Robotics: Science and Systems 2024
Delft, Netherlands, July 15-July 19, 2024

Computation-Aware Learning for Stable Control with Gaussian Process

Wenhan Cao[1,2] Alexandre Capone[3,4] Rishabh Yadav[1] Sandra Hirche[4] Wei Pan[1]
[1]University of Manchester [2]Tsinghua University [3]Carnegie Mellon University [4]Technical University of Munich

***Abstract*—In Gaussian Process (GP) dynamical model learning for robot control, particularly for systems constrained by computational resources like small quadrotors equipped with low-end processors, analyzing stability and designing a stable controller present significant challenges. This paper distinguishes between two types of uncertainty within the posteriors of GP dynamical models: the well-documented mathematical uncertainty stemming from limited data and computational uncertainty arising from constrained computational capabilities, which has been largely overlooked in prior research. Our work demonstrates that computational uncertainty, quantified through a probabilistic approximation of the inverse covariance matrix in GP dynamical models, is essential for stable control under computational constraints. We show that incorporating computational uncertainty can prevent overestimating the region of attraction, a safe subset of the state space with asymptotic stability, thus improving system safety. Building on these insights, we propose an innovative controller design methodology that integrates computational uncertainty within a second-order cone programming framework. Simulations of canonical stable control tasks and experiments of quadrotor tracking exhibit the effectiveness of our method under computational constraints.**

I. Introduction

Dynamical model learning using Gaussian processes (GP) is popular in the field of robotic control [1–3]. The ability to update the model online is particularly desirable as it enables robotic systems to adapt to unpredictable and changing environments [4–7]. Using dynamically updated models makes it feasible to design controllers with stability guarantees [8–12]. This adaptability is crucial in aerial robotics, where the capacity to quickly adjust to new conditions is key to maintaining both stability and optimal performance. Existing approaches assume ample computational resources when learning dynamical models online [13–20]. However, low-end computational hardware, such as those of tiny robots [21], often faces limitations, and even high-end systems can suffer performance degradation due to insufficient resource allocation or high temperatures [22]. Consequently, computational errors are inevitable, whether due to the early termination of the learning process or the processing of only a partial batch of data.

Computational errors in dynamical model learning can deteriorate control performance. An illustrative example can be found in our experiment (see Section VI), where we consider a quadrotor tracking control task. This task utilizes the conjugate gradient (CG) method, an iterative approach, to compute the GP posterior dynamical model. We then synthesize a controller with constraints constructed by the control Lyapunov function (CLF) [19, 20, 23]. The results demonstrate that early termination of the CG algorithm can lead to increased tracking error (see Fig. 7). Intuitively, under a fixed operation time budget, low-end computational hardware allows fewer iterations, resulting in larger model errors, and consequently, greater tracking errors than its high-end counterpart.

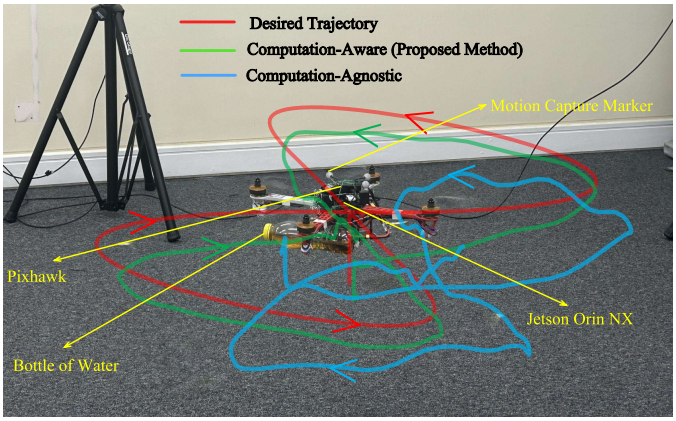

Fig. 1: Quadrotor tracking control under computational constraints. The green line denotes the proposed computation-aware learning control method in this paper, showing improved tracking accuracy due to adaptive, conservative stability constraints compared to the agnostic one (blue line).

Given the inevitability of computational errors during dynamical model learning, understanding their impact on system stability and incorporating them into controller design is crucial. To address this, this paper proposes a computation-aware learning framework for stable control, which includes computation-aware dynamical model learning, computation-aware stability analysis, and computation-aware controller design, as described in Fig. 2. The contributions of this paper can be summarized as follows:

- We demonstrate that the posterior of a GP dynamical model for a control-affine system comprises two kinds of uncertainty: mathematical uncertainty and computational uncertainty, arising due to limited data and constrained computation, respectively. This is achieved by exploiting the iterative approximation of the inverse of the covariance matrix in a probabilistic manner.
- We quantify the impact of computation on the stability of learning-based systems by examining the derivative

Website: https://sites.google.com/view/computation-gp
Correspondence to wei.pan@manchester.ac.uk

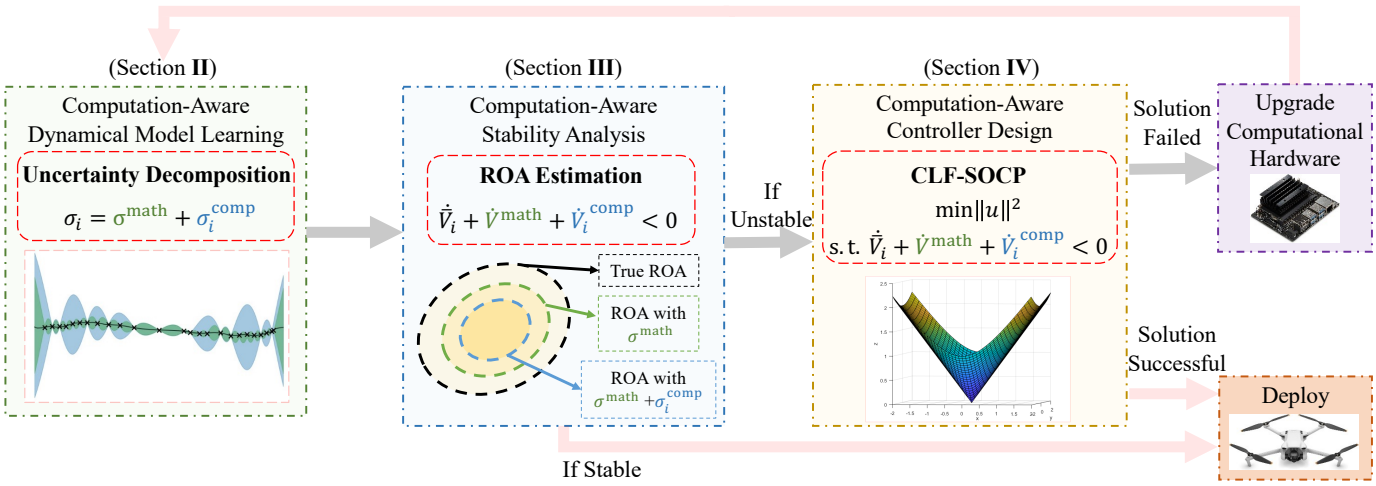

Fig. 2: Workflow of computation-aware learning for stable control. This figure illustrates the three main stages of developing a control system with computational constraints. Initially, in Section II (computation-aware dynamical model learning), the model's uncertainty is broken down into mathematical and computational parts. Following that, in Section III (computation-aware stability analysis), the system's stability is analyzed by estimating its Region of Attraction (ROA) under various uncertainties. In Section IV (computation-aware controller design), a controller is designed to ensure system stability with minimal control effort, using the control lyapunov function-second order cone programming (CLF-SOCP) method. The flowchart concludes with a decision-making process that evaluates whether the system's computational capacity needs upgrading before deployment.

of the Lyapunov function along the trajectories of the dynamical system. Based on this stability condition, the region of attraction (ROA) of the real system is estimated by discretizing the state space with Lipschitz conditions.

- We formulate a second-order cone programming (SOCP) approach to synthesize a minimum-norm stabilizing controller, taking into account both the mathematical and computational uncertainties of the learned dynamical model by leveraging the control Lyapunov function. Additionally, we demonstrate that an explicit control formula can be constructed, extending Sontag's universal formula to include both forms of uncertainty.

The remainder of this paper is organized as follows: Section II introduces computation-aware GP dynamical model learning. Section III presents the computation-aware stability analysis. Section IV describes the computation-aware controller design. The conclusions and discussions are provided in Section VII.

II. Computation-Aware Model Learning

In this section, we address the following question:

Q1: How can the computational error in dynamical model learning be quantified?

The key idea is to regard the computation of the inverse of the covariance matrix in traditional GP dynamical model learning as a probabilistic inference problem. Consider a continuous-time control affine system

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

with system state $x \in \mathcal{X} \subseteq \mathbb{R}^n$ and control input $u \in \mathcal{U} \subseteq \mathbb{R}^m$. In (1), the symbols $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ denote unknown functions that represent the drift dynamics and the input matrix, respectively. We assume that the unknown functions f and g have bounded reproducing kernel Hilbert space (RKHS) norm [24], which is a typical assumption when using GP in robotics [13, 18]:

Assumption 1 (Bounded RKHS norm)**.** *We assume that $f \in \mathcal{H}_f$ and $g \in \mathcal{H}_g$, where $\mathcal{H}_f$ and $\mathcal{H}_g$ represent the RKHSs induced by continuously differentiable and bounded kernels $k^f(x, x')$ and $k^g(x, x')$, i.e., f and g have a bounded RKHS norm with respect to continuously differentiable bounded kernels $k^f(x, x')$ and $k^g(x, x')$. Furthermore, we know corresponding upper bounds B_f and B_g, i.e., $\|f\|_{\mathcal{H}_f} \leq B_f$ and $\|g\|_{\mathcal{H}_g} \leq B_g$.*

Initially, we assume that we have access to measurements $\dot{x} = f(x) + g(x)u$, which is a common assumption for GP dynamical model learning [13, 25, 26]. In practice, the derivative of the state $\dot{x}$ could be difficult to acquire and often substituted with a discrete-time approximation [13]. One notable challenge is to design the GP prior $f(x) + g(x)u \sim \mathcal{GP}(\mu(x, u), k(x, u, x', u'))$ to differentiate the effect of the drift dynamics f and the input matrix g for the control-affine system (1). Following the structure of a control affine system, as suggested in [20, 25], we choose the GP prior mean to be the nominal model, denoted as $\mu(x, u) = \hat{f}(x) + \hat{g}(x)u$. Here, $\hat{f}(x)$ and $\hat{g}(x)$ represent the nominal parts of the drift dynamics and the control matrix, respectively, which can be modeled with relative ease. Furthermore, we select the GP kernel $k(x, u, x', u')$ as the composite GP kernel, defined as

$$k\left(x, u, x', u'\right) \triangleq k^f\left(x, x'\right) + \sum_{j=1}^{m} u_j k^{g_j}\left(x, x'\right) u'_j. \tag{2}$$

Here, $k^f, k^{g_j} : \mathbb{R}^{n\times n} \to \mathbb{R}$ are kernels that capture the behavior of the individual entries of f and g, respectively; u_j is the j_{th} dimension of the control input u, and g_j is the j_{th} column of the input matrix g.

