2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 27 - October 1, 2021. Prague, Czech Republic

Learning to Control an Unstable System with One Minute of Data: Leveraging Gaussian Process Differentiation in Predictive Control

Ivan D. Jimenez Rodriguez, Ugo Rosolia, Aaron D. Ames, Yisong Yue

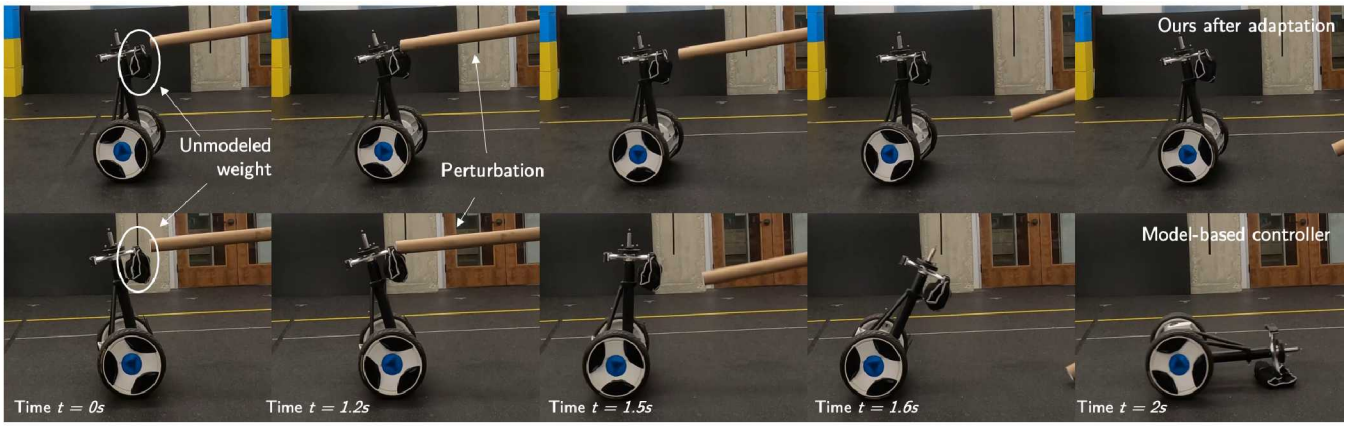


Fig. 1: Stabilizing a segway with an unmodelled weight against a force disturbance. **Top:** Our method is able to stabilize using a fully learned dynamics model that uses only 100 data points collected from one minute of data. **Bottom:** A state-of-the-art model predictive control policy using the nominal dynamics is unable to stabilize.

***Abstract*— We present a straightforward and efficient way to control unstable robotic systems using an estimated dynamics model. Specifically, we show how to exploit the differentiability of Gaussian Processes to create a state-dependent linearized approximation of the true continuous dynamics that can be integrated with model predictive control. Our approach is compatible with most Gaussian process approaches for system identification, and can learn an accurate model using modest amounts of training data. We validate our approach by learning the dynamics of an unstable system such as a segway with a 7-D state space and 2-D input space (using only one minute of data), and we show that the resulting controller is robust to unmodelled dynamics and disturbances, while state-of-the-art control methods based on nominal models can fail under small perturbations. Code is open sourced at https://github.com/learning-and-control/core.**

I. Introduction

System identification is frequently used in robotics to mitigate model imperfections using measured input-output data [1]–[5]. Managing these modeling errors can be critical to achieving desired performance or guaranteeing safety. This problem is particularly challenging in systems with unstable dynamics since even small modeling errors can integrate over time without control inputs that can directly dampen them. For instance, we experimentally show that running state-of-the-art model predictive control [6] on a 7-D state space and 2-D input space segway system using a misidentified model can lead to unsafe and unstable behavior, as depicted in Figure 1.

This work was supported by NSF awards 1637598, 1645832, 1932091, 1924526, and 1923239, and funding from AeroVironment, JPL and BMW.

[1] I. D. Jimenez Rodriguez, U. Rosolia, A. D. Ames and Yisong Yue are at the California Institute of Technology, Pasadena, USA. E-mails: {ivan.jimenez, urosolia, ames, yyue}@caltech.edu.

A useful system identification framework must balance computation time, accuracy and data efficiency. Furthermore, since data often cannot be collected for the entire state space of a real system, an estimate of model uncertainty is also useful to plan around gaps in the knowledge of the learned model. Because of these challenges, much of contemporary research has focused on learning residuals of an already well-developed nominal dynamics model [7]–[19].

In this paper, we aim to learn the full dynamics models of unstable robotic systems. Our goal is to develop a straightforward and data efficient method for system identification that can be easily integrated with state-of-the-art control methods. We ground our approach in Gaussian processes (GPs), which are a popular method for learning dynamics models [9], [11]–[13], [18], [20]–[23]. We leverage the differentiability of GPs [24], [25] to train a discrete-time dynamics model from training data of the form $(x_t, u_t, x_{t+\Delta t})$, while still recovering a state-dependent linearization of the dynamics that exploits the underlying continuous dynamics structure.

Learning a discrete-time linearizable dynamics has three key benefits. First, the approach can be very data efficient, as the differentiated GP model automatically infers the state-dependent linearization at every state. This differs from other approaches where the continuous dynamics model is learned directly and then used with collocation for approximation [26]. As shown in [27], learning the continuous dynamics rather than the discrete flow-map often requires higher sampling frequencies where measurement noise can become significant in practice. Second, one can use the estimated model with state-of-the-art model predictive control (MPC) methods [6] for effective and computationally efficient control synthesis that can handle state and input constraints.

The final benefit is that the approach is generic and can be applied to many GP-based modeling approaches as a drop-in subroutine.

The idea of using GPs for MPC is not new, but prior work either required using computationally expensive procedures [23], or limited themselves to learning only residual models [11]–[13], [18]. Other prior work use of GPs with Dynamic Programming frameworks that do not take into account state and input constraints [28]. Also, unlike reinforcement learning approaches that use Policy Gradient [29] or Value Iteration [30] with GPs, this method uses MPC to infer a policy. Learning the dynamics in MPC separates the policy from the dynamics and allows for using the same dynamics for different objectives.

