2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

Sample-and-computation-efficient Probabilistic Model Predictive Control with Random Features

Cheng-Yu Kuo[1*], Yunduan Cui[1], Takamitsu Matsubara[1]

***Abstract*— Gaussian processes (GPs) based Reinforcement Learning (RL) methods with Model Predictive Control (MPC) have demonstrated their excellent sample efficiency. However, since the computational cost of GPs largely depends on the training sample size, learning an accurate dynamics using GPs result in low control frequency in MPC. To alleviate this trade-off and achieve a sample-and-computation-efficient nature, we propose a novel model-based RL method with MPC. Our approach employs a linear Gaussian model with randomized features using the *Fastfood* as an approximated GP dynamics. Then, we derive an analytic moment-matching scheme in state prediction with the model and uncertain inputs. As a result, the computational cost of the MPC in our RL method does not depend on the training sample size and can improve the control frequency over previous methods. Through experiments with simulated and real robot control tasks, the sample efficiency, as well as the computation efficiency of our model-based RL method, are demonstrated.**

I. Introduction

Reinforcement learning (RL) enables a robot to learn a specific task from scratch but often suffers from being data inefficient. A potential solution to increase data efficiency is the combination of model-based RL(MBRL) [1] and Gaussian Process (GPs) [2]. As a state-of-the-art MBRL method, PILCO [3] explicitly incorporates GPs model uncertainty into planning and control to reduce the effect of noisy observations and data scarcity. Assuming a controllable target dynamics, PILCO learns an optimal policy by long-term planning from an initial state.

Nevertheless, PILCO [3] has no countermeasures to unpredictable in-trial disturbances in an open environment. Some probabilistic/stochastic Model Predictive Control (MPC) is proposed to frequently plans for optimal control sequence by minimizing the long-term expected cost with uncertainty [4]–[7]. Where the long-term expected cost is predicted by propagating uncertainty through time via analytic moment-matching method [8, 9].

However, the computational cost of exploiting moment-matching with these efficient approximations still largely depends on training sample size in a quadratic way. A larger training sample size increases the precision of modeled system dynamics, but the MPC control period will also be extended. Therefore, there is a difficulty in achieving high-frequency MPC with those approaches, where short MPC control period limits training sample size, which returns low precision. Oppositely, given enormous samples will lead to an impractical MPC control period. It is resulting in a trade-off between ideal MPC control period and state prediction precision.

Recently, finite feature space approximating methods are proposed to calculate the kernel expansions efficiently. Random Kitchen Sink (RKS) [10] follows Bochner's theorem to precisely approximate the kernel expansions by randomly sample features from frequency components of the Fourier transformed kernels. Fastfood enhances the RKS by further improve the computation efficiency by approximating the sampled matrix in RKS from scratch within low variance [11]. These methods dramatically reduced the computational cost of calculating the kernel expansions with minimal compromise in precision.

With the above in mind, in this study, we develop a novel MBRL approach with probabilistic MPC, which holds both sample and computation efficient characteristics. We extract the random features from the Fastfood kernel expansion to efficiently approximate GPs dynamics with limited features. Then, we model the probabilistic system dynamics by approximately representing the GPs dynamics with a linear Gaussian model [12] with the Fastfood features; this alleviates the trade-off between MPC control period and prediction precision. We derive an analytic moment-matching scheme with the linear model using random features for propagating the model uncertainty in state prediction. As a result, the MPC control period will stay constant regardless of the training sample size. Compared to previous works, the contributions of this paper are the following:

1) Propose a sample-and-computation-efficient MBRL approach with probabilistic MPC, which is capable of high-control frequency under any training sample size.
2) Empirically verify the sample efficiency as well as the computational cost in our method through experiments with simulated and real robots.

The remainder of this paper is organized as follows: Section II presents preliminaries with some related works to our approach and its' shortcomings. Section III details how our approach manages the shortcomings in Section II. Section IV shows the simulation and real experiment result with arm robot. Discussions follow in Section V.