Remark 1. *Typically, GP learning addresses single-dimensional output. For a vector-valued GP dynamical model, the common approach is to define a separate GP prior for each dimension [20]. Following the conventions of previous works [13, 26], we assume $n = 1$ and $m = 1$ to simplify the notation. In this case, the GP model $f + gu$ is a scalar function, and the kernel function becomes $k(x, u, x', u') = k^f(x, x') + uk^g(x, x')u'$.*

Remark 2. *In robotic control scenarios such as quadrotor tracking, learning a GP dynamical model online is crucial, especially when dealing with varying dynamics. For example, when quadrotors carry payloads like a bottle of water, the fluid dynamics induced by the swaying water cannot be precisely modeled and are considered disturbances, necessitating online learning. As demonstrated in our final experiment, the inputs to the GP for this water-carrying task include the quadrotor's global position, velocity and attitude, all of which are obtained through a motion capture system. The output of the GP, which serves as the label for online dynamical learning, is the disturbance force caused by fluid dynamics. This disturbance is deduced from the quadrotor's nominal model using acceleration, thrust, and the gravity force.*

For N collected measurements $Y = \big[f(x_1) + g(x_1)u_1, \ldots, f(x_N) + g(x_N)u_N\big]^\top$ at state control pairs $[X, U] = \big[(x_1, u_1), \ldots, (x_N, u_N))\big]$, the posterior mean μ_* and covariance functions k_* are given by

$$\begin{aligned}&\mu_*(x,u)\\ =&\mu(x,u) + k(x,u,X,U)K^{-1}(Y-\mu(X,U)),\end{aligned} \tag{3a}$$

$$\begin{aligned}&k_*(x,u,x',u')\\ =&k(x,u,x',u') - k(x,u,X,U)K^{-1}k(X,U,x',u').\end{aligned} \tag{3b}$$

Here, the covariance matrix K, also known as kernel Gram matrix, is computed as $K \triangleq K_f + U_{\text{diag}}^\top K_g U_{\text{diag}}$, with $[K_f]_{(p,q)} \triangleq k^f(x_p, x_q)$, $[K_g]_{(p,q)} \triangleq k^g(x_p, x_q)$ and $U_{\text{diag}} \triangleq \text{diag}\{u_1, u_2, ..., u_N\}$; $k(x, u, X, U)$ is computed as $k(x, u, X, U) \triangleq k^f(x, X) + uk^g(x, X)U_{\text{diag}}$, with $[k^f(x, X)]_{(p)} = k^f(x, x_p)$, $[k^g(x, X)]_{(p)} = k^g(x, x_p)$; and $\mu(X, U)$ is defined by $[\mu(X, U)]_{(p)} = f(x_p) + g(x_p)u_p$.

In (3), computing the inverse of K poses a significant computational burden, scaling cubically with the number of measurements. When the computation is performed exactly, i.e., we have sufficient computational resources to compute K^{-1} exactly, the modeling error is solely quantified by the quantity $k_*(x, u, x', u')$ in (3), which corresponds to the mathematical uncertainty of the learned model [27–29].

However, in real-world applications, computing $v_* = K^{-1}(Y - \mu(X, U))$ exactly is often prohibitive due to constrained computational resources, especially in online learning scenarios [1]. Therefore, the effect of constrained computation must be considered. Analogous to how limited data induces modeling error captured by mathematical uncertainty in GP learning, constrained computation introduces an approximation error that must be accounted for as computational uncertainty [28, 29]. From this perspective, the GP should return a combined uncertainty that includes both mathematical and computational aspects [28, 29].

As shown in [29], v_* can be treated as a random variable with prior distribution $v_* \sim \mathcal{N}(0, K) = \mathcal{N}(v_0, \Sigma_0)$ to include computational uncertainty. In this case, using the canonical CG method [30] to compute K^{-1} can be regarded as performing Bayesian inference on v_*, with each iteration conditioning the current belief distribution $v_* \sim \mathcal{N}(v_{i-1}, \Sigma_{i-1})$ on the linear projection of the residual $s_i^\top r_{i-1} = s_i^\top(y - \mu(X, U) - Kv_{i-1}) = s_i^\top K(v_* - v_{i-1})$. The resulting belief update at the i_{th} iteration now becomes [29]:

$$\begin{aligned}v_i &= v_{i-1} + \Sigma_{i-1}Ks_i(s_i^\top K\Sigma_{i-1}Ks_i)^{-1}s_i^\top K(v_* - v_{i-1})\\ &= C_i(y - \mu(X,U)),\\ \Sigma_i &= \Sigma_{i-1} - \Sigma_{i-1}Ks_i(s_i^\top K\Sigma_{i-1}Ks_i)^{-1}s_i^\top K\Sigma_{i-1}\\ &= K^{-1} - C_i.\end{aligned}$$

For the belief $v_* \sim \mathcal{N}(v_i, \Sigma_i)$, we have

$$p(f + gu) = \int p(f + gu|v_*)p(v_*)\,\mathrm{d}v_* = \mathcal{N}(\mu_i, k_i),$$

with

$$\begin{aligned}&\mu_i(x,u)\\ =&\mu(x,u) + k(x,u,X,U)v_i,\end{aligned} \tag{4a}$$

$$\begin{aligned}&k_i(x,u,x',u')\\ =&\underbrace{k(x,u,x',u') - k(x,u,X,U)C_ik(X,U,x',u')}_{\textbf{Combined Uncertainty}}\\ =&\underbrace{k(x,u,x',u') - k(x,u,X,U)K^{-1}k(X,U,x',u')}_{\textbf{Mathematical Uncertainty}}\\ &+\underbrace{k(x,u,X,U)\Sigma_ik(X,U,x',u')}_{\textbf{Computational Uncertainty}}\\ \triangleq&k^{\text{math}}(x,u,x',u') + k_i^{\text{comp}}(x,u,x',u').\end{aligned} \tag{4b}$$

In (4), C_i is defined as $C_i = S_i(S_i^\top \hat{K} S_i)^{-1}S_i^\top$, which is a rank-$i$ matrix. Here, S_i is given by $S_i \triangleq \begin{bmatrix}s_1 & s_2 & \cdots & s_i\end{bmatrix} \in \mathbb{R}^{N\times i}$, where each s_i represents the direction of the CG update in the i_{th} iteration. It satisfies the relation $s_i = \frac{r_{i-1}^\top Ks_{i-1}}{s_{i-1}^\top Ks_{i-1}}s_{i-1}$, where r_{i-1} is the residual at the $(i-1)_{\text{th}}$ iteration. As we perform more computations, the uncertainty about v_* contracts as $C_i \to K^{-1} = \Sigma_0$ as $i \to N$. After N iterations, $C_N = K^{-1}$, the computational uncertainty reduces to zero, and the combined uncertainty reduces to the mathematical uncertainty.

The decomposition of combined uncertainty in GPs into mathematical and computational uncertainties is already well-established in prior research [28, 29]. These studies primarily emphasize the benefits of leveraging computational uncertainty to enhance generalization for supervised learning tasks with medical datasets [28, 29]. Different from those works, this

paper focuses on robotic control. We provide a novel insight that computational uncertainty can be utilized to quantify computational errors when learning dynamical models. More importantly, in the subsequent sections, we illustrate how this computational uncertainty can further be leveraged to better estimate the region of attraction and design a stable controller in computation-constrained robotic systems.

Remark 3. *Besides CG, other methods like Cholesky decomposition [31, 32] can also be employed for GP dynamical model learning, and the computational uncertainty can similarly be quantified [28, 29].*

Leveraging the convergence result of general GP models from Corollary 1 in [29] and combining it with Assumption 1, the combined uncertainty can be utilized to establish the convergence of the posterior mean of the learned dynamical GP model:

Lemma 1 (Pointwise Convergence of the Posterior Mean)**.** *$\forall\,[x,u] \in \mathcal{X}\times\mathcal{U}$, we have $f(x)+g(x)u \in \mathcal{H}_k$ and*

$$|f(x)+g(x)u-\mu_i(x,u)| \le B_{f,g}\cdot\sqrt{k_i(x,u,x,u)}. \quad (5)$$

Here, $B_{f,g}$ is the upper bound of the RKHS norm satisfying $\|f+gu\|_{\mathcal{H}_k} \le B_{f,g}$.

The detailed proof of Lemma 1 can be found in Appendix A. Lemma 1 shows that, in the sense of point-wise convergence, the combined uncertainty k_i is the correct object characterizing the belief about the latent function $f+gu$ under constrained computation.

Remark 4. *There is an intrinsic conflict between uncertainty quantification and computational efficiency, as quantifying computational uncertainty requires additional costs. Typically, computation-aware GP model learning exhibits a quadratic time complexity of $O(N^2 i)$ for i iterations, which incurs a greater cost compared to linear-time GP approximations such as inducing point methods [33, 34]. However, in the subsequent sections, we will demonstrate that these additional costs are justifiable, as they are indispensable for providing stability guarantees in computation-constrained systems.*

III. Computation-Aware Stability Analysis

In Section II, we establish the error bound in Lemma 1 to link constrained computation with regression performance through computational uncertainty. However, this bound does not directly address the actual objective of control systems, which is achieving stable control. This and the following sections will bridge this gap. In this section, we will first address the question:

> **Q2:** How to verify the stability of the system using the computation-aware GP model?