We validate our approach by controlling unstable robotic systems, both in simulation and on a segway with 7-D state space and 2-D input space. Whereas state-of-the-art control methods can fail to stabilize the segway under small model mismatch, we show that one can robustly stabilize using our model trained on only one minute of data (see Figure 1 above). These results showcase the practical potential of our approach to significantly reduce the effort required for accurate system identification in unstable robotic systems.

II. A Differentiation-based Gaussian Process Model for Dynamics Estimation

In this section, we outline our approach for system identification using a differentiation-based Gaussian process model. We first describe and motivate learning a state-dependent linearized model in Section II-A. We then discuss Gaussian process preliminaries in Section II-B, and how to differentiate a GP to obtain a state-dependent linearized dynamics model in Section II-C.

A. State-Dependent Linearized Dynamics Modeling

We consider the following dynamical system:

$$\dot{x}(t) = f(x(t), u(t)), \tag{1}$$

with states $x \in R^n$, control inputs $u \in R^m$, and where f is unknown (not even the functional form). The above system is subject to the the following state and input constraints:

$$x(t) \in X \text{ and } u(t) \in U, \quad t \geq 0. \tag{2}$$

Our goal is to design a control policy $\pi : R^n \to R^d$ which maps states to actions. In order to compute such a policy we will use historical data to estimate the full system dynamics f, which will be leveraged in a predictive control scheme. We are particularly interested in systems that are passively unstable (e.g., a segway).

We assume access to a dataset of M state-input pairs $\{\hat{x}(iT), u(iT)\}_{i=0}^{M}$, where T is the sampling period. Visually, on the pendulum, this would correspond to the left-most plot in Figure 2. Furthermore, we assume the control action is applied using a sampling-and-hold strategy meaning that:

$$\hat{x}((i+1)T) = \int_{iT}^{(i+1)T} f(x(\tau), u(t))d\tau + x(t) + w, \tag{3}$$

where the noise w a zero-mean Gaussian, i.e, $w \sim N(0, \sigma^2)$.

Rather than performing motion planning directly on these discrete-time dynamics (as is common in the literature [20], [31]), we instead use (3) and a state-dependent linear approximation of the dynamics around the state-input pair $(\bar{x}, \bar{u})$:

$$\begin{aligned} f(x, u) &\approx A(\bar{x}, \bar{u})x + B(\bar{x}, \bar{u})u + C(\bar{x}, \bar{u}), \\ A(\bar{x}, \bar{u}) &= \frac{\partial f(\bar{x}, \bar{u})}{\partial x}, \\ B(\bar{x}, \bar{u}) &= \frac{\partial f(\bar{x}, \bar{u})}{\partial u}, \\ C(\bar{x}, \bar{u}) &= f(\bar{x}, \bar{u}) - \frac{\partial f(\bar{x}, \bar{u})}{\partial x}\bar{x} - \frac{\partial f(\bar{x}, \bar{u})}{\partial u}\bar{u}. \end{aligned} \tag{4}$$

This linearization of the dynamics can be related to a linearization of the discrete flow map in (3) as follows:

$$\begin{aligned} \hat{x}(t + \delta t) &\approx A(\bar{x}, \bar{u})x(t) + B(\bar{x}, \bar{u})u_t + C(\bar{x}, \bar{u}), \\ A(\bar{x}, \bar{u}) &= e^{A(\bar{x}, \bar{u})\delta t}, \\ B(\bar{x}, \bar{u}) &= A(\bar{x}, \bar{u})^{-1}(e^{A(\bar{x}, \bar{u})\delta t} - I)B(\bar{x}, \bar{u}), \\ C(\bar{x}, \bar{u}) &= A(\bar{x}, \bar{u})^{-1}(e^{A(\bar{x}, \bar{u})\delta t} - I)C(\bar{x}, \bar{u}). \end{aligned} \tag{5}$$

Although these matrices alone are sufficient for use in our MPC controller, through the use of matrix logarithms, the local linear approximation of the dynamics can be computed:

$$\begin{aligned} A(\bar{x}, \bar{u}) &\approx \frac{1}{\delta t}\log(A(\bar{x}, \bar{u})), \\ B(\bar{x}, \bar{u}) &\approx A(\bar{x}, \bar{u})^{-1}A(\bar{x}, \bar{u})B(\bar{x}, \bar{u}), \\ C(\bar{x}, \bar{u}) &\approx A(\bar{x}, \bar{u})^{-1}A(\bar{x}, \bar{u})C(\bar{x}, \bar{u}). \end{aligned} \tag{6}$$

As we shall see in Section III, having a state-dependent linearization is crucial for efficient integration with predictive control. In general, computing a state-dependent linearization with GPs in real-time can be challenging which is why most prior work resorts to approximating the GP using inducing inputs [31], [32], a time-varying state-input independent model [13] or learning the residual [11]–[13]. We will show in Section Section II-C how to solve for the matrices of the state-dependent linear dynamical approximation, (A, B, C), by taking derivatives of a Gaussian process dynamics model.

B. Gaussian Process Preliminaries

A Gaussian process (GP) is the defined by a mean function $\mu(s)$ and positive semidefinite covariance function $k(s, s')$. In this work we will primarily use two kernels: the Radial Basis Function (RBF) Kernel:

$$k_{rbf}(s, s') = \exp\left(-\frac{\|s - s'\|_2^2}{2\sigma^2}\right), \tag{7}$$

and the Periodic Kernel:

$$k_p(s, s') = \exp\left(-\frac{2\sin^2(\frac{\pi\|s - s'\|}{\omega_2})}{\ell^2}\right), \tag{8}$$

where σ, ω and ℓ are tunable parameters. We will also construct composite kernels by exploiting the fact that the product of kernels is a valid kernel, in order to encode geometric properties of the dynamical system. In particular,