II. Preliminaries

In this section, we will introduce the comprehension knowledge to our approach. Start with a brief introduction to the standard GPs probabilistic dynamics, following by a moment-matching approach with standard GPs and its shortcomings in the computational cost. Next, we introduce

* Corresponding author: kuo.cheng-yu.jy5@is.naist.jp
[1] Graduate School of Science and Technology, Nara Institute of Science and Technology (NAIST), Nara, Japan

MPC with long-term cost-minimizing by exploiting moment-matching. Following by MBRL methods with MPC.

A. GPs Dynamics

Consider an unknown dynamics $f : \mathbb{X} \times \mathbb{U} \to \mathbb{X}$ with the input of state $\mathbf{x} \in \mathbb{X} \subset \mathbb{R}^D$ and control signal $\mathbf{u} \in \mathbb{U} \subset \mathbb{R}^U$:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) + \epsilon, \tag{1}$$

with system noise $\epsilon \sim \mathcal{N}(0, \mathbf{\Sigma}_n)$, where $\mathbf{\Sigma}_n = \mathrm{diag}\left[\sigma_{\mathrm{n},1}^{-2}, ..., \sigma_{\mathrm{n,D}}^{-2}\right]$. The unknown dynamics is then trained with training input tuples $\tilde{\mathbf{x}}_t := (\mathbf{x}_t, \mathbf{u}_t)$, and corresponding training target $\mathbf{y}_t = [y_{t,1}, ..., y_{t,D}]^\top = \mathbf{x}_{t+1}$. For each target dimension a, a latent GP model $y_{t,a} = f_a(\tilde{\mathbf{x}}_t) + \epsilon_a$ is trained with mean function $\mu_a(\cdot)$ and covariance kernel function:

$$k_a(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j) = \alpha_a^2 \exp\left(-\frac{1}{2}(\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j)^\top \mathbf{\Lambda}_a^{-1}(\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j)\right), \tag{2}$$

where α_a^2 is signal variance, Λ_a^{-1} is squared length scale diagonal matrix. Both are optimized through *Maximum Marginal Likelihood Estimation* (MMLE) [2, 13]. The predictive distribution of a new input $\tilde{\mathbf{x}}_*$ can be derived as:

$$p(f_a(\tilde{\mathbf{x}}_*)|\tilde{\mathbf{X}}, \mathbf{y}_a) \sim \mathcal{GP}\left(\mu_a(\tilde{\mathbf{x}}_*), \sigma_a^2(\tilde{\mathbf{x}}_*)\right), \tag{3}$$

$$\mu_a(\tilde{\mathbf{x}}_*) = \boldsymbol{k}_{a,*}^\top \beta_a, \tag{4}$$

$$\sigma_a^2(\tilde{\mathbf{x}}_*) = k_{a,**} - \boldsymbol{k}_{a,*}^\top \mathbf{K}_{\mathbf{y}_a}^{-1} \boldsymbol{k}_{a,*}, \tag{5}$$

where $\tilde{\mathbf{X}} = [\tilde{\mathbf{x}}_{t,1}, ..., \tilde{\mathbf{x}}_{t,N}]$, $\mathbf{y}_a = [y_{t,a,1}, ..., y_{t,a,N}]$ are the training input and target set. $\boldsymbol{k}_{a,*}^\top = k_a(\tilde{\mathbf{X}}, \tilde{\mathbf{x}}_*)$; $k_{a,**} = k_a(\tilde{\mathbf{x}}_*, \tilde{\mathbf{x}}_*)$; $\mathbf{K}_{a,ij} = k_a(\tilde{\mathbf{x}}_{t,i}, \tilde{\mathbf{x}}_{t,j})$ is the corresponding element in $\mathbf{K}_a$; $\mathbf{K}_{\mathbf{y}_a} = \left(\mathbf{K}_a + \sigma_{n,a}^2 \mathbf{I}_D\right)$ and $\beta_a = \left(\mathbf{K}_a + \alpha_a^2 \mathbf{I}_D\right)^{-1} \mathbf{y}_a$.

