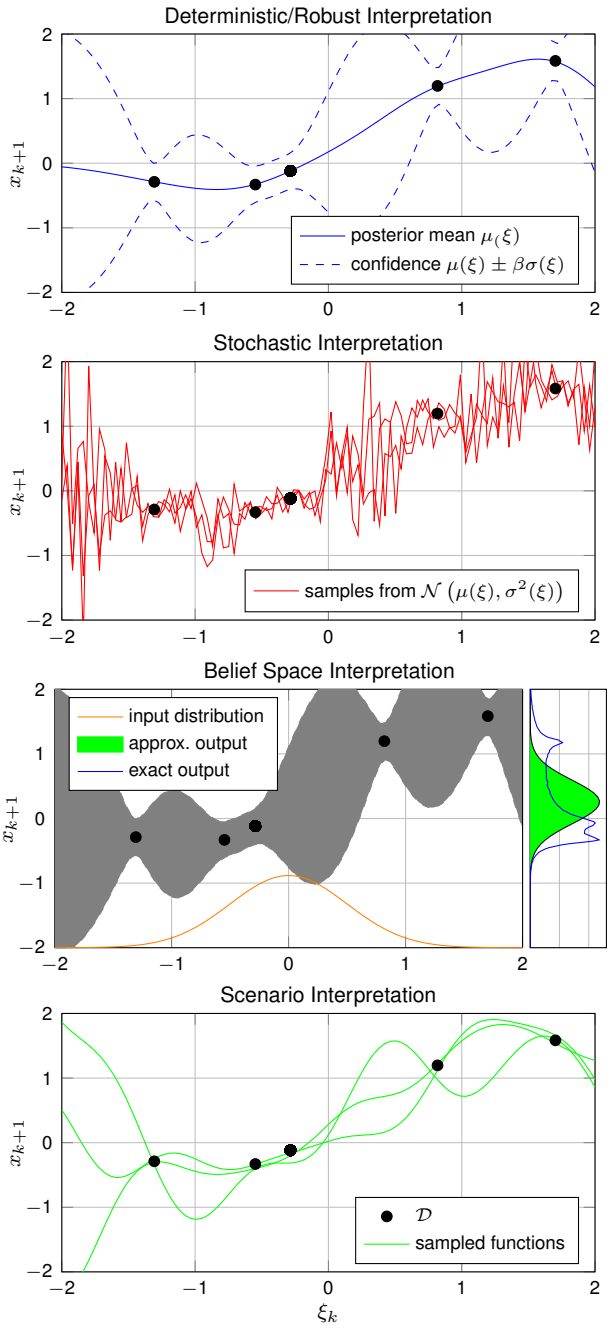

Fig. 1. Different interpretations of GP-SSMs in control.

$\boldsymbol{x}_1 \sim \mathcal{N}\left(\boldsymbol{\mu}(\boldsymbol{\xi}_0), \operatorname{diag}\left(\boldsymbol{\sigma}^2(\boldsymbol{\xi}_0)\right)\right)$ and draw from the distribution $p(\boldsymbol{x}_1|\boldsymbol{\xi}_0, \mathcal{D}, \boldsymbol{\Psi})$, where $\boldsymbol{\xi}_0 = \begin{bmatrix}\boldsymbol{x}_0^\mathsf{T} & \boldsymbol{u}_0^\mathsf{T}\end{bmatrix}^\mathsf{T}$. The realization of $\boldsymbol{x}_1$ is considered as a realization of the function. Thus, for any further predictions, in order to obtain $\boldsymbol{x}_2$, we condition on this observation of the function. The resulting posterior distribution for $\boldsymbol{x}_2$ is

$$p(\boldsymbol{x}_2|\boldsymbol{\xi}_1, \boldsymbol{x}_1, \boldsymbol{\xi}_0, \mathcal{D}, \boldsymbol{\Psi}).$$

The sampled point $(\boldsymbol{\xi}_0, \boldsymbol{x}_1)$ is treated similarly to a training point of the GP, with the important difference, that it is observation noise free: No σ_n^2 is added to the kernel evalu-

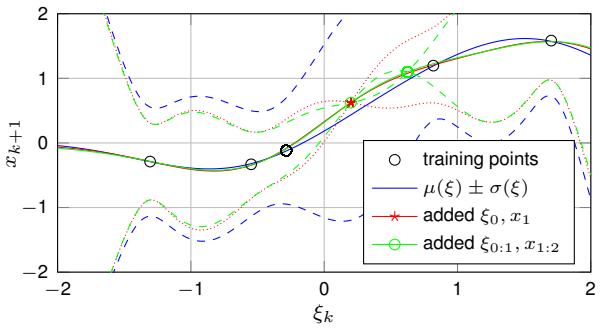

Fig. 2. Illustrative example for forward simulation with a sampled GP. For the points in the prediction horizon the variance is zero, while the training points are subject to observations noise.