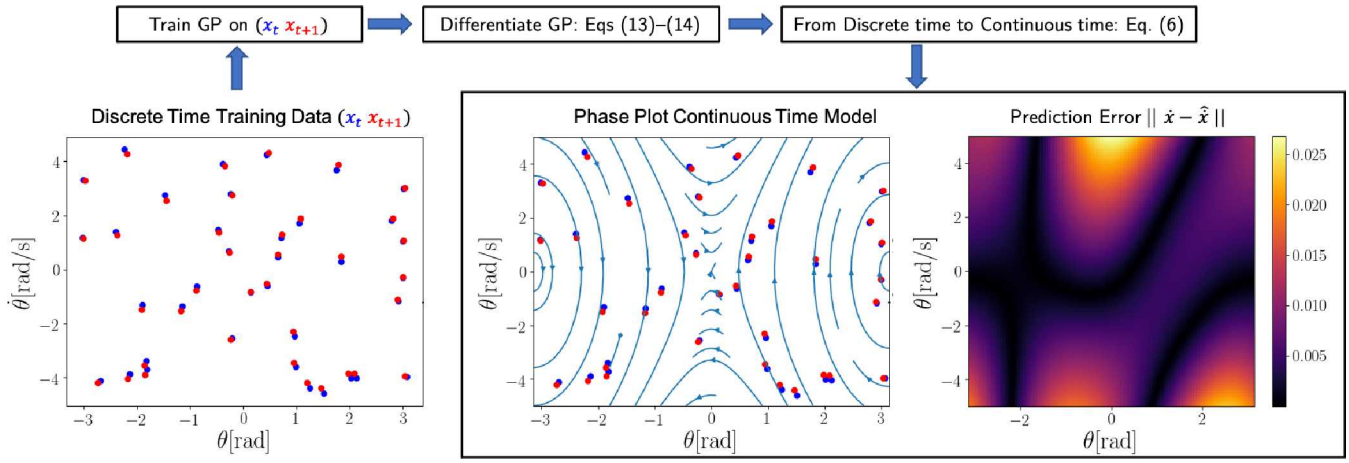


Fig. 2: System identification results for a simulated pendulum. Left-to-Right: The dataset collected (selecting 30 initial points uniformly at random and integrating forward for 0.01s); Phase plot of estimated dynamics with dataset overlaid; Point-wise error between true and estimated dynamics. Phase plots are computed on a 100×100 grid. We see that the error is small and captures the behavior of the system even in regions with few data points.

the Periodic Kernel is useful for modeling angular coordinates, whereas the RBF kernel is more suitable for Euclidean coordinates.

Samples of a GP take the form:

$$h \sim GP\left(\mu\left(\begin{matrix}x\\u\end{matrix}\right), k\left(\begin{matrix}x\\u\end{matrix}, \begin{matrix}x'\\u'\end{matrix}\right)\right), \tag{9}$$

where function samples approximate the integral of the dynamics as follows:

$$h\begin{pmatrix}x(t)\\u(t)\end{pmatrix} \approx \int_t^{t+\delta t} f(x(\tau), u(\tau))d\tau. \tag{10}$$

Multi-dimensional outputs are predicted with an independent GP for each output. For a test input $s = \tilde{x}\ \tilde{u}$, the mean and variance are computed as:

$$\begin{aligned}\mu(s) &= \mathbb{E}_h[h(s)] = k(s, X)\left(k(X, X) + \sigma_6 I\right)^{-1} Y,\\ \sigma_h^2 &= \mathbb{V}_f[h(s)]\\ &= k(s, s) - k(s, X)\left(k(X, X) + \sigma_6 I\right)^{-1} k(X, s).\end{aligned}$$

C. Differentiating a Gaussian Process

From [24], [25], we know that the derivative of a GP is another GP. For the following derivation it is sufficient for the kernel function to be differentiable with respect to the both of its parameters (which is true for both the RBF and Periodic kernels). For an input s we define the derivative of a GP as follows:

$$h' \sim GP(\mu', k'(s, s')), \tag{11}$$

where $h' : \mathbb{R}^{n+m} \to \mathbb{R}^{n+m}$ is the gradient of sampled function h. We derive the mean of the GP derivative as follows:

$$\mu'(S) = \mathbb{E}\left[\frac{\partial}{\partial s}h\right] = \frac{\partial}{\partial s}\mathbb{E}[h] = \frac{\partial}{\partial s}\mu(s), \tag{12}$$

This GP in (9) is related to the linear approximation in (5) as follows:

$$\begin{aligned}A(\bar{x}, \bar{u}) &\approx \frac{\partial h(\bar{x}, \bar{u})}{\partial x} + I,\\ B(\bar{x}, \bar{u}) &\approx \frac{\partial h(\bar{x}, \bar{u})}{\partial u},\\ C(\bar{x}, \bar{u}) &\approx h(\bar{x}, \bar{u}) + \bar{x} - A(\bar{x}, \bar{u})\bar{x} - B(\bar{x}, \bar{u})\bar{u},\end{aligned} \tag{13}$$

These approximates are derived by noting that the GP approximates the integral in (3) which concludes the derivation.

Graphically, this relationship can be seen in the center plot of Figure 2 where the data points are overlapped with the state-dependent continuous linear approximation we computed using Equation (6). Note that since we are training on M, a dataset of state input pairs, we are still learning the discrete time flow map shown in (3). A key aspect of our contribution is to re-interpret the learned dynamics in the context of (1) to directly infer a local linear approximation that is amenable to MPC. This allows our method to be retroactively applied to previous GP-based modeling work that learns a discrete transition model.

III. Control Design

We now describe our control strategy, which is based on state-of-the-art methods for model predictive control (MPC) [6]. This approach works generically with the state-dependent linearized model described in (5). First, we introduce a Finite Time Optimal Control Problem (FTOCP) which is based on a simplified Affine Time-Varying (ATV) model. We then present the proposed algorithm that at each time t solves an FTOCP where the ATV model is updated leveraging the state-dependent linearized model from (5). Our algorithm applies the first action of the planned trajectory to the system and the entire process is repeated at the next time step $t + 1$, yielding to a receding horizon strategy also referred to as model predictive control.

A. Finite Time Optimal Control

At time t and for system's state $x(t)$, we define the following finite time optimal control problem (FTOCP):

$$
\begin{aligned}
J(x(t)) = \min_{u_t} \; & \sum_{k=t}^{t+N-1} l(x_{k|t}, u_{k|t}) + V(x_{t+N|t}), \\
\text{s.t.} \; & x_{k+1|t} = A_k x_{k|t} + B_k u_{k|t} + C_k, \\
& x_{k|t} \in X, u_{k|t} \in U, \; k \in \{t, \dots, t+N\} \\
& x_{t|t} = x(t)
\end{aligned} \tag{14}
$$

where $u_t = [u_{t|t}, \dots, u_{t+N|t}]$ is a sequence of open-loop control actions, the stage cost $l : \mathbb{R}^n \times \mathbb{R}^d \to \mathbb{R}$, the terminal cost $V : \mathbb{R}^n \to \mathbb{R}$ and the sets $X \subseteq \mathbb{R}^n$ and $U \subseteq \mathbb{R}^d$ represent the state and input constraints, respectively. In the above problem, the time varying matrices A_k, B_k and C_k define a discrete time ATV approximation of the true system (1) and we will compute them using the differentiable GP from Section II-C.