The core idea is to examine the Lyapunov function derivative along the trajectories of the system incorporating the computation-aware GP model. First, we make a common assumption about the equilibrium point of the system [35]:

Assumption 2 (Equilibrium Point)**.** *The origin is an equilibrium point of* (1) *with $f(0)=g(0)=0$.*

Stability is crucial for robot control. The controller of a robotic system, denoted as $\pi:\mathcal{X}\to\mathcal{U}$, must stabilize around a reference point or follow a reference trajectory. Specifically, asymptotic stability implies that if the system starts 'close enough' to the equilibrium point, it remains 'close enough' indefinitely and eventually converges to it:

Definition 1 (Asymptotic Stability)**.** *System* (1) *is considered asymptotically stable if, for every $\epsilon>0$, there exists a $\delta>0$ such that $\|x(0)\|\le\delta$ implies $\|x(t)\|<\epsilon$ and $\lim_{t\to\infty}\|x(t)\|=0$ for every $t\ge 0$.*

A common way to verify stability is to use a well-defined Lyapunov function, which can be established through natural energy functions [36] or experimentation [11]:

Assumption 3 (Well-defined Lyapunov Function)**.** *A Lyapunov function $V(x)$ is fixed and two-times continuously differentiable.*

In this paper, we assume that such a function is known in advance. For asymptotic stability, for every $x\in\mathbb{R}^n\setminus 0$, the derivative of the Lyapunov function is required to be less than 0 for the closed-loop system:

$$\dot{V}(x)=\frac{\partial V}{\partial x}\left(f(x)+g(x)\pi(x)\right)<0. \quad (6)$$

However, $\dot{V}(x)<0$ is too stringent to achieve for all $x\neq 0$. In such cases, the stability of the system is typically assessed by the size of the region where $\dot{V}(x)<0$. This region is proven to be an invariant set, which is specifically referred to as the region of attraction (ROA) [35]:

Definition 2 (Region of Attraction)**.** *A set $\mathcal{R}$ is defined as a ROA of the system if, for all $x(0)\in\mathcal{R}$, it holds that $\lim_{t\to\infty}x(t)=0$.*

The concept of ROA is not only theoretically significant but also practically useful for robotic systems. It provides an essential verification of the controller's safety before its real-world deployment. For example, in drone navigation, the ROA would specify the conditions under which the drone can be expected to stabilize to a desired flight path or hover point. Finding the exact ROA analytically might be difficult or even impossible [35]. Nevertheless, we can instead use a level set of Lyapunov functions to conservatively estimate it, as shown in the following lemma:

Lemma 2 (Level Set Estimates of ROA [35])**.** *Consider a level set of the Lyapunov function $\mathcal{V}(c)=\{x\in\mathcal{X}|V(x)\le c\}$ with $c>0$. If for every $x\in\mathcal{V}(c)\setminus\{0\}$, condition* (6) *is satisfied, then $\mathcal{V}(c)$ is an invariant set and $\mathcal{V}(c)\subseteq\mathcal{R}$.*

Lemma 2 shows under which condition the level set of a Lyapunov function $\mathcal{V}(c)$ can be verified as a subset of ROA. Consequently, the largest subset is the one most closely approximating the ROA and is often used as the ROA estimate. The goal of this section is to use the computation-aware GP

model to find the largest c that satisfies the condition $\dot{V}(x) < 0$ for every $x \in \mathcal{V}(c) \setminus \{0\}$.

Using Lemma 1, for a given controller $u = \pi(x)$, we have

$$\begin{aligned}&|f(x) + g(x)\pi(x) - \mu_i(x, \pi(x))| \\ \leq& B_{f,g} \cdot \left(\sigma^{\text{math}}(x, \pi(x)) + \sigma_i^{\text{comp}}(x, \pi(x))\right),\end{aligned}$$

where $B_{f,g}$ is the upper bound of the RKHS norm satisfying $\|f + gu\|_{\mathcal{H}_k} \leq B_{f,g}$; $\sigma^{\text{math}}(x, u) \triangleq \sqrt{k^{\text{math}}(x, u, x, u)}$ and $\sigma_i^{\text{comp}}(x, u) \triangleq \sqrt{k_i^{\text{comp}}(x, u, x, u)}$ are the standard deviation of the GP posterior at (x, u). Thus, the derivative of the Lyapunov function $\dot{V}(x)$ can be bounded by:

$$\begin{aligned}\dot{V}(x) =& \frac{\partial V(x)}{\partial x}\left(f(x) + g(x)\pi(x)\right) \\ =& \frac{\partial V(x)}{\partial x}\left(\mu_i(x, \pi(x)) + f(x) + g(x)\pi(x) - \mu_i(x, \pi(x))\right) \\ \leq& \frac{\partial V(x)}{\partial x}\mu_i(x, \pi(x)) \\ &+ \left|\frac{\partial V(x)}{\partial x}\right| |f(x) + g(x)\pi(x) - \mu_i(x, \pi(x))| \\ \leq& \underbrace{\frac{\partial V(x)}{\partial x}\mu_i(x, \pi(x))}_{\triangleq \dot{\bar{V}}_i(x)} + \underbrace{\left|\frac{\partial V(x)}{\partial x}\right| \cdot B_{f,g} \cdot \sigma^{\text{math}}(x, \pi(x))}_{\triangleq \dot{V}^{\text{math}}(x)} \\ &+ \underbrace{\left|\frac{\partial V(x)}{\partial x}\right| \cdot B_{f,g} \cdot \sigma_i^{\text{comp}}(x, \pi(x))}_{\dot{V}_i^{\text{comp}}(x)}.\end{aligned}$$

From the derivation above, we find that $\dot{V}(x)$ can be upper bounded by the sum of $\dot{\bar{V}}_i(x)$, $\dot{V}_i^{\text{comp}}(x)$, and $\dot{V}^{\text{math}}(x)$. Here, $\dot{\bar{V}}_i(x)$ can be considered the mean part of the Lyapunov derivative, while $\dot{V}^{\text{math}}(x)$ relates to mathematical uncertainty, and $\dot{V}_i^{\text{comp}}(x)$ pertains to computational uncertainty. This decomposition of $\dot{V}(x)$ allows us to estimate the ROA using the computation-aware GP model:

Theorem 1 (ROA Estimation using Computation-Aware GP Model)**.** *If, for all $x \in \mathcal{V}(c) \setminus \{0\}$, the following inequality is satisfied:*

$$\dot{\bar{V}}(x) \triangleq \dot{\bar{V}}_i(x) + \dot{V}_i^{\text{comp}}(x) + \dot{V}^{\text{math}}(x) < 0, \tag{7}$$

then it follows that $\mathcal{V}(c) \subseteq \mathcal{R}$.

Proof: Note that (7) represents a sufficient condition for (6). Therefore, this theorem can be directly derived from Lemma 2. ∎

This theorem offers a sufficient condition for estimating the ROA using a computation-aware GP model. The decomposition of $\dot{V}(x)$ indicates that both mathematical and computational uncertainties are indispensable in estimating the ROA. With a fixed amount of collected data, the mathematical uncertainty part, $\dot{V}^{\text{math}}(x)$, remains constant, while the computational uncertainty part, $\dot{V}_i^{\text{comp}}(x)$, typically decreases as more computations are performed [28, 29].

Remark 5. *Previous ROA estimation methods that use GP dynamic model learning, such as those described in [13], do not account for computational uncertainty. This oversight can lead to estimated ROAs that lack stability guarantees, meaning the estimated level sets of the Lyapunov function cannot assure they are subsets of the ROA under constrained computation. For instance, it has been observed that the ROAs estimated using the method in [13] exceed the actual ROAs in scenarios with computational constraints, as demonstrated in Fig. 5.*

In practice, it is often impossible to evaluate $\dot{V}(x)$ everywhere in a continuous domain. Nonetheless, the continuity of $\dot{V}$ allows for evaluating it only at a finite number of points without losing guarantees, as shown in the subsequent corollary:

Corollary 1 (ROA Estimation Through Discretization)**.** *Assume that the given policy π is bounded in $\mathcal{X}$ satisfying $\|\pi\|_\infty \leq B_\pi$. If, for every $x \in \{\mathcal{X}_\tau \cap \{\mathcal{V}(c) \setminus 0\}\}$, the following inequality is satisfied:*

$$\dot{\bar{V}}_i(x) + \dot{V}^{\text{math}}(x) + \dot{V}_i^{\text{comp}}(x) < -L\tau, \tag{8}$$

where L is the Lipschitz constant of $\dot{V}(x)$ defined by

$$\begin{aligned}L =& \left(B_f^2\|k^f\|_\infty + B_g^2 B_\pi \|k^g\|_\infty\right) \cdot \left\|\frac{\partial^2 E(x)}{\partial x^2}\right\|_\infty \\ &+\sqrt{2}\left\|\frac{\partial E(x)}{\partial x}\right\|_\infty \cdot \left(B_f\sqrt{\|k^f\|_\infty \left\|\frac{\partial k^f}{\partial x}\right\|_\infty}\right) \\ &+\sqrt{2}\left\|\frac{\partial E(x)}{\partial x}\right\|_\infty \cdot \left(B_g\pi_g\sqrt{\|k^g\|_\infty \left\|\frac{\partial k^g}{\partial x}\right\|_\infty}\right).\end{aligned}$$

It then follows that $\mathcal{V}(c) \subseteq \mathcal{R}$. Here, $\mathcal{X}_\tau \in \mathcal{X}$ is a discretization of $\mathcal{X}$ satisfying $|x - [x]_\tau| \leq \frac{\tau}{2}$ for all $x \in \mathcal{X}$; $[x]_\tau$ defines the nearest point in $\mathcal{X}_\tau$ to $x \in \mathcal{X}$.

The detailed proof of Corollary 1 can be found in Appendix B. The Lipschitz continuity of $\dot{V}(x)$ allows for the extension of results from $\mathcal{X}_\tau$ to the broader set $\mathcal{X}$. Additionally, the trade-off between accuracy and computational effort can be managed by adjusting the discretization factor τ. A smaller value of τ leads to a less conservative estimate of the ROA.

This corollary provides a way for estimating the ROA by means of binary search. For high-dimensional systems, this approach may become computationally demanding due to the exponential increase in the number of points to be checked with each added dimension. Nevertheless, the primary purpose of quantifying the ROA is to evaluate the stability of a given controller before deployment. Therefore, it is typically performed offline, and the required time could be manageable for offline computation.

IV. Computation-Aware Controller Design

Section III introduces the method to verify the stability of a given controller. If a given controller does not meet the desired ROA, it needs to be redesigned using the computation-aware GP model. This section answers the following question:

Q3 How can a stable controller be synthesized for a computation-constrained system?