ation $k_j(\boldsymbol{\xi}_0, \boldsymbol{\xi}_0)$. This ensures, that every further evaluation of the Gaussian process at the same input location $\boldsymbol{\xi}_0$ will result in exactly the same output, since the variance (10) is zero and $\boldsymbol{\mu}(\boldsymbol{\xi}_0) = \boldsymbol{x}_1$. A visualization for two steps is shown in Fig. 2. In step k, the joint distribution is given by

$$\begin{bmatrix} x_{k+1,j} \\ \boldsymbol{x}_{1:k,j} \\ \boldsymbol{Y}_{(j,:)} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} k_j(\boldsymbol{\xi}_k, \boldsymbol{\xi}_k) & \boldsymbol{k}_j(\boldsymbol{\xi}_k, \tilde{\boldsymbol{\Xi}}) & \boldsymbol{k}_j(\boldsymbol{\xi}_k, \boldsymbol{\Xi}) \\ \boldsymbol{k}_j(\tilde{\boldsymbol{\Xi}}, \boldsymbol{\xi}_k) & \boldsymbol{k}_j(\tilde{\boldsymbol{\Xi}}, \tilde{\boldsymbol{\Xi}}) & \boldsymbol{k}_j(\tilde{\boldsymbol{\Xi}}, \boldsymbol{\Xi}) \\ \boldsymbol{k}_j(\boldsymbol{\Xi}, \boldsymbol{\xi}_k) & \boldsymbol{k}_j(\boldsymbol{\Xi}, \tilde{\boldsymbol{\Xi}}) & \boldsymbol{K}_j + \sigma_n^2 \boldsymbol{I}_N \end{bmatrix} \right)$$

where $\boldsymbol{x}_{1:k,j} = \begin{bmatrix} x_{1,j} & \cdots & x_{k,j} \end{bmatrix}^\mathsf{T} \in \mathbb{R}^k$ concatenates the j-th dimension and $\tilde{\boldsymbol{\Xi}} = \begin{bmatrix} \boldsymbol{\xi}_0 & \cdots & \boldsymbol{\xi}_{k-1} \end{bmatrix} \in \mathbb{R}^{p \times k}$. Based on this notation, the prediction procedure for a horizon H is summarized in Algorithm 1.

Algorithm 1: Sampling procedure for GP-SSMs

Input : $\mathcal{D}$, $\boldsymbol{\Psi}$, $\boldsymbol{x}_0$, H, $\boldsymbol{u}_{0:H\text{-}1}$
Output: $\boldsymbol{x}_{1:H}$
$k = 0$, $\tilde{\boldsymbol{Y}} = [\,]$, $\tilde{\boldsymbol{\Xi}} = [\,]$;
while $k < H$ **do**
  $\boldsymbol{\xi}_k = [\boldsymbol{x}_k^\mathsf{T} \ \boldsymbol{u}_k^\mathsf{T}]^\mathsf{T}$;
  Draw from $p(\boldsymbol{x}_{k+1} | \boldsymbol{\xi}_k, \tilde{\boldsymbol{\Xi}}, \tilde{\boldsymbol{Y}}, \mathcal{D}, \boldsymbol{\Psi})$;
  $\tilde{\boldsymbol{\Xi}} = [\tilde{\boldsymbol{\Xi}} \ \boldsymbol{\xi}_k]$, $\tilde{\boldsymbol{Y}} = [\tilde{\boldsymbol{Y}} \ \boldsymbol{x}_{k+1}]$;
  $k = k + 1$;
end

D. From functions samples to scenarios

The previous section explained how a trajectory of one dynamical model is consistently sampled. To obtain a robust design, we draw multiple realizations of the GP to account for the model uncertainty. The proposed GP forward simulation in Algorithm 1 is therefore employed M times to generate M different scenarios. The initial state is the same for all, however for each scenario a different function is sampled from the GP resulting in different predicted trajectories. The M different dynamic models are all itself deterministic and denoted by $\boldsymbol{f}^{[m]}$ with state $\boldsymbol{x}^{[m]}$, $m = 1, \ldots, M$ and

$$\boldsymbol{x}_{k+1}^{[m]} = \boldsymbol{f}^{[m]}\left(\boldsymbol{x}_k^{[m]}, \boldsymbol{u}_k\right).$$

Given a nominal control sequence $\boldsymbol{u}_{0:H\text{-}1}$, nominal trajectories for each scenario $\boldsymbol{x}_{0:H}^{[m]} = \left\{\boldsymbol{x}_0^{[m]}, \ldots, \boldsymbol{x}_H^{[m]}\right\}$ can be computed. For notational simplicity, we introduce the concatenated state vector/transition function

$$\boldsymbol{x}_k^{[1:M]} = \begin{bmatrix} \boldsymbol{x}_k^{[1]} \\ \vdots \\ \boldsymbol{x}_k^{[M]} \end{bmatrix}, \boldsymbol{f}^{[1:M]}\left(\boldsymbol{x}^{[1:M]}, \boldsymbol{u}\right) = \begin{bmatrix} \boldsymbol{f}^{[1]}\left(\boldsymbol{x}^{[1]}, \boldsymbol{u}\right) \\ \vdots \\ \boldsymbol{f}^{[M]}\left(\boldsymbol{x}^{[M]}, \boldsymbol{u}\right) \end{bmatrix}.$$

The control input $\boldsymbol{u}$ is the same for all scenarios, since only a single control input is applied to the real system.