B. Long-term State Prediction Using Moment-Matching

Uncertainty propagation using Moment-matching predicts future states $\mathbf{x}_{t+1}$ with uncertainties [8, 9]. Given a probabilistic state $\mathbf{x}_t \sim \mathcal{N}(\boldsymbol{\mu}_t, \mathbf{\Sigma}_t) \in \mathbb{R}^D$. Following the law of iterated expectations (Fubini's theorem), the predictive mean $\mu_{a,t+1}$ and variance $\sigma_{a,t+1}^2$ of a^{th} latent model of $\mathbf{x}_{t+1}$ are obtained using exploiting moment-matching. We obtain $\mu_{a,t+1}$ by

$$\mu_{a,t+1} = \mathbb{E}_{\mathbf{x}_t}[f_a(\mathbf{x}_t, \mathbf{u}_t)|\boldsymbol{\mu}_t, \mathbf{\Sigma}_t] = \beta_a^\top \mathbf{q}_a, \tag{6}$$

where $\mathbf{q}_a = \left[q_1, ...q_N\right]^\top \in \mathbb{R}^N$ with

$$\begin{aligned} q_i = \alpha_a^2 \left|\mathbf{\Sigma}_t \mathbf{\Lambda}_a^{-1} + \mathbf{I}\right|^{-\frac{1}{2}} \\ \times \exp(-\frac{1}{2}(\tilde{\mathbf{X}}_i - \boldsymbol{\mu})^\top (\mathbf{\Sigma}_t + \mathbf{\Lambda}_a)^{-1}(\tilde{\mathbf{X}}_i - \boldsymbol{\mu})). \end{aligned} \tag{7}$$

We obtain $\sigma_{a,t+1}^2$ by

$$\begin{aligned} \sigma_{a,t+1}^2 = \mathbb{E}_{\mathbf{x}_t}[\sigma_a^2(\mathbf{x}_t, \mathbf{u}_t)|\boldsymbol{\mu}_t, \mathbf{\Sigma}_t] \\ + (\mathbb{E}_{\mathbf{x}_t}[\mu_a(\mathbf{x}_t, \mathbf{u}_t)^2|\boldsymbol{\mu}_t, \mathbf{\Sigma}_t] \\ - \mathbb{E}_{\mathbf{x}_t}[\mu_a(\mathbf{x}_t, \mathbf{u}_t)|\boldsymbol{\mu}_t, \mathbf{\Sigma}_t]^2); \end{aligned} \tag{8}$$

following Eq.(3), we further calculate

$$\sigma_{a,t+1}^2 = \alpha_a^2 - \mathrm{tr}(\mathbf{K}_{\mathbf{y}_a}^{-1}\mathbf{Q}_a) + \beta_a^\top \mathbf{Q}_a \beta_a - \mu_{a,t+1}^2. \tag{9}$$

The entries of $\mathbf{Q}_a \in \mathbb{R}^{N \times N}$ are given by

$$\begin{aligned} \mathbf{Q}_{a,ij} = \frac{k_a(\tilde{\mathbf{X}}_i, \boldsymbol{\mu}_t) k_a(\tilde{\mathbf{X}}_j, \boldsymbol{\mu}_t)}{\left|2\mathbf{\Sigma}_t \mathbf{\Lambda}_a^{-1} + \mathbf{I}\right|^{\frac{1}{2}}} \\ \times \exp(\tilde{Z}_{ij,t}^\top (\mathbf{\Sigma}_t + \frac{1}{2}\mathbf{\Lambda}_a)^{-1} \mathbf{\Sigma}_t \mathbf{\Lambda}_a^{-1} \tilde{Z}_{ij,t}), \end{aligned} \tag{10}$$

where $\tilde{Z}_{ij,t} = ((\tilde{\mathbf{X}}_i - \tilde{X}_j)/2) - \boldsymbol{\mu}_t$.

The computational cost of exploiting moment-matching with standard GPs is largely depend on the training sample size N in a quadratic way, owing to the calculation of $\mathbf{Q}_a$.