The core idea involves formulating an optimization problem that minimizes the control input norm, subject to the inequality constraints $\dot{V} < 0$, by employing a computation-aware GP model. We begin with the definition of the control Lyapunov function (CLF):

Definition 3 (Control Lyapunov Function)**.** *The function $V(x)$ is a control Lyapunov function for the system* (1) *if, for each $x \in \mathbb{R}^n \setminus 0$, it holds that*

$$\inf_{u \in \mathbb{R}^m} \underbrace{\frac{\partial V(x)}{\partial x} f(x) + \frac{\partial V(x)}{\partial x} g(x) u}_{\triangleq \dot{V}(x,u)} < 0.$$

The existence of such a CLF indicates that the system is globally stabilizable if V is radially unbounded, i.e., $V(x) \to \infty$ as $|x| \to \infty$ [37]. The objective then becomes to find a local Lipschitz continuous feedback control law $u = \pi(x)$, ensuring that the condition $\dot{V}(x, \pi(x)) < 0$ is met for all $x \in \mathbb{R}^n \setminus 0$. For unconstrained inputs $u \in \mathbb{R}^m$, such a feedback control law can be readily derived in a closed form [23]. However, robotic systems typically face actuator limitations that manifest as control constraints $u \in \mathcal{U}$. Here, the set $\mathcal{U} \subseteq \mathbb{R}^m$ represents the compact set of all admissible control inputs. In this case, there is no exact closed-form to construct a stable control policy and it may be unfeasible to find a control input ensuring $\dot{V}(x, u) < 0$ for every $x \in \mathbb{R}^n \setminus 0$, even if V is a valid CLF for the system. In such cases, it is possible to identify a level set of the Lyapunov function $\mathcal{V}(c) = \{x \in \mathcal{X} | V(x) \leq c\}$, where

$$\inf_{u \in \mathcal{U}} \frac{\partial V(x)}{\partial x} f(x) + \frac{\partial V(x)}{\partial x} g(x) u < 0,$$

is valid for all $x \in \mathcal{V}(c) \setminus \{0\}$. To facilitate the convexification of the optimization problem, and also impose a stronger notion of stabilizability for exponential convergence [20, 38, 39], we can instead consider:

$$\inf_{u \in \mathcal{U}} \frac{\partial V(x)}{\partial x} f(x) + \frac{\partial V(x)}{\partial x} g(x) u + \lambda \cdot V(x) \leq 0, \quad (9)$$

with $\lambda > 0$. A straightforward method to synthesize such a control law is by enforcing (9) as a constraint in a min-norm optimization problem:

CLF-QP:

$$\pi^*(x) = \quad \arg\min_{u \in \mathcal{U}} \quad \|u\|^2, \quad (10a)$$

$$\text{s.t.} \quad \frac{\partial V(x)}{\partial x} f(x) + \frac{\partial V(x)}{\partial x} g(x) u + \lambda \cdot V(x) \leq 0. \quad (10b)$$

In this paper, we assume that the input constraints are linear, which renders problem (10) a quadratic program (QP). This optimization problem formulates a feedback control law, $\pi^*(x)$, that selects the min-norm input to ensure exponential convergence of the system state to the origin. However, this QP is not always feasible as the actual model inevitably contains uncertainties. Therefore, a principled solution involves reformulating the min-norm stabilizing controller, as defined in (10), to accommodate model uncertainty. Following (7) from Theorem 1, a sufficient condition for (10b) is given by:

$$\begin{aligned} &\frac{\partial V(x)}{\partial x} \cdot \mu_i(x, u) + B_{f,g} \cdot \left| \frac{\partial V(x)}{\partial x} \right| \cdot \sigma_i^{\text{comp}}(x, u) \\ &+ B_{f,g} \cdot \left| \frac{\partial V(x)}{\partial x} \right| \cdot \sigma^{\text{math}}(x, u) + \lambda V(x) \leq 0. \end{aligned} \quad (11)$$

Therefore, we can replace the constraint in (10b) with its sufficient condition (11):

Computation-Aware GP-CLF-SOCP:

$$\begin{aligned} \pi^*(x) = \quad & \arg\min_{u \in \mathcal{U}} \quad \|u\|^2 \\ \text{s.t.} \quad & (11). \end{aligned} \quad (12)$$

The optimization problem (12) is classified as a second-order cone program (SOCP). This arises from the inherent characteristics of the composite kernels. Specifically, μ_i is a linear function of u, while $\left(\sigma_i^{\text{comp}}\right)^2$ and $\left(\sigma^{\text{math}}\right)^2$ are both positive definite quadratic functions of u, as elucidated in [20].

Theorem 2. *If there exists a constant $c > 0$ such that the solution to the optimization problem* (12) *exists for all $x \in \mathcal{V}(c) \setminus 0$, then $\mathcal{V}(c)$ is a subset of ROA.*

Proof: The proof of ROA is straightforward: for all x at the boundary of $\mathcal{V}(c)$, we can always find a control input u such that $\dot{V}(x, u) < 0$ ensured by (11). ■

Theorem 2 ensures the stability of the controller synthesized through solving (12). We want to emphasize that the constraints in the optimization problem often neglect computation uncertainty [20], i.e., $\sigma_i^{\text{comp}}(x, u) \equiv 0$ is implicitly assumed. Under this assumption, (11) no longer serves as a sufficient condition for (10b).

Remark 6. *In practice, the constraint* (11) *can be relaxed by introducing a slack variable, ensuring the feasibility of the problem in cases where it is not locally satisfied. Additionally, instead of minimizing the norm of the control signals, we often minimize their weighted norm to normalize the impact of the magnitude of each control quantity. Consequently, the optimization problem in* (11) *can be reformulated as*

$$\begin{aligned} \pi^*(x) = \quad & \arg\min_{u \in \mathcal{U},\ d \in \mathbb{R}} \quad \|u\|_W^2 + p\, d^2 \\ \text{s.t.} \quad & \frac{\partial V(x)}{\partial x} \cdot \mu_i(x, u) + B_{f,g} \cdot \left| \frac{\partial V(x)}{\partial x} \right| \cdot \sigma_i^{\text{comp}}(x, u) \\ & + B_{f,g} \cdot \left| \frac{\partial V(x)}{\partial x} \right| \cdot \sigma^{\text{math}}(x, u) + \lambda V(x) \leq d. \end{aligned}$$

Here, $\|u\|_W^2 \triangleq u^\top W u$.

In a specific scenario where only the drift dynamics $f(x)$ is unknown, and the control matrix $g(x)$ is known, we need

to learn only $f(x)$. In this context, the posterior mean for learning $f(x)$ is denoted as $\mu_i(x)$, while the uncertainties are expressed as $\sigma_i^{\text{comp}}(x)$ and $\sigma^{\text{math}}(x)$, respectively. Under mild assumptions, it is possible to directly construct an explicit form of a stable controller for unconstrained inputs without the necessity of solving an optimization problem:

$$\pi^{\text{explicit}}(x) = -\frac{a(x) + \sqrt{a^2(x) + \|b(x)\|^4}}{\|b(x)\|^2} b(x). \tag{13}$$

Here, $a(x)$ is a scalar function defined as

$$a(x) \triangleq \frac{\partial V(x)}{\partial x}\mu_i(x) + B_f \cdot \left|\frac{\partial V(x)}{\partial x}\right| \cdot \left(\sigma_i^{\text{comp}}(x) + \sigma^{\text{math}}(x)\right), \tag{14}$$

and $b(x)$ is a vector function defined by

$$b(x) \triangleq g^\top(x)\frac{\partial V(x)}{\partial x^\top}. \tag{15}$$

The stability of this explicit control policy is summarized in the subsequent Proposition:

Proposition 1 (Explict Form of Stable Control Policies)**.** *Assuming that the input matrix $g(x) \neq 0$ and $\frac{\partial V(x)}{\partial x} \neq 0$ for all x. Then, the explicit control policy in* (13) *stabilizes the closed-loop system.*

Proof: Defining $a(x)$ and $b(x)$ as in (14) and (15), we have

$$\begin{aligned}
&\frac{\partial V(x)}{\partial x}\mu_i(x) + B_f \cdot \left|\frac{\partial V(x)}{\partial x}\right| \cdot \left(\sigma_i^{\text{comp}}(x) + \sigma^{\text{math}}(x)\right) \\
&+\frac{\partial V(x)}{\partial x} g(x)\pi^{\text{explicit}}(x) \\
=&a(x) + b^\top(x)\left(-\frac{a(x) + \sqrt{a^2(x) + \|b(x)\|^4}}{\|b(x)\|^2} b(x)\right) \\
=&-\sqrt{a^2(x) + \|b(x)\|^4} < 0.
\end{aligned}$$

■

The control policy described in (13) can be viewed as an extension of Sontag's universal formula [40]. The key difference lies in our theorem's additional consideration of the model's mathematical and computational uncertainties. If the model learning is sufficiently accurate, for example, using sufficient data and ample computation, our approach reduces to Sontag's universal formula.

Remark 7. *We have rigorously proven that by using a computation-aware GP model, we can synthesize a stable controller in computation-constrained systems. However, there are certain trade-offs involved. Intuitively, when computational uncertainty is present, the complexity of model learning increases, as noted in remark 4. However, this increase in complexity is usually not critical and can often be disregarded [29]. Meanwhile, compared to approaches that do not account for computational uncertainty, the constraints in solving the min-norm optimization problem are tighter, often leading to larger control signals.*

V. Simulations

In this section, we consider two simulations of the canonical control tasks to show the effectiveness of computation-aware learning, computation-aware stability analysis, and computation-aware controller design.

A. Nonlinear 1D System

Consider a canonical 1D system [13] described by

$$\dot{x} = f(x) + u, \tag{16}$$

where $x \in \mathbb{R}$ is the system state, and the system dynamics $f(x)$ is sampled from a GP with zero mean and a composite kernel, i.e., $f \sim \mathcal{GP}(0, k)$. To achieve a reasonably good performance, the kernel k is chosen as a product of a linear kernel and a Matérn kernel, defined as $k(x, x') \triangleq k_{\text{linear}}(x, x') * k_{\text{Matérn}}(x, x')$ [13].

As demonstrated in Fig. 3, we decompose the uncertainty of the learned GP model into two types: mathematical uncertainty and computational uncertainty. The mathematical uncertainty of the GP dynamical model remains constant since the number of acquired data points is fixed. In contrast, the computational uncertainty decreases as the number of iterations increases. Notably, when the number of iterations equals the number of data points, we achieve ideal computation, and the computational uncertainty is reduced to zero. Furthermore, the true model is encompassed within the error bar of the combined uncertainty but exceeds that of the mathematical uncertainty alone. This indicates that the combined uncertainty provides a tight worst-case bound on the relative error between the posterior mean of the GP dynamical model and the true system. It also suggests that relying solely on mathematical uncertainty is inadequate for effective model error quantification.