IV. ITERATIVE LQR FOR SCENARIOS

We make use of the scenario-based forward simulations of GP-SSMs in an optimal control setting. We introduce the notation for the cost function over all scenarios $c\left(\boldsymbol{x}^{[1:M]}, \boldsymbol{u}\right) = \sum_{m=1}^{M} c\left(\boldsymbol{x}^{[m]}, \boldsymbol{u}\right)$, $c_f\left(\boldsymbol{x}^{[1:M]}\right) = \sum_{m=1}^{M} c_f\left(\boldsymbol{x}^{[m]}\right)$ and define the value function $V_k : \mathbb{X}^M \to \mathbb{R}$

$$V_k\left(\boldsymbol{x}^{[1:M]}\right) = \min_{\boldsymbol{u}_{0:H\text{-}1}} \sum_{i=k}^{H-1} c\left(\boldsymbol{x}_i^{[1:M]}, \boldsymbol{u}_i\right) + c_f\left(\boldsymbol{x}_H^{[1:M]}\right),$$

which is the minimal cost-to-go from any state $\boldsymbol{x}^{[1:M]} \in \mathbb{X}^M$. Following Bellman's principle of optimality the value function is written recursively as

$$V_k\left(\boldsymbol{x}^{[1:M]}\right) = \min_{\boldsymbol{u}} c\left(\boldsymbol{x}^{[1:M]}, \boldsymbol{u}\right) + V_{k+1}\left(\boldsymbol{f}^{[1:M]}(\boldsymbol{x}^{[1:M]}, \boldsymbol{u})\right)$$

with the boundary condition $V_H\left(\boldsymbol{x}^{[1:M]}\right) = c_f\left(\boldsymbol{x}^{[1:M]}\right)$. Therefore, the value function can only be solved backwards in time along a nominal trajectory (*backward pass*). Given the value function at time step $k+1$, the optimal control input is computed for time k, leading to the value function at time k. An updated control input results in a new nominal trajectory which requires a new simulation forward in time (*forward pass*) [16]. Alternating forward and backward pass leads to convergence to a (locally) optimal trajectory [17].

a) Backward Pass: We define the cost-to-go function $C_k : \mathbb{X}^M \times \mathbb{U} \to \mathbb{R}$ in terms of deviations $\boldsymbol{\delta}_{\boldsymbol{x}}^{[1:M]} \in \mathbb{R}^{nM}$, $\boldsymbol{\delta}_{\boldsymbol{u}} \in \mathbb{R}^r$ from the nominal trajectories $\boldsymbol{x}_{0:H}^{[1:M]}$, $\boldsymbol{u}_{0:H\text{-}1}$

$$\begin{aligned} C_k\left(\boldsymbol{\delta}_{\boldsymbol{x}}^{[1:M]}, \boldsymbol{\delta}_{\boldsymbol{u}}\right) &= c\left(\boldsymbol{\delta}_{\boldsymbol{x}}^{[1:M]} + \boldsymbol{x}_k^{[1:M]}, \boldsymbol{u}_k + \boldsymbol{\delta}_{\boldsymbol{u}}\right) \\ &\quad + V_{k+1}\left(\boldsymbol{f}^{[1:M]}(\boldsymbol{\delta}_{\boldsymbol{x}}^{[1:M]} + \boldsymbol{x}_k^{[1:M]}, \boldsymbol{u}_k + \boldsymbol{\delta}_{\boldsymbol{u}})\right). \end{aligned}$$

This cost function is now approximated for all $k = 1, \ldots, H-1$ by a linear model around the nominal trajectory with dynamic matrix $\boldsymbol{A}_k^{[1:M]} \in \mathbb{R}^{Mn \times Mn}$ and input matrix $\boldsymbol{B}_k^{[1:M]} \in \mathbb{R}^{Mn \times r}$

$$\boldsymbol{A}_k^{[1:M]} = \frac{\partial \boldsymbol{f}^{[1:M]}}{\partial \boldsymbol{x}^{[1:M]}}, \quad \boldsymbol{B}_k^{[1:M]} = \frac{\partial \boldsymbol{f}^{[1:M]}}{\partial \boldsymbol{u}}. \tag{12}$$