C. MPC with Long-term Cost Minimizing

The MPC is a closed-loop feedback controller with each time step an open-loop an H steps control sequence planning. For each time step t, the MPC observes the current state $\mathbf{x}_t$ from system feedbacks and repeats moment-matching state predictions recursively to plan an open-loop H steps control sequence $\mathbf{u}_t, ..., \mathbf{u}_{t+H-1}$ by minimizing long-term cost via Sequential Quadratic Programming(SQP) [14]:

$$\left[\mathbf{u}_t, ..., \mathbf{u}_{t+H-1}\right] = \underset{\mathbf{u}_t, ..., \mathbf{u}_{t+H-1}}{\arg\min} \sum_{s=t}^{t+H-1} \ell\left(\mathbf{x}_s, \mathbf{u}_s\right). \tag{11}$$

where ℓ is the immediate cost, defined to accomplish the intended task. The first control signal $\mathbf{u}_t$ is then executed, shifts the MPC to the next time step $\mathbf{x}_{t+1}$. Repeating this routine until reaching the goal state.

D. MBRL with probabilistic MPC

We follow the similar MBRL approach to [7], Algorithm 1. At the first trail, the GPs dynamics are initially updated with samples from operating random control signals to the environment. Next, the RL process starts by running N_{trial} trials in total with $L_{rollout}$ each. At each rollout, given the observed current state, the MPC optimizes the control sequence and operate the first control signal to the environment. Corresponding samples are collected in each rollout and update the GPs dynamic at the end of each trial.

III. Approach

Based on Section II, in this section, we introduce our approach, Sample-and-computation-efficient Probabilistic MPC (SCP-MPC). Applying Linear Gaussian Model (LGM) with random features to exploiting moment-matching, we alleviate the trade-off between MPC control period and prediction precision at "*ActionSearch*" stage in Algorithm 1.

A. Fastfood Kernel Approximation and Feature Extraction

The RKS [10] approximates the covariance kernel expansions $k(x, x')$ by drawing entire $\mathbf{Z}_\omega$ i.i.d from Fourier transformed kernel $\rho(\omega)$,

$$\begin{aligned} k(x, x') = \int \rho(\omega) \exp\left(i \left\langle \omega, x - x' \right\rangle\right) \mathrm{d}\omega \\ \approx \left\langle \phi_j(x), \phi_j(x') \right\rangle, \end{aligned} \tag{12}$$

where $\phi_j(x) = n^{-\frac{1}{2}} \exp(i\mathbf{Z}_\omega x) \in \mathbb{R}^n$.

Algorithm 1: MBRL with MPC

Initialization
number of trial N_{trial}
length of rollout for each trial $L_{rollout}$
step horizon of MPC prediction H
samples for training dynamics model $\mathbf{x} = \{\}$,$\mathbf{y} = \{\}$
cost function for intended task $\ell()$
Generate random samples with random actions
`ResetPosition()`
for $i = 1, 2, ..., L_{rollout}$ **do**
  $\mathbf{x}_i$ = `ReadSensor()`
  u_i = `RandomActions()`
  `Operate(`u_i`)`
  $\mathbf{y_i}$ = `ReadSensor()`
  $\mathbf{x} = \{\mathbf{x}, \mathbf{x_i}\}, \mathbf{y} = \{\mathbf{y}, \mathbf{y_i}\}$
MBRL with MPC
$model$ = `trainModel(`$\mathbf{x}, \mathbf{y}$`)`
for $i = 1, 2, ..., N_{trial}$ **do**
  `ResetPosition()`
  for $j = 1, 2, ..., L_{rollout}$ **do**
    $\mathbf{x}^*$ = `ReadSensor()`
    $\mathbf{u}$ = `ActionSearch(`$\mathbf{x}^*, H, model, \ell()$`)`
    `Operate(`$\mathbf{u}(1)$`)`
    $\mathbf{y}^*$ = `ReadSensor()`
    $\mathbf{x} = \{\mathbf{x}, \mathbf{x}^*\}, \mathbf{y} = \{\mathbf{y}, \mathbf{y}^*\}$
  $model$ = `trainModel(`$\mathbf{x}, \mathbf{y}$`)`

Fastfood [11] further generates the sampled matrix in RKS from scratch and approximates kernel expansion via

$$k(x, x') \approx \langle \phi(x), \phi(x') \rangle , \tag{13}$$

where $\phi(x) = n^{-\frac{1}{2}} \exp(i\mathbf{V}x)$ with $\mathbf{V} = \frac{1}{\sigma_f \sqrt{d}} SHG\Pi HB$, and $\mathbf{V} \in \mathbb{R}^{n \times d}$. The number of features are decided by $d = 2^m$ with $m \in \mathbb{N}$.