Building on this computation-aware GP model, we quantify the ROA using Theorem 1, adhering to the fixed feedback control policy $\pi(x) = -2.5x$, as established in [13]. As shown in Fig. 3, we observe that the ROA estimate increases as the computational uncertainty decreases, but never exceeds the true ROA, which is within the interval $[-0.921, 0.921]$. This observation implies that our ROA estimates are conservative approximations of the actual ROA. Additionally, we note that the ROA does not change linearly or uniformly with the number of iterations. For example, in this simple one-dimensional system, the ROA determined with 12 iterations is the interval $[-0.915, 0.915]$, and for 25 iterations, it becomes $[-0.918, 0.918]$. This suggests that a computation with 12 iterations is sufficient to accurately estimate the ROA.

Finally, we implement our proposed explicit control policy (13) to construct a stable controller using the GP model acquired by 4, 12, and 25 CG iterations. For comparison, we use a similar control policy form but assume that the computational uncertainty is zero. To better evaluate their stability, we directly depict the true ROA using the real system model instead of estimating it with the GP model. Fig. 3 clearly shows that the computation-aware controller achieves stability throughout the interval of $[-1, 1]$. On the contrary, the controller without computational awareness leads to a much

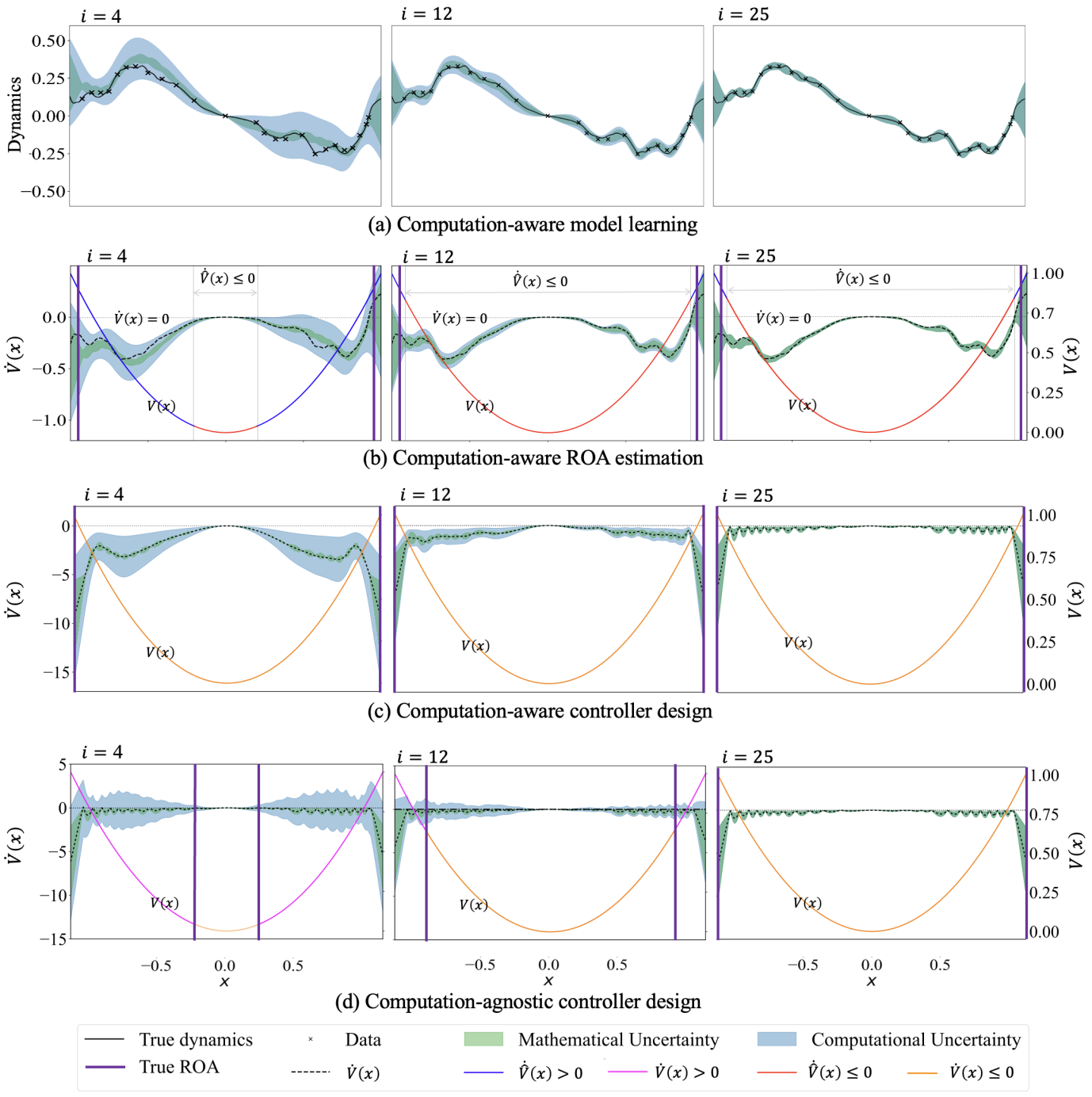

Fig. 3: Comprehensive evaluation of model learning and stable control in a nonlinear 1D System (16). During the GP model training, we systematically gather 25 data points per online update and vary the CG iterations for model learning ($i = 4$, $i = 12$, $i = 25$). The combined uncertainty of the learned dynamics, Lyapunov function $V(x)$ and its derivative $\dot{V}(x)$ in (6) respectively, decompose into the mathematical uncertainty (■) and computational uncertainty (■). (a) Computation-aware model learning. The computational uncertainty diminishes with increasing i, while mathematical uncertainty remains constant. (b) Computation-aware ROA estimation. The estimated ROA grows with each iteration, converging towards the true ROA using a given control policy $\pi(x) = -2.5x$. (c) Computation-aware controller design. Our method (12) always ensures a negative $\dot{V}(x)$, signifying stable control, which is impressive even when the model is learned with only 4 iterations. (d) Computation-agnostic controller design. This design (setting $\sigma_i^{\text{comp}} \equiv 0$ in (12)), which omits computational uncertainty, yields a much smaller ROA compared to the counterparts in (c), potentially leading to unsafety in regions where $\dot{V}(x)$ is positive.

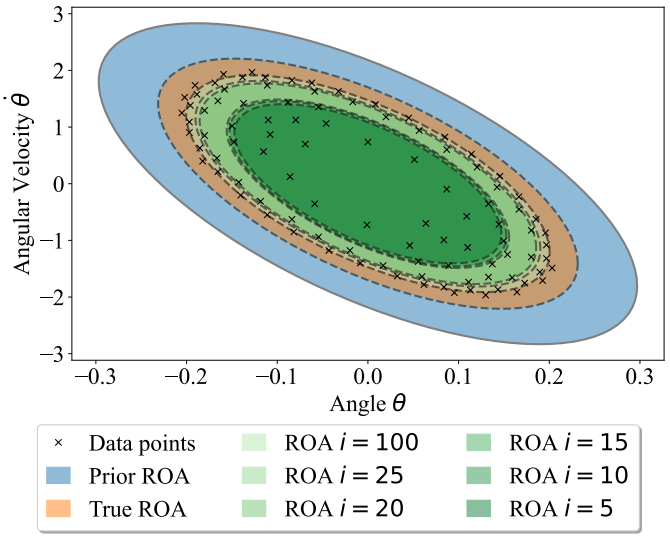

Fig. 4: Computation-aware ROA estimation for inverted pendulum system given the linear quadratic controller. The ROA estimate using the prior model exceeds the true ROA. In contrast, the ROA estimates using computation-aware GP models do not exceed the true ROA and increase with the number of CG iterations.

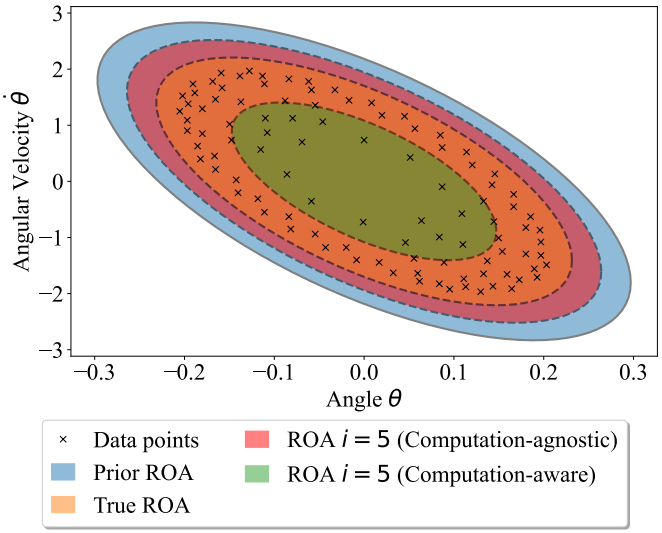

Fig. 5: Comparison of computation-aware (Our method) and computation-agnostic [13] ROA estimation for inverted pendulum system under computation constraints. Neglecting computational uncertainty can lead to overly optimistic estimates of a system's stability, potentially misclassifying unstable regions (■) as stable.

smaller ROA. For example, in the case of 4 CG iterations, the system is stable only within the interval of $[-0.2, 0.2]$. This implies that computational uncertainty is indispensable for designing a stable controller.

B. Inverted Pendulum

We further investigate the ROA estimation using the canonical inverted pendulum system [13]:

$$\begin{bmatrix}\dot{\theta}(t)\\ \ddot{\theta}(t)\end{bmatrix} = \begin{bmatrix}\dot{\theta}(t)\\ \frac{mgl\sin(\theta(t))-\mu\dot{\theta}(t)}{ml^2}\end{bmatrix} + \begin{bmatrix}0\\ \frac{1}{ml^2}\end{bmatrix} u(t). \tag{17}$$

Here, θ represents the angle, $m = 0.15\text{kg}$ is the mass, $l = 0.5\text{m}$ is the length, and $\mu = 0.05$ is the friction coefficient, which are consistent with [13].

We assume that our knowledge is limited to a linear approximation of the dynamics (17) at the upright equilibrium point. In this approximation, we neglect friction and consider the mass to be 0.05kg lighter. This linearized model serves as the nominal model and, thus, as the prior model for GP model learning. For estimating the ROA, we employ a linear quadratic regulator (LQR) based on the nominal model as the baseline controller, which uses the same weighted matrices as in [13]. In addition, the level set of its quadratic Lyapunov function is leveraged to estimate the ROA.