Approximating the cost function as quadratic, allows the following approximation of the cost-to-go function

$$\begin{aligned} \tilde{C}_k\left(\boldsymbol{\delta}_{\boldsymbol{x}}^{[1:M]}, \boldsymbol{\delta}_{\boldsymbol{u}}\right) &= \boldsymbol{\delta}_{\boldsymbol{x}}^{[1:M]\mathsf{T}} \boldsymbol{Q}_k^{[1:M]} \boldsymbol{\delta}_{\boldsymbol{x}}^{[1:M]} + \boldsymbol{\delta}_{\boldsymbol{u}}^\mathsf{T} \boldsymbol{R}_k \boldsymbol{\delta}_{\boldsymbol{u}} \\ &\quad + \boldsymbol{q}_k^{[1:M]} \boldsymbol{\delta}_{\boldsymbol{x}}^{[1:M]} + \boldsymbol{r}_k \boldsymbol{\delta}_{\boldsymbol{u}} + \boldsymbol{\delta}_{\boldsymbol{u}}^\mathsf{T} \boldsymbol{S}_k^{[1:M]} \boldsymbol{\delta}_{\boldsymbol{x}}^{[1:M]} \\ &\quad + q_k^{[1:M]} + V_{k+1}\left(\boldsymbol{x}_{k+1}^{[1:M]}\right), \end{aligned} \tag{13}$$

where

$$\begin{aligned}
\boldsymbol{Q}_k^{[1:M]} &= \frac{\partial^2 c}{\partial \boldsymbol{x}\partial \boldsymbol{x}} + \boldsymbol{A}_k^{[1:M]T}\boldsymbol{W}_{k+1}^{[1:M]}\boldsymbol{A}_k^{[1:M]},\\
\boldsymbol{R}_k &= \frac{\partial^2 c}{\partial \boldsymbol{u}\partial \boldsymbol{u}} + \boldsymbol{B}_k^{[1:M]T}\boldsymbol{W}_{k+1}^{[1:M]}\boldsymbol{B}_k^{[1:M]},\\
\boldsymbol{q}_k^{[1:M]} &= \frac{\partial c}{\partial \boldsymbol{x}} + \boldsymbol{w}_{k+1}^{[1:M]}\boldsymbol{A}_k^{[1:M]},\\
\boldsymbol{r}_k &= \frac{\partial c}{\partial \boldsymbol{u}} + \boldsymbol{w}_{k+1}^{[1:M]}\boldsymbol{B}_k^{[1:M]},\\
\boldsymbol{S}_k^{[1:M]} &= \frac{\partial c}{\partial \boldsymbol{x}\partial \boldsymbol{u}} + \boldsymbol{A}_k^{[1:M]\intercal}\boldsymbol{W}_{k+1}^{[1:M]}\boldsymbol{B}_k^{[1:M]},
\end{aligned} \tag{14}$$

and $q_k^{[1:M]} = c\left(\boldsymbol{x}_k^{[1:M]}, \boldsymbol{u}_k\right)$, where the gradient and Hessian of the next step value function V_{k+1}, denoted by $\boldsymbol{W}_{k+1}^{[1:M]} \in \mathbb{R}^{Mn\times Mn}$ and $\boldsymbol{w}_{k+1}^{[1:M]} \in \mathbb{R}^{Mn}$ are used. These are obtained iteratively backwards in time, starting with

$$\boldsymbol{W}_H^{[1:M]} = \frac{\partial^2 c_f}{\partial \boldsymbol{x}\partial \boldsymbol{x}}, \quad \boldsymbol{w}_H^{[1:M]} = \frac{\partial c_f}{\partial \boldsymbol{x}}$$

For the recursive solution of the value function, the approximate cost-to-go (13) is minimized over the deviation $\boldsymbol{\delta_u}$

$$V_k\left(\boldsymbol{x}^{[1:M]}\right) = \min_{\boldsymbol{\delta_u}} C_k\left(\boldsymbol{\delta}_{\boldsymbol{x}}^{[1:M]}, \boldsymbol{\delta_u}\right),$$

which is obtained for

$$\boldsymbol{\delta_u} = \boldsymbol{L}_k^{[1:M]}\boldsymbol{\delta}_{\boldsymbol{x}}^{[1:M]} + \boldsymbol{l}_k, \tag{15}$$

where $\boldsymbol{L}_k^{[1:M]} = -\boldsymbol{R}_k{}^{-1}\boldsymbol{S}_k^{[1:M]} \in \mathbb{R}^{r\times Mn}$, $\boldsymbol{l}_k = -\boldsymbol{R}_k{}^{-1}\boldsymbol{r}_k \in \mathbb{R}^r$. Substituting the optimal control (15) in the approximate cost-to-go function (13) yields the quadratic value function

$$\begin{aligned}
V_k\left(\boldsymbol{x}^{[1:M]}\right) &= w_k^{[1:M]} + \boldsymbol{w}_k^{[1:M]}\left(\boldsymbol{x}^{[1:M]} - \boldsymbol{x}_k^{[1:M]}\right)\\
&+ \frac{1}{2}\left(\boldsymbol{x}^{[1:M]} - \boldsymbol{x}_k^{[1:M]}\right)^\intercal \boldsymbol{W}_k^{[1:M]}\left(\boldsymbol{x}^{[1:M]} - \boldsymbol{x}_k^{[1:M]}\right),
\end{aligned}$$