We denote the optimal state-input sequences to the FTOCP (14) as:

$$
(\tilde{x}_t, \tilde{u}_t) = \tilde{x}_{t|t}, \tilde{u}_{t|t}, \dots, \tilde{x}_{t+N|t}, \tilde{u}_{t+N-1|t} \tag{15}
$$

which minimize the predicted cost while satisfying state and input constraints from (2). When the above optimal predicted trajectory is computed at time t, we have that $x_{k|t}$ denotes the predicted state of at time k. This notation will be useful later on when we are going to differentiate between the optimal state $x_{k|t}$ at time k predicted at time t and the optimal state $x_{k|t+1}$ at time k predicted at time $t+1$. In what follows, we use the optimal state-input sequences (15) to synthesize a control policy for the dynamical system (1).

B. Policy Synthesis

This section describes the control synthesis strategy. At each time t, we solve the FTOCP (14), where the time varying matrices $\{A_k, B_k, C_k\}_{k=t}^{t+N-1}$ are computed using the differentiable GP evaluated along the candidate state input sequences:

$$
(\bar{x}_t, \bar{u}_t) = [(\bar{x}_{t|t}, \bar{u}_{t|t}), \dots, (\bar{x}_{t+N|t}, \bar{u}_{t+N-1|t})]
$$

At time $t = 0$, we initialize the candidate trajectory with an initial guess and afterwards we update the candidate solution using the optimal trajectory from (15), as shown in Algorithm 1. In particular, in Algorithm 1 we update the candidate trajectory by shifting the optimal solution computed as the previous time time (Lines 3-8). Afterwards, we update ATV matrices used to define the FTOCP problem (14). Finally, we solve problem (14) and we store the optimal state-input trajectories. The strategy described in Algorithm 1 is repeated at each time t based on the new measurement x_t.

It is clear that the prediction model defined by the ATV matrices $\{A_k, B_k, C_k\}_{k=t}^{t+N-1}$ plays a crucial role in determining the success of the MPC. If the prediction model is inaccurate, then the closed-loop system will deviate from the planned trajectory. This deviation may result in poor closed-loop performance and safety constraint violation. We validate this point in our experiments showing that controlling using an inaccurate model can be unsafe, thus highlighting the need to quickly learn accurate dynamics models.

Algorithm 1 Control Policy

Init Parameters: x_{t-1}, u_{t-1}, $GP(\mu^0(s), k^0(s, s'))$
Input: x_t
if $t > 0$ **then** . Update Candidate Trajectory
  Set $\bar{x}_{k|t} = x_{k|t-1}$, $k \in \{t, \dots, t+N-1\}$
  Set $\bar{u}_{k|t} = u_{k|t-1}$, $k \in \{t, \dots, t+N-2\}$
  Set $\bar{x}_{t+N|t} = x_{t+N-1|t-1}$
  Set $\bar{u}_{t+N-1|t} = u_{t+N-2|t-1}$
end if
for $k \in \{t, \dots, t+N-1\}$ **do** . Update Model
  Set $A_k = A(\bar{x}_{k|t}, \bar{u}_{k|t})$ from expectation of (13)
  Set $B_k = B(\bar{x}_{k|t}, \bar{u}_{k|t})$ from expectation of (13)
  Set $C_k = C(\bar{x}_{k|t}, \bar{u}_{k|t})$ from expectation of (13)
end for
Solve the FTOCP (14) with $\{A_k, B_k, C_k\}_{k=t}^{t+N-1}$
Store $x_t = [x_{t|t}, \dots, x_{t+N|t}]$
Store $u_t = [u_{t|t}, \dots, u_{t+N-1|t}]$
Set $u_t = u_{t|t}$
Outputs u_t

IV. Simulation Results

In this section, we validate our approach in simulation on two unstable systems: the inverted pendulum and the segway. The goal of this evaluation is to provide both an intuition as well as a demonstration of the theoretical limits for our approach. Specifically, we aim to address:

- Can we learn an accurate dynamics model with few training examples?
- Can we integrate our dynamics model with control for steering and motion planning?

A. Pendulum Simulation

We first test on a continuous time inverted pendulum. The state of the system is $x = [\theta, \dot{\theta}]$ with torque input u. The system has a point mass of 0.25kg and a length of 0.5m.

1) Data Collection & System Identification: First, we estimate the discrete flow map for the unactuated system using 37 samples collected uniformly at random so that $\theta(t_i) \sim U(-\pi, \pi)$ and $\dot{\theta}(t_i) \sim U(-5, 5)$. We capture the

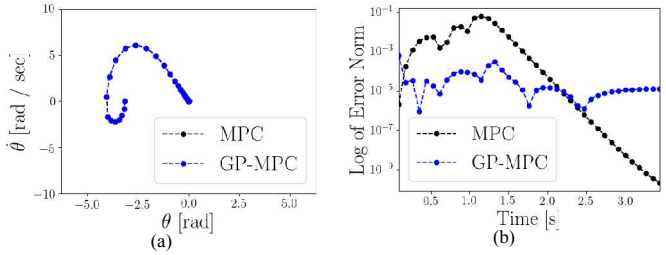


Fig. 3: Figure 3a shows the closed-loop trajectory of the pendulum. MPC with both the true model and the GP produce near identical trajectories. Figure 3b shows the one-step prediction error of the MPC policy for both models.