In this study, we collect 100 data points and present the ROA estimates for various numbers of CG iterations. As shown in Fig. 4, the ROA estimate using the prior model is excessively large, exceeding the true value. In contrast, the ROA estimates using computation-aware GP models are more conservative. These estimates do not exceed the true ROA and increase with the number of CG iterations. Furthermore, to demonstrate the critical role of computational uncertainty in ROA estimates for computation-constrained systems, we compare our approach with a computation-agnostic method [13], which overlooks computational uncertainty in estimating ROA. From Fig. 5, we observe that for 5 CG iterations, the ROA estimate of the computation-agnostic method can exceed the true value. This suggests that neglecting computational uncertainty can lead to overly optimistic estimates of a system's stability, potentially misclassifying unstable regions as stable.

VI. Experiment on Quadrotor Tracking

In this section, we present an experiment on a quadrotor tracking task with a payload to evaluate the effectiveness of our proposed framework. The experimental setup is composed of a motion capture system with 6 cameras, a WiFi router for communication and a customized quadrotor with a flight controller Pixhawk 2.4.8 and an onboard Linux computer (Jetson Orin NX 16GB RAM). We retrofitted the quadrotor with 4 reflective infrared markers to acquire accurate position and attitude using the motion capture system, see Fig. 6. Additionally, the accurate velocity and acceleration of the quadrotor can be obtained from the positional differences.

Considering the system dynamics of a quadrotor model [41]:

$$\dot{p} = v, \qquad m\dot{v} = mg_v + Rf_u + f_d, \tag{18a}$$
$$\dot{R} = RS(\omega), \qquad J\dot{\omega} = J\omega \times \omega + \tau_u + \tau_d, \tag{18b}$$

Here, (18a) and (18b) describe the position and attitude dynamics of the quadrotor, respectively. The variables $p \triangleq [p_x, p_y, p_z]^\top \in \mathbb{R}^3$, $v \in \mathbb{R}^3$, $R \in \text{SO}(3)$ and $\omega \in \mathbb{R}^3$ represent the global position, velocity, attitude rotation matrix, and body angular velocity, respectively. The symbols m and J

Fig. 6: Customized quadrotor equipped with a Pixhawk 2.4.8 flight controller and a Jetson Orin NX onboard computer, carrying a bottle of water.

denote the mass and inertia matrix, while S denotes the skew-symmetric mapping. $g_v \triangleq [0,\, 0,\, -g]^\top$ is the gravity vector; $f_u = [0,\, 0,\, T]^\top$ and $\tau_u = [\tau_x,\, \tau_y,\, \tau_z]^\top$ are the total thrust and body torques.

In our experiment, we consider the task where a quadrotor carries a bottle of water and attempts to follow a single continuous trajectory, as shown in Fig. 1. Here, f_d and τ_d represent the unknown disturbance forces and disturbance torques, respectively. These disturbances are originated from the complex slosh dynamics [42] and the asymmetric centroid caused by the swaying water in the bottle, which are difficult to model accurately. To address the disturbance forces f_d, we use GP to learn them online. Following [41], we employ a different, highly robust controller to manage the attitude dynamics, thus the disturbance torque τ_d is not our concern. The frequency of the position control loop (18a) is set to 10 Hz to accommodate online GP learning and controller synthesis, whereas the frequency of the attitude control loop (18b) is 250 Hz, which is the default setting in the Pixhawk flight controller.

To fully predict the disturbance forces f_d, we use position p, velocity v and rotation R as inputs to the GP model. Additionally, we employ the relationship $f_d = m\dot{v} - mg_v - Rf_u$ to calculate f_d, which serves as the measurement for the GP model. For every set of 20 data points, we perform online learning for both the computation-aware and computation-agnostic GP models using $i = 5$, $i = 10$, and $i = 15$ CG iterations, respectively. For trajectory tracking, we define a sliding variable s, which is a manifold on which the tracking error $\tilde{p} \triangleq p - p_d$ converges to zero exponentially:

$$s \triangleq \dot{\tilde{p}} + \Lambda\tilde{p} = \dot{p} - \dot{p}_d + \Lambda\tilde{p},$$

where Λ is a positive definite diagonal matrix and p_d is the desired trajectory. Following [41], we use the sliding variable to construct the Lyapunov function $V(s) = \frac{1}{2}m\|s\|^2$. Taking the time derivative on $V(s)$, we have

$$\begin{aligned}\dot{V}(s) &= ms^\top \dot{s} \\ &= ms^\top \left(\ddot{p} - \ddot{p}_d + \Lambda\dot{\tilde{p}}\right) \\ &= s^\top \left(mg_v + Rf_u + f_d - m\ddot{p}_d + \Lambda m\dot{\tilde{p}}\right).\end{aligned}$$

As shown in Section III, $\dot{V}(s)$ can be bounded by the sum

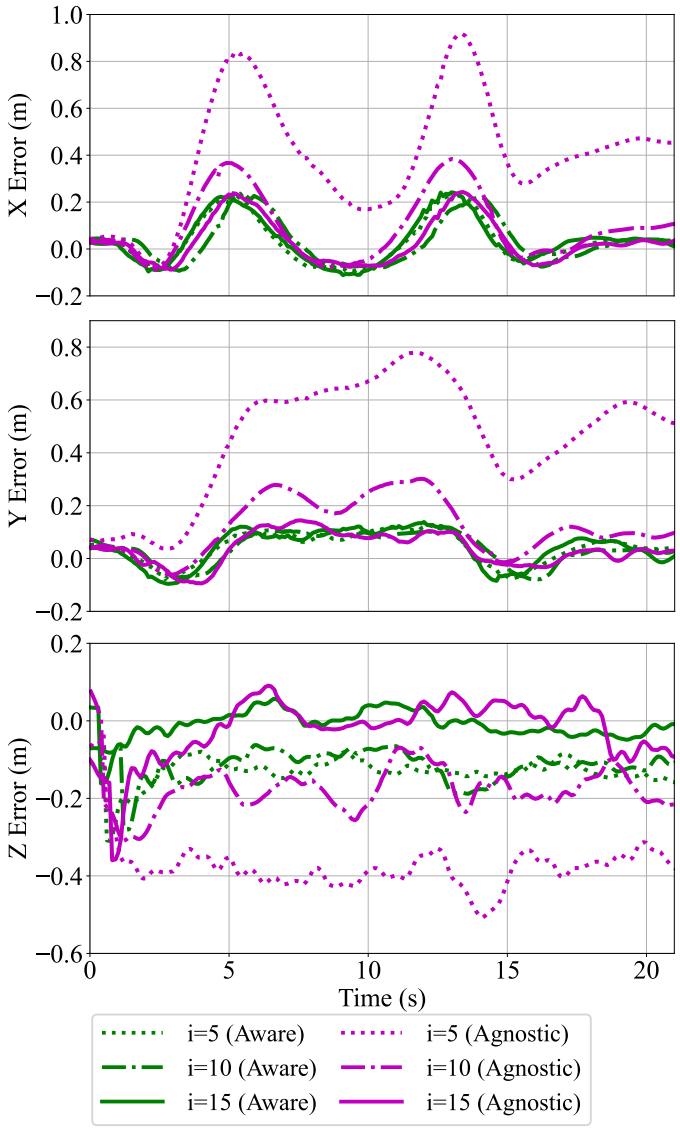

Fig. 7: Comparsion of the tracking error for computation-aware and computation-agnostic controllers.

of three terms $\dot{\tilde{V}}_i$, $\dot{V}_i^{\text{comp}}$ and $\dot{V}^{\text{math}}$, which are defined as

$$\begin{aligned}\dot{\tilde{V}}_i(s) &= s^\top \left(mg_v + Rf_u + \hat{f}_d - m\ddot{p}_d + \Lambda m\dot{\tilde{p}}\right), \\ \dot{V}^{\text{math}}(s) &= |s|^\top \cdot B_f \cdot \sigma^{\text{math}}, \\ \dot{V}_i^{\text{comp}}(s) &= |s|^\top \cdot B_f \cdot \sigma_i^{\text{comp}},\end{aligned}$$

where $\hat{f}_d$, σ^{math}, and σ_i^{comp} denote the posterior mean and the mathematical and computational uncertainties of the learned GP model of f_d, respectively. Additionally, B_f is defined as the upper bound of the RKHS norm according to Lemma 1. To avoid solving the SOCP problem, we utilize Proposition 1 to construct an explicit control policy. For comparison purposes, we also synthesize another explicit form by setting σ_i^{comp} to zero, referring to the resulting controller as a computation-agnostic controller.

In Fig. 7, we plot the tracking error for both

TABLE I: Root Mean Square Error of Position Tracking

	Computation-Aware			Computation-Agnostic		
	$i = 5$	$i = 10$	$i = 15$	$i = 5$	$i = 10$	$i = 15$
$\tilde{p}_x$ (m)	0.0915	0.0911	0.0977	0.467	0.161	0.0949
$\tilde{p}_y$ (m)	0.0664	0.0655	0.0752	0.509	0.154	0.0685
$\tilde{p}_z$ (m)	0.142	0.122	0.0387	0.387	0.197	0.0765

TABLE II: Average Computation Time for Each Control Loop

	Computation-Aware			Computation-Agnostic		
	$i = 5$	$i = 10$	$i = 15$	$i = 5$	$i = 10$	$i = 15$
GP (s)	0.0155	0.0327	0.0452	0.0152	0.0311	0.0434

the computation-aware and computation-agnostic controllers across different CG iterations. As shown in Fig. 7, as the number of CG iterations decreases, the tracking error for the computation-agnostic controller increases significantly, indicating that constrained computation can indeed negatively impact tracking performance. However, the degradation is slight for our proposed computation-aware controller, and even 5 CG iterations achieve better performance in the X, Y, and Z directions than the 10 CG iteration case for the computation-agnostic controller.

Interestingly, for the computation-aware controller, the position error in the X and Y directions with 5 and 10 CG iterations can even be better than with 15 CG iterations. This may be due to two reasons: First, slosh dynamics primarily cause effects akin to changes in the quadrotor's gravity, thus greatly affecting the Z direction; Second, as we have mentioned, our attitude dynamics employ a highly robust controller that is distinct from the position dynamics, potentially compensating for inaccuracies in the position dynamics control loop. However, it is important to note that for the computation-agnostic scenario, due to the lack of consideration for computational uncertainty, the X and Y errors still increase greatly as the number of CG iterations decreases.

The average computation time for each control loop for GP dynamical model learning is presented in TABLE II. As shown in TABLE II, the computation time for dynamical GP learning increases with the number of CG iterations. For a fixed i, the computation-aware GP exhibits a marginal increase in computation time as it additionally needs to quantify computational uncertainty, compared to the computation-agnostic counterpart. However, given its improvement in tracking performance (with a reduction in tracking error on the X, Y, and Z axes by approximately $0.37m$, $0.44m$, and $0.24m$ respectively for $i = 5$), the trade-off is worthwhile.