where

$$\begin{aligned}
\boldsymbol{W}_k^{[1:M]} &= \boldsymbol{Q}^{[1:M]} + \boldsymbol{L}_k^{[1:M]\intercal}\boldsymbol{R}_k\boldsymbol{L}_k^{[1:M]}\\
&\quad + \boldsymbol{S}^{[1:M]\intercal}\boldsymbol{L}_k^{[1:M]} + \boldsymbol{L}_k^{[1:M]\intercal}\boldsymbol{S}^{[1:M]}\\
\boldsymbol{w}_k^{[1:M]} &= \boldsymbol{q}^{[1:M]} + \boldsymbol{L}_k^{[1:M]\intercal}\boldsymbol{R}_k\boldsymbol{l}_k + \boldsymbol{S}^{[1:M]\intercal}\boldsymbol{l}_k + \boldsymbol{L}_k^{[1:M]\intercal}\boldsymbol{r}_k\\
w_k^{[1:M]} &= \frac{1}{2}\boldsymbol{l}_k^\intercal\boldsymbol{R}_k\boldsymbol{l}_k + \boldsymbol{r}_k\boldsymbol{l}_k + w_{k+1}^{[1:M]} + q_k^{[1:M]}.
\end{aligned}$$

b) Forward Pass: For the update of the nominal trajectory, we start with $\boldsymbol{x}_0^{[m]} = \boldsymbol{x}_0^{[m,\text{new}]}$ and apply the optimal control (15) derived in the backward pass

$$\begin{aligned}
\boldsymbol{u}_k^{\text{new}} &= \boldsymbol{u}_k + \boldsymbol{L}^{[m]}\left(\boldsymbol{x}_k^{[m,\text{new}]} - \boldsymbol{x}_k^{[m]}\right) + \boldsymbol{l}_k\\
\boldsymbol{x}_{k+1}^{[m,\text{new}]} &= \boldsymbol{f}^{[m]}\left(\boldsymbol{x}_k^{[m,\text{new}]}, \boldsymbol{u}_k^{\text{new}}\right).
\end{aligned}$$

Remark 1: Because the true cost-to-go function is not quadratic, the new trajectory not necessarily improves the overall cost. However, a line search in the direction of improvement is commonly implemented, for more details see [10].

Remark 2: The linear approximation of the dynamic in (12) requires to linearize each of the M GPs along the nominal trajectory. Due to the linearity of the differentiation operation, the gradient of a GP is again a GP, which has no uncertainty at previously visited points. We therefore utilize the gradient of the mean functions.

However, there is no guarantee how well the control law (15) performs on the true system. Generally, providing performance guarantees based on an uncertain model is very difficult. Nevertheless, for linear system this is possible as we show in the following to demonstrate the consistency of our approach.

V. GUARANTEES FOR LINEAR SYSTEMS

This section shows that the presented approach is consistent with scenario-based MPC for linear systems, which allows to derive probabilistic performance guarantees of the control law on the real system. We therefore consider the specific case of an LQ problem in the following.

A. Reformulation for the LQ case

It is now assumed, that the unknown system (1) has a finite reproducing kernel Hilbert space (RKHS) norm [13] under the linear kernel (3), formally written as

$$\|\boldsymbol{f}\|_{k_{\boldsymbol{\psi}}^{\text{lin}}} < \infty, \tag{16}$$

which is equivalent to consider linear systems with unknown dynamic and input matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, $\boldsymbol{B} \in \mathbb{R}^{n\times r}$, respectively. Let $\boldsymbol{Q}, \boldsymbol{Q}_f \in \mathbb{R}^{n\times n}$, $\boldsymbol{R} \in \mathbb{R}^{r\times r}$ define the cost

$$c(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{x}^\intercal\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{u}^\intercal\boldsymbol{R}\boldsymbol{u}, \quad c_f(\boldsymbol{x}) = \boldsymbol{x}^\intercal\boldsymbol{Q}_f\boldsymbol{x} \tag{17}$$

with $\boldsymbol{R}, \boldsymbol{Q}, \boldsymbol{Q}_f \succ 0$. The optimization

$$\min_{\boldsymbol{u}_{0:H\text{-}1}} \sum_{k=0}^{H-1} c(\boldsymbol{x}_k, \boldsymbol{u}_k) + c_f(\boldsymbol{x}_H) \tag{18}$$

is then solved exactly by the proposed dynamic programming scheme, since the employed quadratic form of the cost-to-go holds exactly. Thus only a single forward-backward pass is required since the linearization of the dynamics and the quadratic cost functions hold globally and are not updated with a new nominal trajectory. However, instead of employing DDP, the problem can also be reformulated to

$$\begin{aligned}
\min_{\boldsymbol{u}_{0:H\text{-}1}} &\left(\boldsymbol{G}\boldsymbol{u}_{0:H\text{-}1} + \boldsymbol{H}\boldsymbol{x}_0\right)^\intercal \boldsymbol{Q}_{\text{all}}\left(\boldsymbol{G}\boldsymbol{u}_{0:H\text{-}1} + \boldsymbol{H}\boldsymbol{x}_0\right)\\
&+ \boldsymbol{u}_{0:H\text{-}1}^\intercal\boldsymbol{R}_{\text{all}}\boldsymbol{u}_{0:H\text{-}1}
\end{aligned} \tag{19}$$