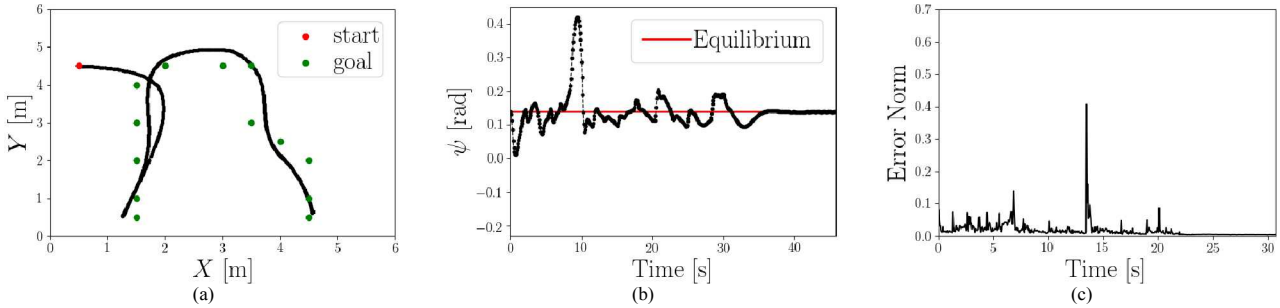


Fig. 4: Control results on simulated segway. Figure 4a, we plot the position of the segway as it reaches a sequence of target goals. Figure 4b shows : the angle of the segway with respect to the upright position. Notice that the segway must deviate from its equilibrium in order to accelerate forwards or backwards. Figure 4c shows the one-step prediction error of the MPC policy using GP dynamics.

geometric structure of the pendulum's sate-space by using the following kernel:

$$k(x, x') = k_p(\theta, \theta')k_{rbf}(\theta, \theta'), \tag{16}$$

since $x \in S^1 \times \mathbb{R}$. Next, we use a local linear approximation to compute a point-wise estimate of the continuous dynamics as shown in (13). The entire dataset of state transitions is shown in the left plot of Figure 2.

2) Evaluating Model Accuracy: We divide the state-space of the pendulum into a 100 x 100 grid. For each point in the grid we compute the true value of $\dot{x}$ using (1) and an estimated value using the linear approximation in (4) where A, B and C are computed using the derivative of the GP as in (13). The phase plot in Figure 2 correspond to the direction of the estimated $\dot{x}$. For each point in the grid we can compute the 2-norm of the error which is shown in the heat map on the right. Overall, we can see that our approach can recover an accurate state-dependent linearization of the true continuous time dynamics.

3) Evaluating Motion Planning: Next, we consider the task of steering the pendulum to the upright position starting from $x(0) = (-\pi, 0)$ with the input constraint set $U = \{u \mid -.6 < u < .6\}$. Notice that with these constraints the pendulum is unable to reach the goal in a single swing. For this task, we collect a new dataset of 34 uniformly distributed samples where states are sampled as before and $u(t_i) \sim U(-.6, .6)$. Finally, we run the control policy from Algorithm 1. Both our strategy and an MPC using true dynamics are able to swing up the pendulum reaching the unstable equilibrium state as shown in Figure 3a.

To evaluate the performance of the GP model for motion planning, we compute the difference between the first state predicted by the MPC policy and the actual state observed by the system. Figure 3b shows the errors for the true model and the learned model at each time step of the pendulum's swing-up. We see that throughout the pendulum swing up the MPC model has a higher prediction error than the GP. Towards the end, as the pendulum stabilizes, the error of the MPC policy with the true model falls to 0 while the GP controller maintains a low but stable error. We expect the MPC with continuous time dynamics to have a slightly higher error than the GP as the continuous time dynamics are linearized and discretized to compute the predicted trajectory, while GP provides an estimate of the discrete-time map which is used to compute the next step.

B. High-Fidelity Segway Simulation

We next evaluate our strategy on a high-fidelity simulated segway based on the 6-D state space and 2-D input space system shown in Figure 5. The state of a segway is $x = [X, Y, \theta, v, \dot{\theta}, \psi, \dot{\psi}]$, where (X, Y) represents the position of the center of mass, $(\theta, \dot{\theta})$ the heading angle and yaw rate, v the velocity and $(\psi, \dot{\psi})$ the rod's angle and angular velocity. The control input $u = [T_l, T_r]$, where T_l and T_r are the torques to the left and right wheel motors, respectively. For all experiments, we limit $|T_l| < 6$ and $|T_r| < 6$.

1) Data Collection & System Identification: Recall that we are learning the mapping shown in (5). As a prior, we know that X and Y have no effect on the dynamics of the system. Therefore we only need to learn a mapping from the state excluding X and Y at the current time step to the change in state at the next time step. We encode the property of θ being an angle via the following kernel:

$$k(x, x') = k_p(\theta, \theta')k_{rbf}(s, s'), \tag{17}$$

where $s = [v, \theta, \psi, \dot{\psi}]$. Although ψ is also an angle, since the system cannot rotate about that axis without catastrophic failure, we use a regular RBF kernel for it.

In simulation, we record the segway performing a task consisting of 1000 state-transitions at a frequency $T = 0.05$ which is approximately one minute of data. We then find 180 clusters using a hierarchical clustering algorithm and select the nearest neighbor for each cluster as the data-point for the training set. We test the ability of the our strategy to perform the same task that was used to collect the data but with the GP dynamics model.

2) Evaluating Motion Planning: Figure 4a shows the path that the segway takes while reaching the targets. Once the segway is within 1m of a goal the next one is provided. Notice that peaks and troughs in Figure 4b correspond to moments of forwards and backwards acceleration (since the segway must tilt to move forward). Those same moments of high acceleration also match with the peaks of high one-step prediction error observed in Figure 4c.

V. Experimental Results on Segway

We finally evaluate our approach experimentally on the segway system (see Figure 5). The state representation is the same as in Section IV-B.

We aim to demonstrate that:

- Our method can control a physical open-loop unstable system to perform a simple move-forward task.
- Our method is able to overcome perturbations with unmodelled dynamics in a physical open-loop unstable system where a state-of-the art MPC controller fails.

1) Data Collection & System Identification: We record the trajectory of the segway performing a task that takes 1000 measurements recorded every $\mathsf{T} = 0.05\mathsf{s}$ to complete. The data is then preprocessed using the same procedure as for the simulated system. We evaluate on two tasks: the first is a standard moving forward task while staying upright, and the second is stabilizing task under a external force disturbance. We note that since this a real system an estimation is performed online with on-board sensors there is significant estimation error as well. Although our method is capable of running at 20Hz with up to 300 data-points, we required less data than this for the experiments.