VII. Conclusions and Discussions

Conclusion: In this paper, we propose a computation-aware framework for GP dynamical model learning to ensure stable control in robotic systems subjected to computational constraints. We thoroughly investigate the impacts of constrained computations on model learning errors by utilizing Gaussian processes. We find that computational errors in model learning are inevitable and can lead to deterioration in control performance. Subsequently, we quantify the consequences of these computational errors on system stability by evaluating the region of attraction. Finally, we present a novel, robust controller design methodology that incorporates these computational considerations using second-order cone programming. The effectiveness of the framework is demonstrated through several canonical control tasks, elucidating its potential to enhance control performance under computational constraints.

Discussion: It's crucial to emphasize that our method extends beyond stable control in robotics. Issues such as safety and passivity in robotic systems, which are characterized by energy-like functions, can also be addressed in a similar manner. A more detailed explanation can be found in Appendix C. Besides, the primary idea behind integrating computational uncertainty into GP is the probabilistic approximation of the inverse of the Gram matrix [28, 29]. Although this idea is deeply rooted in the architecture of GPs, the innovative concept of computational uncertainty may offer potential for extension to other probabilistic models, including Bayesian neural networks [43] and deep Gaussian processes [44].

Acknowledgements

This work was supported by the Consolidator Grant "Safe data-driven control for human-centric systems" (CO-MAN) of the European Research Council (ERC) under grant agreement IDs 864686. We would also like to express our gratitude to Ribhav Ojha and Samuel Belkadi, undergraduate students at the University of Manchester, for their invaluable assistance in building and assembling the experimental hardware.

References

[1] D. Nguyen-Tuong and J. Peters, "Local gaussian process regression for real-time model-based robot control," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 380–385.

[2] M. P. Deisenroth, D. Fox, and C. E. Rasmussen, "Gaussian processes for data-efficient learning in robotics and control," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 2, pp. 408–423, 2013.

[3] D. Jang, J. Yoo, C. Y. Son, D. Kim, and H. J. Kim, "Multi-robot active sensing and environmental model learning with distributed gaussian process," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5905–5912, 2020.

[4] A. Haddadi and K. Hashtrudi-Zaad, "Online contact impedance identification for robotic systems," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 974–980.

[5] D. Nguyen-Tuong, M. Seeger, and J. Peters, "Real-time local gp model learning," *From Motor Learning to Interaction Learning in Robots*, pp. 193–207, 2010.

[6] G. Fang, X. Wang, K. Wang, K.-H. Lee, J. D. Ho, H.-C. Fu, D. K. C. Fu, and K.-W. Kwok, "Vision-based online learning kinematic control for soft robots using

local gaussian process regression," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1194–1201, 2019.

[7] B. Wilcox and M. C. Yip, "Solar-gp: Sparse online locally adaptive regression using gaussian processes for bayesian robot model learning and control," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2832–2839, 2020.

[8] J. T. Wen, "A unified perspective on robot control: The energy lyapunov function approach," in *29th IEEE Conference on Decision and Control*. IEEE, 1990, pp. 1968–1973.

[9] V. Santibanez and R. Kelly, "Strict lyapunov functions for control of robot manipulators," *Automatica*, vol. 33, no. 4, pp. 675–682, 1997.

[10] S. M. Khansari-Zadeh and A. Billard, "Learning control lyapunov function to ensure stability of dynamical system-based robot reaching motions," *Robotics and Autonomous Systems*, vol. 62, no. 6, pp. 752–765, 2014.

[11] P. Giesl and S. Hafstein, "Review on computational methods for lyapunov functions," *Discrete and Continuous Dynamical Systems-B*, vol. 20, no. 8, pp. 2291–2331, 2015.

[12] Q. Nguyen and K. Sreenath, "Optimal robust control for bipedal robots through control lyapunov function based quadratic programs." in *Robotics: Science and Systems*, vol. 11. Rome, Italy, 2015.

[13] F. Berkenkamp, R. Moriconi, A. P. Schoellig, and A. Krause, "Safe learning of regions of attraction for uncertain, nonlinear systems with gaussian processes," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 4661–4666.

[14] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin, "A general safety framework for learning-based control in uncertain robotic systems," *IEEE Transactions on Automatic Control*, vol. 64, no. 7, pp. 2737–2752, 2018.

[15] 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.

[16] J. Umlauft, L. Pöhler, and S. Hirche, "An uncertainty-based control lyapunov approach for control-affine systems modeled by gaussian process," *IEEE Control Systems Letters*, vol. 2, no. 3, pp. 483–488, 2018.

[17] A. Lederer, J. Umlauft, and S. Hirche, "Uniform error bounds for gaussian process regression with application to safe control," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[18] T. Beckers, D. Kulić, and S. Hirche, "Stable gaussian process based tracking control of euler–lagrange systems," *Automatica*, vol. 103, pp. 390–397, 2019.

[19] M. Khan, T. Ibuki, and A. Chatterjee, "Safety uncertainty in control barrier functions using gaussian processes," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 6003–6009.

[20] F. Castaneda, J. J. Choi, B. Zhang, C. J. Tomlin, and K. Sreenath, "Gaussian process-based min-norm stabilizing controller for control-affine systems with uncertain input effects and dynamics," in *2021 American Control Conference (ACC)*. IEEE, 2021, pp. 3683–3690.

[21] S. M. Neuman, B. Plancher, B. P. Duisterhof, S. Krishnan, C. Banbury, M. Mazumder, S. Prakash, J. Jabbour, A. Faust, G. C. de Croon *et al.*, "Tiny robot learning: challenges and directions for machine learning in resource-constrained robots," in *2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE, 2022, pp. 296–299.

[22] X. Zhou, J. Yang, M. Chrobak, and Y. Zhang, "Performance-aware thermal management via task scheduling," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 7, no. 1, pp. 1–31, 2010.

[23] E. D. Sontag, "A 'universal'construction of artstein's theorem on nonlinear stabilization," *Systems & control letters*, vol. 13, no. 2, pp. 117–123, 1989.

[24] A. Berlinet and C. Thomas-Agnan, *Reproducing kernel Hilbert spaces in probability and statistics*. Springer Science & Business Media, 2011.

[25] J. Umlauft and S. Hirche, "Feedback linearization based on gaussian processes with event-triggered online learning," *IEEE Transactions on Automatic Control*, vol. 65, no. 10, pp. 4154–4169, 2019.

[26] A. Lederer, A. Capone, J. Umlauft, and S. Hirche, "How training data impacts performance in learning-based control," *IEEE Control Systems Letters*, vol. 5, no. 3, pp. 905–910, 2020.

[27] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006, vol. 2.

[28] P. Hennig, M. A. Osborne, and H. P. Kersting, *Probabilistic Numerics: Computation as Machine Learning*. Cambridge University Press, 2022.

[29] J. Wenger, G. Pleiss, M. Pförtner, P. Hennig, and J. P. Cunningham, "Posterior and computational uncertainty in gaussian processes," *Advances in Neural Information Processing Systems*, vol. 35, pp. 10 876–10 890, 2022.

[30] M. R. Hestenes, E. Stiefel *et al.*, "Methods of conjugate gradients for solving linear systems," *Journal of research of the National Bureau of Standards*, vol. 49, no. 6, pp. 409–436, 1952.

[31] N. J. Higham, "Analysis of the cholesky decomposition of a semi-definite matrix," 1990.

[32] A. Krishnamoorthy and D. Menon, "Matrix inversion using cholesky decomposition," in *2013 signal processing: Algorithms, architectures, arrangements, and applications (SPA)*. IEEE, 2013, pp. 70–72.

[33] J. Hensman, N. Fusi, and N. D. Lawrence, "Gaussian processes for big data," *arXiv preprint arXiv:1309.6835*, 2013.

[34] M. Titsias, "Variational learning of inducing variables in sparse gaussian processes," in *Artificial intelligence and statistics*. PMLR, 2009, pp. 567–574.

[35] H. Khalil, *Nonlinear Systems*, ser. Pearson Education.

Prentice Hall, 2002. [Online]. Available: https://books.google.co.uk/books?id=t_d1QgAACAAJ
[36] H. K. Khalil, *Nonlinear systems; 3rd ed.* Upper Saddle River, NJ: Prentice-Hall, 2002, the book can be consulted by contacting: PH-AID: Wallet, Lionel. [Online]. Available: https://cds.cern.ch/record/1173048
[37] Z. Artstein, "Stabilization with relaxed controls," *Nonlinear Analysis: Theory, Methods & Applications*, vol. 7, no. 11, pp. 1163–1173, 1983.
[38] R. A. Freeman and J. A. Primbs, "Control lyapunov functions: New ideas from an old source," in *Proceedings of 35th IEEE conference on decision and control*, vol. 4. IEEE, 1996, pp. 3926–3931.
[39] R. K. Cosner, P. Culbertson, A. J. Taylor, and A. D. Ames, "Robust safety under stochastic uncertainty with discrete-time control barrier functions," *Robotics: Science and Systems*, 2023.
[40] Y. Lin and E. D. Sontag, "A universal formula for stabilization with bounded controls," *Systems & control letters*, vol. 16, no. 6, pp. 393–397, 1991.
[41] 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.
[42] R. A. Ibrahim, *Liquid sloshing dynamics: theory and applications*. Cambridge University Press, 2005.
[43] J. M. Hernández-Lobato and R. Adams, "Probabilistic backpropagation for scalable learning of bayesian neural networks," in *International conference on machine learning*. PMLR, 2015, pp. 1861–1869.
[44] A. Damianou and N. D. Lawrence, "Deep gaussian processes," in *Artificial intelligence and statistics*. PMLR, 2013, pp. 207–215.
[45] I. Steinwart and A. Christmann, *Support vector machines*. Springer Science & Business Media, 2008.
[46] S.-C. Hsu, X. Xu, and A. D. Ames, "Control barrier function based quadratic programs with application to bipedal robotic walking," in *2015 American Control Conference (ACC)*. IEEE, 2015, pp. 4542–4548.
[47] M. Rauscher, M. Kimmel, and S. Hirche, "Constrained robot control using control barrier functions," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 279–285.
[48] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European control conference (ECC)*. IEEE, 2019, pp. 3420–3431.
[49] C. Dawson, Z. Qin, S. Gao, and C. Fan, "Safe nonlinear control using robust neural lyapunov-barrier functions," in *Conference on Robot Learning*. PMLR, 2022, pp. 1724–1735.
[50] W. Xiao, T.-H. Wang, R. Hasani, M. Chahine, A. Amini, X. Li, and D. Rus, "Barriernet: Differentiable control barrier functions for learning of safe robot control," *IEEE Transactions on Robotics*, 2023.
[51] J. Zhao and D. J. Hill, "Passivity and stability of switched systems: A multiple storage function method," *systems & control letters*, vol. 57, no. 2, pp. 158–164, 2008.
[52] T. Hatanaka, N. Chopra, and M. W. Spong, "Passivity-based control of robots: Historical perspective and contemporary issues," in *2015 54th IEEE Conference on Decision and Control (CDC)*. IEEE, 2015, pp. 2450–2452.
[53] B. Capelli, C. Secchi, and L. Sabattini, "Passivity and control barrier functions: Optimizing the use of energy," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1356–1363, 2022.
[54] D. Henrion and M. Korda, "Convex computation of the region of attraction of polynomial control systems," *IEEE Transactions on Automatic Control*, vol. 59, no. 2, pp. 297–312, 2013.
[55] U. Topcu, A. K. Packard, P. Seiler, and G. J. Balas, "Robust region-of-attraction estimation," *IEEE Transactions on Automatic Control*, vol. 55, no. 1, pp. 137–142, 2009.
[56] C. Dawson, S. Gao, and C. Fan, "Safe control with learned certificates: A survey of neural lyapunov, barrier, and contraction methods for robotics and control," *IEEE Transactions on Robotics*, 2023.
[57] R. Ortega, A. Loria, P. J. Nicklasson, H. Sira-Ramirez, R. Ortega, A. Loría, P. J. Nicklasson, and H. Sira-Ramírez, *Euler-Lagrange systems*. Springer, 1998.
[58] O. Romeo, L. Antonio, N. P. Johan, and S.-R. Hebertt, "Passivity-based control of euler-lagrange systems: Mechanical electrical and electro-mechanical applications," *Editorial Springer-Verlag. Great Britain*, 1998.