G is the diagonal Gaussian scaling matrix with each element drawn iid from $G_{ii} \sim \mathcal{N}(0, 1)$. B is the diagonal binary scaling matrix with each element drawn iid from $B_{ii} \in \{\pm 1\}$. Next, apply Walsh-Hadamard transformation H on G and B for a denser matrix with varieties between rows. $HG\Pi HB$ generates a matrix with m iid Gaussian random vairables in each row, where Π is permutation matrix to decorrelate two Hadamard transformation. We follow [15] to generate the diagonal scaling matrix S with element $S_{ii} = \|\omega_i\| \|G\|_{\text{Frob}}^{-1}$ by drawing ω_i from the spherically symmetric distribution defined by $\rho(\omega)$ in Eq.(12). The value of σ_f is optimized in Section III-B.

Rewrite a real map of the complex feature ϕ as:

$$\begin{aligned} \langle \phi(x), \phi(x') \rangle &= n^{-1} \exp(i\mathbf{V}(x - x')) = n^{-1} \cos(\mathbf{V}(x - x')) \\ &= n^{-1} (\cos(\mathbf{V}x) \cos(\mathbf{V}x') + \sin(\mathbf{V}x) \sin(\mathbf{V}x')) \\ &= \langle \phi^*(x), \phi^*(x') \rangle , \end{aligned} \tag{14}$$

where $\phi^*(x) = n^{-\frac{1}{2}} \left[\cos(\mathbf{V}x), \sin(\mathbf{V}x)\right]^\top$, [15].

After adding a bias term, our use of feature map is defined as $\Phi(\mathbf{x}_t, \mathbf{u}_t) = \begin{bmatrix} 1 & \phi^*(\mathbf{x}_t, \mathbf{u}_t) \end{bmatrix}^\top : \mathbb{X} \times \mathbb{U} \to \mathbb{R}^M$.

B. Approximated GPs with Finite Random Features

We represent the function $f(\cdot)$ in Eq. (1) by approximating the GPs dynamics via LGM [12] with Fastofood random features $\Phi(\mathbf{x}, \mathbf{u})$, LGM-FF.

The predictive distribution of output given input tuple $\tilde{\mathbf{x}}_*$ is:

$$p(f_a^*(\cdot)|\tilde{\mathbf{x}}_*, \tilde{\mathbf{X}}, \mathbf{y}_a) \sim \mathcal{GP}\left(\mu_a(\tilde{\mathbf{x}}_*), \sigma_a^2(\tilde{\mathbf{x}}_*)\right), \tag{15}$$

$$\mu_a(\tilde{\mathbf{x}}_*) = \bar{\mathbf{w}}_a^\top \Phi_a(\tilde{\mathbf{x}}_*), \tag{16}$$

$$\sigma_a^2(\tilde{\mathbf{x}}_*) = \Phi_a(\tilde{\mathbf{x}}_*)^\top \mathbf{A}_a^{-1} \Phi_a(\tilde{\mathbf{x}}_*), \tag{17}$$

where $\mathbf{w}_a \in \mathbb{R}^{M \times D}$ is corresponding weight.

Given the prior knowledge to the weight of a^{th} latent model $\mathbf{w}_a \sim \mathcal{N}(0, \mathbf{\Sigma}_{p,a})$ where $\mathbf{\Sigma}_{p,a} = \sigma_{p,a}^2 \mathbf{I}_M$. Each value of $\bar{\mathbf{w}}_a$ is learned through *maximum a posteriori* (MAP) estimation [2, 12, 13], given as followed:

$$\bar{\mathbf{w}}_a = \sigma_{n,a}^{-2} \mathbf{A}_a^{-1} \Phi_a(\tilde{\mathbf{X}}) \mathbf{y}_a, \tag{18}$$

$$\mathbf{A}_a = \sigma_{n,a}^{-2} \Phi_a(\tilde{\mathbf{X}}) \Phi_a(\tilde{\mathbf{X}})^\top + \sigma_{p,a}^{-2} \mathbf{I}_M. \tag{19}$$

where $\sigma_{n,a}^{-2}$, $\sigma_{p,a}^{-2}$ and σ_f in $\Phi_a(\cdot)$ are optimized via MMLE.