2) Simple Move-Forward Task: We start by considering the move-forward task. For this we only require 130 data points. As can be observed in Figure 7, the system is able to stabilize at a point, move forward and stabilize close to the new location with some minor oscillations around the target point, as highlighted in Figure 6. The first and second peaks in Figure 6b correspond to the acceleration and deceleration respectively. Notice that due to a combination of modeling and estimation estimation error, the segway balances slightly off the equilibrium point. Finally, from Figure 6c we see that the model error spikes at the moments of high acceleration.

3) Robustness To Perturbations: We now evaluate the performance of the learned model under perturbations. To test the robustness of the learned model, we start by collecting 100 data points from a dataset of the MPC policy with nominal dynamics completing the task with an unmodelled weight of 2kg. Although this results in slightly different behavior, the amount of data collected is the same as in previous experiments. Next, we attach 4kg of unmodelled weight as shown in Figure 1. Notice that the weight is not perfectly centered and that it is allowed to sway back and forth from its point of contact.

In Figure 8 and Figure 9, we can see the result of applying force perpendicular to the axis of the wheels to the MPC policy with the nominal and GP dynamics, respectively. Both controllers have a spike in input following each disturbance, and in both cases the control action is saturated. Notice that because of the symmetry of the nominal model, the MPC policy applies the same force on each input as shown in Figure 9a. Meanwhile the learned dynamics captures some of the asymmetry resulting from the weights which causes uneven outputs and a more robust system. In Figure 8b and Figure 9b, we can see that even though the initial disturbances are of similar magnitude, the controller with nominal dynamics exhibits much larger oscillations and falls after the third perturbation. Although both models have sharp increases in one-step prediction error after a disturbance, the MPC model reaches much higher one-step prediction errors than with the GP, as shown in Figure 8c and Figure 9c.

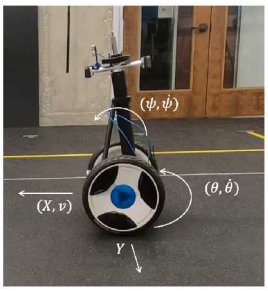


Fig. 5: Experimental platform.

VI. Discussion and Future Work

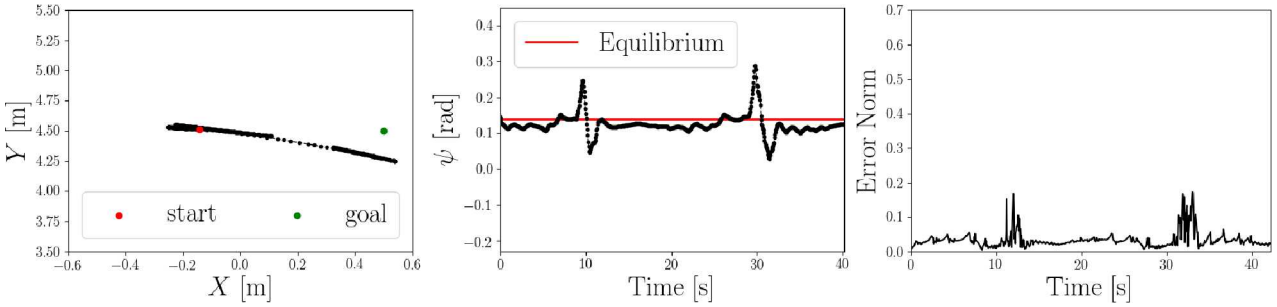
We presented a methodology for full dynamics learning that has been validated on an open-loop unstable robotic system. Using one minute of highly correlated data we are able estimate an accurate enough model for motion planning that is resilient to perturbations. Furthermore, the results presented in this paper are for a worst-case scenario where no useful prior is provided. Finally, our approach is generic and can be applied to many Gaussian process modeling approaches as a drop-in subroutine.

There are many directions for future work. A natural one is to study even higher-dimensional systems where one would likely need to combine learning with a prior nominal model. Another direction is dealing with noise in the state estimation as well as delays, which are significant issues for unstable dynamical systems. Using techniques to correct noisy estimation data would significantly improve the performance of our method in real systems. This would be true when dealing with outlying measurements that strongly violate the Gaussian assumption implicit in the GP. One could also consider integration with perceptual systems [33].

Another direction for future work is how to intelligently and autonomously collect training data. A relevant line of work here is the area of safe exploration [9], [34]–[38]. It is also important to understand the fundamental limits of how much data we need to learn a reliable model, a concept known as sample complexity in the machine learning literature [39].

This work exploited the differentiability of Gaussian processes, but largely ignored the uncertainty quantification aspect. In cases where there are more complex constraints to be satisfied, such as reachability [40] or chance constraints [35], it would be interesting to develop a more holistic framework that reasons about uncertainty quantication in differentiated GP models.

A final direction for future work is scalability. For more complex systems, it would be beneficial to collected and store more data points for estimating the GP model. However, it is known that the computational complexity of GP inference can scale poorly with the amount of training data. Leveraging various methods for scaling up GP training and inference could be beneficial [41], [42].



(a) (b) (c)

Fig. 6: The 7-D state space and 2-D input space segway going between two points. In Figure 6a we plot the physical position of the segway. In Figure 6b we see : the angle between segway's pole and the upright position. The two peaks correspond to the segway accelerating and decelerating. Figure 6c shows the MPC policy's one-step prediction error using the GP dynamics.