where $\boldsymbol{Q}_{\text{all}} = \text{diag}(\boldsymbol{Q}, \ldots, \boldsymbol{Q}, \boldsymbol{Q}_f) \in \mathbb{R}^{n(H+1)\times n(H+1)}$, $\boldsymbol{R}_{\text{all}} = \text{diag}(\boldsymbol{R}, \ldots, \boldsymbol{R}) \in \mathbb{R}^{Hr\times Hr}$ and $\boldsymbol{G} \in \mathbb{R}^{n(H+1)\times rH}$, $\boldsymbol{H} \in \mathbb{R}^{n(H+1)\times n}$ are defined such that the state along the horizon is given by $\boldsymbol{x}_{0:H} = \boldsymbol{G}\boldsymbol{u}_{0:H\text{-}1} + \boldsymbol{H}\boldsymbol{x}_0$. Since matrices $\boldsymbol{A}$, $\boldsymbol{B}$ (constructing $\boldsymbol{G}$ and $\boldsymbol{H}$) are unknown, they are sampled from a GP-SSM with linear kernel.

B. Scenario sampling for the linear kernel

Other than the SE kernel, the linear kernel is not universal and it requires only p noise free samples to fix the GP-SSM to a single model. Thus, after $k = p$ steps of Algorithm 1, the variance of the GP is globally zero. The dynamics for each scenario are then given globally by $\boldsymbol{A}^{[m]}$, $\boldsymbol{B}^{[m]}$ for $m = 1, \ldots, M$. The optimal control input is obtained

by either using DDP (Sec. IV) or solving the epigraphic reformulation

$$h^* = \arg \min_{\boldsymbol{u}_{0:H\text{-}1}, h} h \tag{20}$$
$$\text{s.t.} \left(\boldsymbol{G}^{[m]}\boldsymbol{u}_{0:H\text{-}1} + \boldsymbol{H}^{[m]}\boldsymbol{x}_0\right)^\intercal \boldsymbol{Q}_{\text{all}} \left(\boldsymbol{G}^{[m]}\boldsymbol{u}_{0:H\text{-}1} + \boldsymbol{H}^{[m]}\boldsymbol{x}_0\right) + \boldsymbol{u}_{0:H\text{-}1}^\intercal \boldsymbol{R}_{\text{all}} \boldsymbol{u}_{0:H\text{-}1} < h, \quad m = 1, \dots, M$$

where h is the scalar slack variable. Each of the m convex constraints corresponds to a realization of the random variables $\boldsymbol{A}^{[m]}$, $\boldsymbol{B}^{[m]}$. This setting is known as scenario optimization which is used to relax convex optimization problems where the constraint depends on a continuous-valued random variable [12]. These problems are simplified by considering only a finite number of realizations of the random variable, as shown in (20). Since not all constraints (corresponding to all linear dynamics) are considered, it is of interest if the performance h^* is also achieved for all other possible dynamics:

Theorem 1: For an unknown system (1) under assumption (16) and quadratic cost (17), control law (15) achieves a cost less or equal h^* from (20) with probability no smaller than $1-\beta$, $\beta \in (0,1)$ on all systems $\boldsymbol{f}$ with $\|\boldsymbol{f}\|_{k_{\boldsymbol{\psi}}^{\text{lin}}} < \infty$ but at most an α-fraction, $\alpha \in (0,1)$, if at least

$$M \geq 2\left(Hr - \log\beta\right)/\alpha$$

samples are drawn according to Algorithm 1.

Proof: Since (19) is quadratic in $\boldsymbol{u}_{0:H\text{-}1}$, the optimization (18) is convex for a system with $\|\boldsymbol{f}\|_{k_{\boldsymbol{\psi}}^{\text{lin}}} < \infty$ and quadratic cost function (17). Therefore, [18, Theorem 1] is applicable and the cost of the worst case scenario h^* is with probability $1-\beta$ an upper bound for an α-fraction of all possible dynamics with bounded RKHS norm under the linear kernel to which the control (15) is applied. ■

From an intuitive perspective, the *confidence parameter* β denotes the probability, that a not representative subset of dynamical systems from the set of all dynamics is sampled. This is important from a theoretical point, however, from a practical point, it can be made very small, since the number of samples only increases with $\log\beta$. The *violation parameter* α indicates the fraction of all constraints which are violated by the given solution. Thus in our case, for how many linear systems the computed performance is not achieved. This allows to chose the number of samples, such that a desired fraction of all dynamics is ensured to achieve a specific level of performance.