APPENDIX

A. Proof of Lemma 1

Proof: First, we will prove that the latent function $f + gu \in \mathcal{H}_k$, where $\mathcal{H}_k$ is the RKHS associated with GP kernel k. Under Assumption 1, we have $f(x) = \sum_{p=1}^{\infty} \alpha_p k^f(x, x_p)$, $g(x) = \sum_{q=1}^{\infty} \beta_q k^g(x, x_q)$, for some real coefficients $\{\alpha_p\}, \{\beta_q\}$ and points $\{x_p\}, \{x_q\}$. Thus, the function $f(x) + g(x)u$ can be expressed as

$$f(x) + g(x)u = \sum_{p=1}^{\infty} \alpha_p k^f(x, x_p) + u \sum_{q=1}^{\infty} \beta_q k^g(x, x_q).$$

Because the GP kernel function corresponds to a composite kernel $k(x, u, x', u') = k^f(x, x') + uk^g(x, x')u'$, we have

$$\begin{aligned} k^f(x, x') &= k(x, u, x', 0), \\ uk^g(x, x') &= k(x, u, x', 1) - k(x, u, x', 0). \end{aligned}$$

To show that $f + gu \in \mathcal{H}_k$, we need to express $f + gu$ as a linear combination of k. Setting $u' = 1$ in the definition of k,

we get

$$\begin{aligned}&f(x)+g(x)u\\=&\sum_{p=1}^{\infty}\alpha_p k(x,u,x_p,0)+\sum_{q=1}^{\infty}\beta_q k(x,u,x_q,1)\\-&\sum_{q=1}^{\infty}\beta_q k(x,u,x_q,0),\end{aligned}$$

implying that $f(x)+g(x)u\in\mathcal{H}_k$. Thus, there exists $B_{f,g}>0$, such that $\|f+gu\|_{\mathcal{H}_k}\leq B_{f,g}$. Then (5) can be proved by leveraging Corollary 1 in [29]. ∎

B. Proof of Corollary 1

Proof: By Assumption 1, the function f and g are both Lipschitz continuous with Lipschitz constants L_f and L_g satisfying $L_f^2 = 2B_f^2\|k^f\|_\infty\left\|\frac{\partial k^f}{\partial x}\right\|_\infty$ and $L_g^2 = 2B_g^2\|k^g\|_\infty\left\|\frac{\partial k^g}{\partial x}\right\|_\infty$, and bounded by $\|f\|_\infty\leq B_f^2\|k^f\|_\infty$ and $\|g\|_\infty\leq B_g^2\|k^g\|_\infty$ [13, 45]. For $x,x'\in\mathcal{X}$, the derivative of the Lyapunov function $\dot{V}(x)$ satisfies

$$\begin{aligned}&\left|\dot{V}(x)-\dot{V}(x')\right|\\=&\left|\frac{\partial V(x)}{\partial x}\left(f(x)+g(x)\pi(x)\right)-\frac{\partial V(x')}{\partial x'}\left(f(x')+g(x')\pi(x')\right)\right|\\\leq&|f(x)+g(x)\pi(x)|\cdot\left|\frac{\partial V(x)}{\partial x}-\frac{\partial V(x')}{\partial x'}\right|\\&+\left|\frac{\partial V(x')}{\partial x'}\right|\cdot|f(x)+g(x)\pi(x)-f(x')-g(x')\pi(x')|\\\leq&\left(B_f^2\|k^f\|_\infty+B_g^2B_\pi\|k^g\|_\infty\right)\cdot\left\|\frac{\partial^2V(x)}{\partial x^2}\right\|_\infty\cdot|x-x'|\\&+\sqrt{2}\left\|\frac{\partial V(x)}{\partial x}\right\|_\infty\cdot\left(B_f\sqrt{\|k^f\|_\infty\left\|\frac{\partial k^f}{\partial x}\right\|_\infty}\right)\cdot|x-x'|\\&+\sqrt{2}\left\|\frac{\partial V(x)}{\partial x}\right\|_\infty\cdot\left(B_g\pi_g\sqrt{\|k^g\|_\infty\left\|\frac{\partial k^g}{\partial x}\right\|_\infty}\right)\cdot|x-x'|\\\triangleq&L\cdot|x-x'|,\end{aligned}$$

where L is the Lipschitz constant of $\dot{V}$. Then, (8) is the sufficient condition of (7), and this corollary can be proved by using Theorem 1. ∎

C. Energy Shaping Control

In this section, we demonstrate that the computation-aware learning framework is not limited to stable control tasks. Indeed, it can be extended to analyze and achieve safety and passivity based on energy-like functions, which we refer to as energy-shaping control.

Energy shaping control is extensively used in controller design. Its core idea is to synthesize a controller based on an energy-like function, reshaping the system's natural energy to achieve desired control performance. For example, in regulation and tracking tasks, Lyapunov functions are widely used to design stable controllers [8–12]. In reach-avoid tasks, barrier functions or potential functions are commonly employed to ensure the safety of controllers [46–50]. Additionally, in the control of Euler-Lagrange models, storage functions are used to achieve passivity [51–53].

The performance of a given controller, denoted as $\pi(x)$, can be effectively evaluated by determining the invariant set within which key properties like stability, safety, and passivity are maintained.

Definition 4 (Invariant Set)**.** *A set $\mathcal{S}$ is said to be an invariant set if for all $x(0)\in\mathcal{S}$, we have $x(t)\in\mathcal{S}$, $\forall t>0$.*

For stability analysis, ROA is the invariant set that represents the area where a given control law $\pi(x)$ guarantees the asymptotic stability of an equilibrium point [54, 55]. Besides, for safety analysis, the safe set [48, 56] is the invariant set that defines the states in which the system's operation remains within safe boundaries. This ensures that safety constraints are not violated, which is significant for safety-critical systems. The invariant set is typically related to a well-defined energy function:

Definition 5 (Energy Function and Power Function)**.** *For a given policy $\pi(x)$, an energy function $E(x)\in\mathbb{C}^2$ and a power function $P(x)\in\mathbb{C}^1$ are well-defined for set $\mathcal{S}$ if the following holds: for all $x\in\mathcal{S}=\{x\in\mathcal{X}|E(x)\in\mathcal{E}\}$, if the following condition is satisfied:*

$$\dot{E}(x)=\frac{\partial E(x)}{\partial x}\left(f(x)+g(x)\pi(x)\right)\leq P(x). \tag{19}$$

Then, $\mathcal{S}$ is an invariant set.

In Definition 5, we call (19) as the energy variation condition because it requires the time variation of the energy function $\dot{E}(x)$ to be slower than a power function $P(x)$.

In control theory, the terms "energy function" and "power function" are not formally defined, but the concept of using energy-like functions for analyzing control systems is widespread. For stability analysis, the energy function $E(x)$ is often chosen as the Lyapunov function, and the power function $P(x)$ is set by $P(x)=-\gamma(E(x))$, where γ is a class $\mathcal{K}$ function. The invariant set $\mathcal{S}$ is defined as the ROA of the system, given by $\mathcal{S}=\{x\in\mathcal{X}\,|\,0\leq E(x)\leq c\}$, with $c>0$. For safety analysis, the negation of the barrier function is used as the energy function, and the power function is defined as $P(x)=\alpha(E(x))$, where α is an extended class $\mathcal{K}_\infty$ function. The invariant set $\mathcal{S}$ is considered as the safe set, denoted by $\mathcal{S}=\{x\in\mathcal{X}\,|\,E(x)\leq 0\}$.

Specifically, for mechanical systems represented by the input-output gravity-compensated Euler-Lagrange model [57]:

$$\begin{cases}M(\chi)\ddot{\chi}+C(\chi,\dot{\chi})\dot{\chi}+D\dot{\chi}+\frac{\partial P(\chi)}{\partial\chi}=u,\\y=\dot{\chi},\end{cases}$$

where χ represents the system's position, and $x=\left[\chi,\,\dot{\chi}\right]$ is the system state, u and y are the control input and output respectively. For passivity analysis [53, 58], the energy function is often selected as the storage function of the system, satisfying $E(x)=P(\chi)+\frac{1}{2}\dot{\chi}^TM(\chi)\dot{\chi}$, and the power function is defined as $P(x)=\pi(x)y$ for a given policy $\pi(x)$. The

power dissipated by the system, $P_d(x) = \pi(x)y - \dot{E}(x)$, also considered as a passivity margin, should satisfy $P_d(x) \geq 0$ for a passive system. In this case, the invariant set $\mathcal{S}$ is defined as $\mathcal{S} = \{x \in \mathcal{X} | E(x) \geq 0\}$.

Using the concept of energy functions and invariant sets, the stability analysis and controller design techniques presented in this paper can be easily extended to various energy shaping controls, including those for safety or passivity properties.