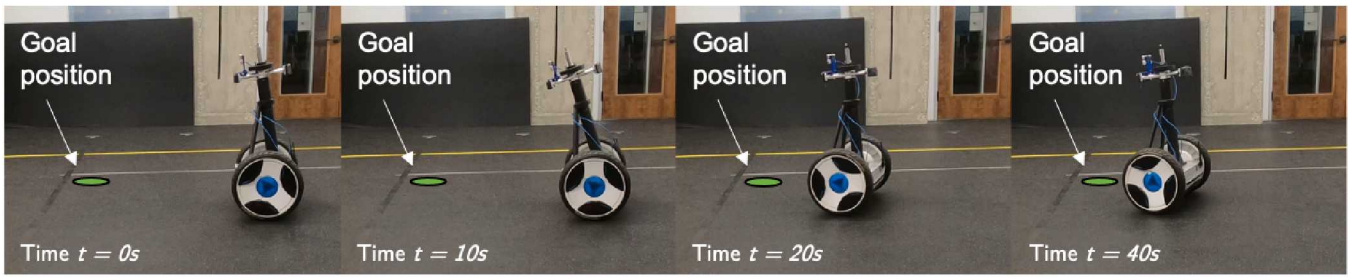
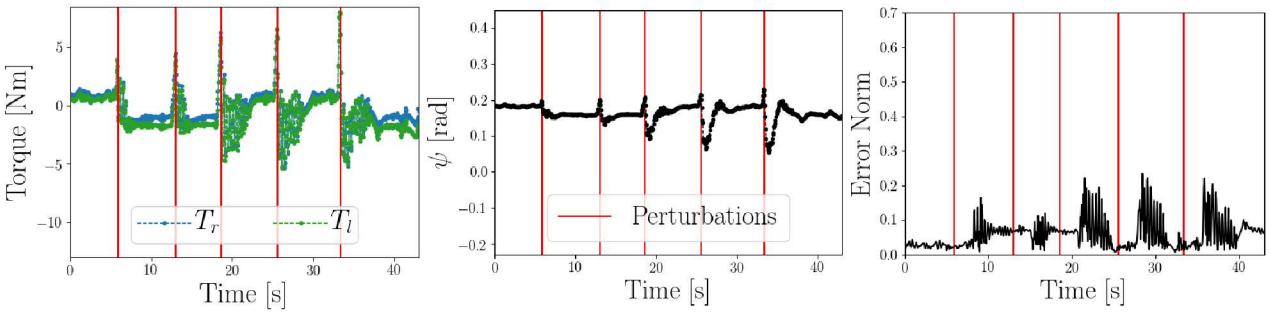
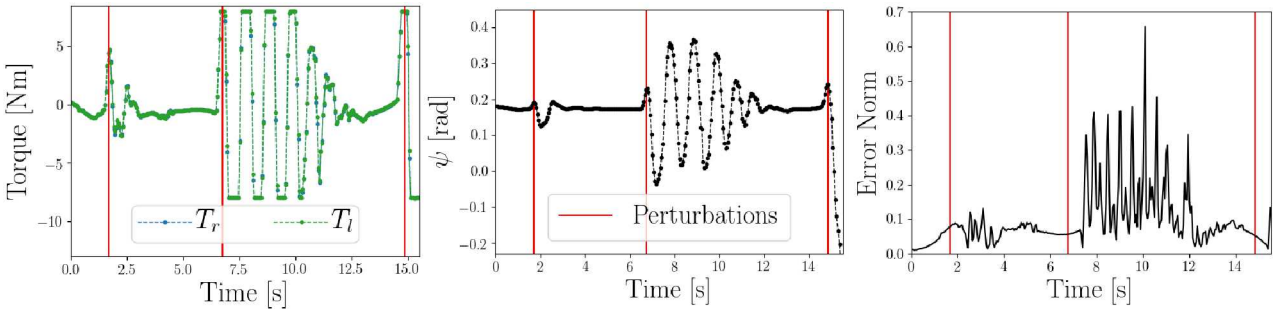


Fig. 7: This image sequence shows the segway going to a point using the full order dynamics learned with our GP method.



(a) (b) (c)

Fig. 8: The MPC policy with GP dynamics responding to 5 perturbations. The segway remains stable after each disturbance. Figure 8a shows the inputs spiking after each disturbance. The difference between inputs suggests the learned dynamics model captures asymmetries induced by the placement and sway of the unmodelled weights. Figure 9b plots the angle of the segway with respect to the ground. Figure 9c shows the MPC's one step prediction error remains low through all disturbances.



(a) (b) (c)

Fig. 9: The MPC policy with nominal dynamics responding three perturbations. The segway remains stable after the first disturbance, oscillates before stabilizing for the second disturbance and falls down after the third disturbance. Figure 8a shows the input spike on each input after the disturbances. Notice that for all three perturbations, the both motors act in unison to stabilize the system. Figure 9b shows the angle of the segway with respect to the ground. Figure 9c shows the MPC's one-step prediction error using the nominal dynamics. Notice that the oscillations cause the weight to swing which magnify the prediction error.

ACKNOWLEDGMENTS

We would like to thank Andrew Singletary and Ellen Novoseller for their help in setting up the robotics and the mathematical derivations, respectively. We would also like to thank the follwing software packages: PyTorch [43], CVXPY [44] and Gurobi [45].

REFERENCES

[1] T. L. Lai, C. Z. Wei *et al.*, "Least squares estimates in stochastic regression models with applications to identification and control of dynamic systems," *Annals of Statistics*, vol. 10, no. 1, pp. 154–166, 1982.

[2] P. K. Khosla and T. Kanade, "Parameter identification of robot dynamics," in *1985 24th IEEE Conference on Decision and Control*, pp. 1754–1760.

[3] S. Chen, S. Billings, and P. Grant, "Non-linear system identification using neural networks," *International journal of control*, vol. 51, no. 6, pp. 1191–1214, 1990.

[4] U. C aydas and S. Ekici, "Support vector machines models for surface roughness prediction in CNC turning of AISI 304 austenitic stainless steel," vol. 23, no. 3, pp. 639–650. [Online]. Available: https://doi.org/10.1007/s10845-010-0415-2

[5] M. Willis, H. Hiden, M. Hinchliffe, and B. McKAY, "Systems modelling using genetic programming," p. 6.

[6] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[7] A. Aswani, H. Gonzalez, S. S. Sastry, and C. Tomlin, "Provably safe and robust learning-based model predictive control," *Automatica*, vol. 49, no. 5, pp. 1216–1226, 2013.

[8] D. Papadimitriou, U. Rosolia, and F. Borrelli, "Control of unknown nonlinear systems with linear time-varying MPC." [Online]. Available: http://arxiv.org/abs/2004.03041

[9] T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause, "Learning-based model predictive control for safe exploration," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 6059–6066.

[10] U. Rosolia and F. Borrelli, "Learning how to autonomously race a car: a predictive control approach," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 6, pp. 2713–2719, 2019.

[11] L. Hewing, J. Kabzan, and M. N. Zeilinger, "Cautious model predictive control using gaussian process regression," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 6, pp. 2736–2743, 2019.

[12] J. Kabzan, L. Hewing, A. Liniger, and M. N. Zeilinger, "Learning-based model predictive control for autonomous racing," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3363–3370, 2019.