VI. SIMULATION

A. Setup

To evaluate the approach numerically, consider

$$\boldsymbol{f}(\boldsymbol{x}, u) = \begin{bmatrix} 0.8x_1 - 0.5x_2 + 0.1\cos(3x_1)x_2 \\ 0.4x_1 + 0.5x_2 + (1 + 0.3\sin(2x_2))u \end{bmatrix} \tag{21}$$

and the unstable linear system

$$\boldsymbol{x}_{k+1} = \underbrace{\begin{bmatrix} 0.9 & 0.2 \\ 1 & 0 \end{bmatrix}}_{=:\boldsymbol{A}_{\text{sim}}} \boldsymbol{x}_k + \underbrace{\begin{bmatrix} 1 \\ 0.5 \end{bmatrix}}_{=:\boldsymbol{B}_{\text{sim}}} u_k \tag{22}$$

$\boldsymbol{x}_0$	$\boldsymbol{Q}$	$\boldsymbol{Q}_f$	R_{nl}	R_{lin}	N	H	$\mathbb{U}$	σ_n
$[2\ 2]^\intercal$	$\boldsymbol{I}_2$	$\boldsymbol{I}_2$	10^{-3}	1	125	9	$[-3\ 3]$	10^{-2}

TABLE I
PARAMETERS EMPLOYED IN THE SIMULATION.

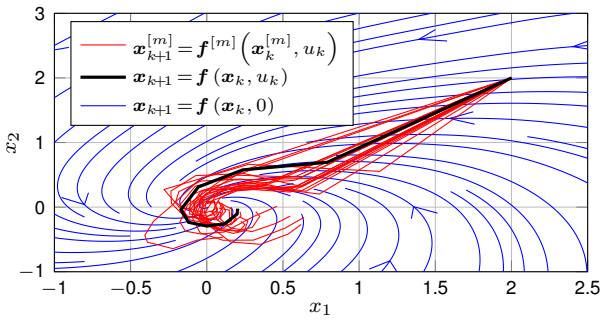

Fig. 3. Simulation nonlinear system: Blue arrows show behavior for zero control input, red lines trajectories for 20 sampled scenarios used in control design and black shows behavior of the real system for the derived controller.

to show an application of Theorem 1. For both, we utilize the quadratic cost functions (14) and the parameters in Table I.

In a first step, $N = 5^3 = 125$ training points are taken on the uniform grid $[-3\ 3] \times [-3\ 3] \times [-3\ 3] \subset \mathbb{X} \times \mathbb{U}$. To obtain the hyperparameters for the SE kernel (4) for the nonlinear system and the linear kernel (3) for the linear system, a conjugate gradient solver from [2] is employed. Based on the resulting GP-SSMs, scenarios (=dynamical models) are sampled starting from the same initial point $\boldsymbol{x}_0$ using Algorithm 1. The iterative LQR approach described in Sec. IV is employed to find optimal control inputs $\boldsymbol{u}_{0:H\text{-}1}$ and the corresponding trajectories for each scenario. For the nonlinear (nl) case, $M_{\text{nl}} = 20$ scenarios are used. For the linear (lin) case, a confidence parameter $\beta = 10^{-2}$ and a violation parameter $\alpha = 0.1$ is chosen, which results in $M_{\text{lin}} \geq 273$ scenarios according to Theorem 1. For both cases, we follow the procedure described in Sec. IV. For verification, we apply $\boldsymbol{u}_{0:H\text{-}1}$ to the true dynamics.

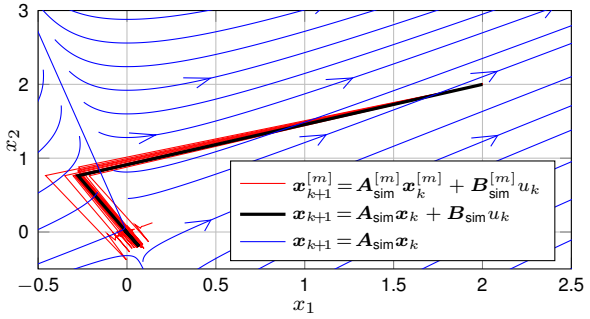

Fig. 4. Simulation linear system: Blue arrows show behavior for zero control input, red lines trajectories for 273 sampled scenarios used in control design and black shows behavior of the real system for the derived controller.

B. Results

Figure 3 shows that the proposed scenario-based approach optimizes the control law for all dynamic models drawn from the GP-SSM. The resulting control law achieves the desired behavior on the real system. For the unstable linear system, Fig. 4 shows that all 273 trajectories are stabilized by the computed control. The maximum cost of all scenarios is ≈ 7.58. Thus, Theorem 1 concludes that with probability 99% the computed control sequence result in a cost below ≈ 7.58 for at least 90% of all dynamics. It turns out, that the true dynamics lies within this 90% as a cost of ≈ 7.43 is achieved.

C. Discussion

The proposed scenario-based optimal control method for GP-SSM is applicable to a very broad class of nonlinear systems with arbitrary cost functions. As it relies on a purely nonparametric model, also complex functions are modeled with high precision. The resulting control law takes into account the uncertainty in the model by optimizing the performance over many realizations of the stochastic process. For the linear case, the approach allows to impose probabilistic guarantees for the performance of the control law. For the general nonlinear case, DDP does not guarantee globally optimal solutions. The result will therefore depend on the initialization of the control $\boldsymbol{u}_{0:H\text{-}1}$. However, convergence of the iterative approximation to a local optimal control is ensured according to [10]. The probabilistic guarantee provided by Theorem 1 requires the strong assumption (16). However, according to the no-free- lunch theorems [19], we cannot expect any generalization without any assumptions. In general, the hyperparameters are not guarantee to be optimal due to the non-convex likelihood optimization. However, for the linear case, it becomes convex and therefore optimal parameters are obtained. Generally, is is still unclear, whether randomization is the best way to achieve robustness. For a detailed discussion, we refer the reader to [20].