[13] E. D. Klenske, M. N. Zeilinger, B. Schölkopf, and P. Hennig, "Gaussian process-based predictive control for periodic error correction," *IEEE Transactions on Control Systems Technology*, vol. 24, no. 1, pp. 110–121, 2015.

[14] G. Shi, X. Shi, M. O'Connell, R. Yu, K. Azizzadenesheli, A. Anandkumar, Y. Yue, and S.-J. Chung, "Neural lander: Stable drone landing control using learned dynamics," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 9784–9790.

[15] G. Shi, W. Hönig, Y. Yue, and S.-J. Chung, "Neural-swarm: Decentralized close-proximity multirotor control using learned interactions," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 3241–3247.

[16] A. Taylor, A. Singletary, Y. Yue, and A. Ames, "Learning for safety-critical control with control barrier functions," in *Learning for Dynamics and Control*. PMLR, 2020, pp. 708–717.

[17] A. J. Taylor, V. D. Dorobantu, H. M. Le, Y. Yue, and A. D. Ames, "Episodic learning with control lyapunov functions for uncertain robotic systems," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 6878–6884.

[18] A. H. Chang, C. M. Hubicki, J. J. Aguilar, D. I. Goldman, A. D. Ames, and P. A. Vela, "Learning to jump in granular media: Unifying optimal control synthesis with gaussian process-based regression," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 2154–2160.

[19] M. Bujarbaruah, X. Zhang, U. Rosolia, and F. Borrelli, "Adaptive mpc for iterative tasks," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 6322–6327.

[20] M. P. Deisenroth, D. Fox, and C. E. Rasmussen, "Gaussian processes for data-efficient learning in robotics and control," vol. 37, no. 2, pp. 408–423. [Online]. Available: http://ieeexplore.ieee.org/document/6654139/

[21] L. Hewing, J. Kabzan, and M. N. Zeilinger, "Cautious model predictive control using gaussian process regression," vol. 28, no. 6, pp. 2736–2743. [Online]. Available: https://ieeexplore.ieee.org/document/8909368/

[22] L. Wang, E. A. Theodorou, and M. Egerstedt, "Safe learning of quadrotor dynamics using barrier certificates," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 2460–2465.

[23] J. Kocijan, R. Murray-Smith, C. E. Rasmussen, and A. Girard, "Gaussian process model based predictive control," in *Proceedings of the 2004 American control conference*, vol. 3. IEEE, 2004, pp. 2214–2219.

[24] A. McHutchon, "Differentiating gaussian processes," 2013. [Online]. Available: http://mlg.eng.cam.ac.uk/mchutchon/DifferentiatingGPs.pdf

[25] E. Solak, R. Murray-smith, W. E. Leithead, D. J. Leith, and C. E. Rasmussen, "Derivative observations in gaussian process models of dynamic systems," p. 8.

[26] G. Lee, S. S. Srinivasa, and M. T. Mason, "GP-ILQG: Data-driven robust optimal control for uncertain nonlinear dynamical systems." [Online]. Available: http://arxiv.org/abs/1705.05344

[27] M. E. Levine and A. M. Stuart, "A framework for machine learning of model error in dynamical systems," 2021.

[28] Y. Pan and E. Theodorou, "Probabilistic differential dynamic programming," p. 9.

[29] M. P. Deisenroth and C. E. Rasmussen, "PILCO: A model-based and data-efficient approach to policy search," p. 8.

[30] M. Lutter, S. Mannor, J. Peters, D. Fox, and A. Garg, "Value iteration in continuous actions, states and time." [Online]. Available: http://arxiv.org/abs/2105.04682

[31] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-based model predictive control: Toward safe learning in control," vol. 3, no. 1, pp. 269–296, eprint: https://doi.org/10.1146/annurev-control-090419-075625. [Online]. Available: https://doi.org/10.1146/annurev-control-090419-075625

[32] J. Quinonero-Candela and C. E. Rasmussen, "A unifying view of sparse approximate gaussian process regression," *The Journal of Machine Learning Research*, vol. 6, pp. 1939–1959, 2005.

[33] S. Dean, N. Matni, B. Recht, and V. Ye, "Robust guarantees for perception-based control," in *Learning for Dynamics and Control*. PMLR, 2020, pp. 350–360.

[34] A. Liu, G. Shi, S.-J. Chung, A. Anandkumar, and Y. Yue, "Robust regression for safe exploration in control." [Online]. Available: http://arxiv.org/abs/1906.05819

[35] Y. K. Nakka, A. Liu, G. Shi, A. Anandkumar, Y. Yue, and S.-J. Chung, "Chance-constrained trajectory optimization for safe exploration and learning of nonlinear systems," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 389–396, 2020.

[36] Y. Sui, A. Gotovos, J. Burdick, and A. Krause, "Safe exploration for optimization with gaussian processes," in *International Conference on Machine Learning*. PMLR, 2015, pp. 997–1005.

[37] A. Wachi, Y. Sui, Y. Yue, and M. Ono, "Safe exploration and optimization of constrained mdps using gaussian processes," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[38] M. Turchetta, F. Berkenkamp, and A. Krause, "Safe exploration for interactive machine learning," *arXiv preprint arXiv:1910.13726*, 2019.

[39] K. P. Wabersich and M. Zeilinger, "Bayesian model predictive control: Efficient model exploration and regret bounds using posterior sampling," in *Learning for Dynamics and Control*. PMLR, 2020, pp. 455–464.

[40] 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 IEEE Conference on Decision and Control*. IEEE, 2014, pp. 1424–1431.

[41] A. Wilson and H. Nickisch, "Kernel interpolation for scalable structured gaussian processes (kiss-gp)," in *International Conference on Machine Learning*. PMLR, 2015, pp. 1775–1784.

[42] E. Snelson and Z. Ghahramani, "Local and global sparse gaussian process approximations," in *Artificial Intelligence and Statistics*. PMLR, 2007, pp. 524–531.

[43] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alche´-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035.

[44] A. Agrawal, R. Verschueren, S. Diamond, and S. Boyd, "A rewriting system for convex optimization problems," *Journal of Control and Decision*, vol. 5, no. 1, pp. 42–60, 2018.

[45] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2021. [Online]. Available: https://www.gurobi.com