VII. CONCLUSION

This paper introduces a control design for a scenario-based interpretation of the GP-SSM. By drawing deterministic dynamic models from a GP and optimizing over all these scenarios, we consider the probabilistic nature of the model while avoiding approximations (like in the belief space view) or injecting process noise, which is not present in the true system (in the stochastic interpretation). The optimal control problem is solved using differential dynamic programming, more specifically the iterative LQ regulator. We show that for the specific case of a linear dynamical system and quadratic cost function, results from randomized robust optimization can be employed to derive probabilistic performance guarantees.

ACKNOWLEDGMENTS

The ERC Starting Grant "Control based on Human Models" supported this work under grant agreement no. 337654. We also thank the reviewers for their constructive feedback.

REFERENCES

[1] J. Kocijan, *Modelling and Control of Dynamic Systems Using Gaussian Process Models*. Springer, 2016. [Online]. Available: http://www.myilibrary.com?ID=875148

[2] C. E. Rasmussen and C. K. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA, USA: MIT Press, Jan. 2006.

[3] J. Umlauft, A. Lederer, and S. Hirche, "Learning stable Gaussian process state space models," in *American Control Conference (ACC)*. IEEE, May 2017, pp. 1499–1504.

[4] J. Umlauft, T. Beckers, M. Kimmel, and S. Hirche, "Feedback linearization using Gaussian processes," in *Conference on Decision and Control (CDC)*, Dec 2017, pp. 5249–5255.

[5] J. Umlauft, Y. Fanger, and S. Hirche, "Bayesian uncertainty modeling for programming by demonstration," in *International Conference on Robotics and Automation (ICRA)*. IEEE, Jun. 2017.

[6] Y. Fanger, J. Umlauft, and S. Hirche, "Gaussian processes for dynamic movement primitives with application in knowledge-based cooperation," in *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Oct. 2016, pp. 3913–3919.

[7] T. Beckers, J. Umlauft, D. Kulic, and S. Hirche, "Stable Gaussian process based tracking control of Lagrangian systems," in *Conference on Decision and Control (CDC)*, Dec 2017, pp. 5180–5185.

[8] D. H. Jacobson and D. Q. Mayne, *Differential dynamic programming*, ser. Modern analytic and computational methods in science and mathematics. New York: Elsevier, 1970.

[9] M. Gaggero, G. Gnecco, and M. Sanguineti, "Approximate dynamic programming for stochastic n-stage optimization with application to optimal consumption under uncertainty," *Computational Optimization and Applications*, vol. 58, no. 1, pp. 31–85, May 2014. [Online]. Available: https://doi.org/10.1007/s10589-013-9614-z

[10] E. Todorov and W. Li, "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems," in *American Control Conference (ACC)*. IEEE, 2005, pp. 300–306.

[11] G. C. Calafiore and L. Fagiano, "Robust model predictive control via scenario optimization," *IEEE Transactions on Automatic Control*, vol. 58, no. 1, pp. 219–224, Jan 2013.

[12] G. C. Calafiore and M. C. Campi, "The scenario approach to robust control design," *IEEE Transactions on Automatic Control*, vol. 51, no. 5, pp. 742–753, 2006.

[13] N. Srinivas, A. Krause, S. M. Kakade, and M. W. Seeger, "Information-theoretic regret bounds for Gaussian process optimization in the bandit setting," *IEEE Transactions on Information Theory*, vol. 58, no. 5, pp. 3250–3265, May 2012.

[14] T. Beckers, J. Umlauft, and S. Hirche, "Stable model-based control with Gaussian process regression for robot manipulators," in *World Congress of the International Federation of Automatic Control (IFAC)*, vol. 50, no. 1, 2017, pp. 3877 – 3884.

[15] Y. Pan, K. Saigol, and E. A. Theodorou, "Belief space stochastic control under unknown dynamics," in *American Control Conference (ACC)*, May 2017, pp. 3764–3770.

[16] Z. Xie, C. K. Liu, and K. Hauser, "Differential dynamic programming with nonlinear constraints," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 695–702.

[17] D. Mayne, "A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems," *International Journal of Control*, vol. 3, no. 1, pp. 85–95, 1966.

[18] M. C. Campi, S. Garatti, and M. Prandini, "The scenario approach for systems and control design," *Annual Reviews in Control*, vol. 33, no. 2, pp. 149–157, 2009.

[19] D. H. Wolpert, "The supervised learning no-free-lunch theorems," in *Soft Computing and Industry*. Springer, 2002, pp. 25–42.

[20] M. C. Campi, "Why is resorting to fate wise? A critical look at randomized algorithms in systems and control," *European Journal of Control*, vol. 16, no. 5, pp. 419–430, 2010.