The selection of features largely depends on parameters d, $\bar{\mathbf{w}}$ and σ_f. The weight $\bar{\mathbf{w}}$ and σ_f adjust the amplitude and frequency of the frequency-based features. Since both $\bar{\mathbf{w}}$ and σ_f are optimized by MMLE, number of features M is simply decided by gradually increase the value of d until the LGM precisely fits the samples. However, with frequency-based features, the minimum requirement of M mainly depends on the state dynamics.

The computational cost of learning the system dynamics with LGM is $\mathcal{O}(M^3)$ given by the inversion of matrix $\mathbf{A}_a \in \mathbb{R}^{M \times M}$, where M is the dimension of feature space which is fixed by the system dynamics. In comparison, the standard GPs is $\mathcal{O}(N^3)$. Since all latent models are trained independently, for a simpler notation, subscription a is omitted in the following sections.

C. Moment-Matching with LGM-FF

Given a probabilistic state $\mathbf{x}_t$ and a deterministic control signal $\mathbf{u}_t$, the predictive state distribution $p(\mathbf{x}_{t+1}) = p(f(\mathbf{x}_t, \mathbf{u}_t)|\boldsymbol{\mu}_t, \mathbf{\Sigma}_t, \mathbf{u}_t) = \mathcal{N}\left(\boldsymbol{\mu}_{t+1}, \boldsymbol{\sigma}_{t+1}^2\right)$ are calculated by following Fubini's law. In the followings, notation is simplified as $\Phi_t := \Phi(\mathbf{x}_t, \mathbf{u}_t)$; $\mu_t := \mu(\mathbf{x}_t, \mathbf{u}_t)$; $\sigma_t^2 := \sigma^2(\mathbf{x}_t, \mathbf{u}_t)$. For each latent model, we calculate μ_{t+1} by

$$\begin{aligned} \mu_{t+1} &= \mathbb{E}_{\mathbf{x}_t}[f(\mathbf{x}_t, \mathbf{u}_t)|\boldsymbol{\mu}_t, \mathbf{\Sigma}_t] = \int \mu_t p(\mathbf{x}_t) \mathrm{d}\mathbf{x}_t \\ &= \bar{\mathbf{w}}^\top \int \Phi_t p(\mathbf{x}_t) \mathrm{d}\mathbf{x}_t = \bar{\mathbf{w}}^\top \mathbf{q}, \end{aligned} \tag{20}$$

where $\mathbf{q} = \int \Phi_t p(\mathbf{x}_t) \mathrm{d}\mathbf{x}_t = [1, \tilde{\mathbf{q}}]^\top \in \mathbb{R}^M$ is calculated by

$$\tilde{\mathbf{q}} = \begin{bmatrix} n^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{V}_\mathbf{x} \mathbf{\Sigma}_t)^{\circ 2}\right) \circ \cos(\mathbf{V}[\boldsymbol{\mu}_t \ \mathbf{u}_t]^\top) \\ n^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{V}_\mathbf{x} \mathbf{\Sigma}_t)^{\circ 2}\right) \circ \sin(\mathbf{V}[\boldsymbol{\mu}_t \ \mathbf{u}_t]^\top) \end{bmatrix}; \tag{21}$$

$\circ$ denotes component-wise operations. Also, $\mathbf{V} = [\mathbf{V}_\mathbf{x}, \mathbf{V}_\mathbf{u}]$ such that $\mathbf{V}\left[\boldsymbol{\mu}_t, \mathbf{u}_t\right]^\top = \mathbf{V}_\mathbf{x} \boldsymbol{\mu}_t + \mathbf{V}_\mathbf{u} \mathbf{u}_t \in \mathbb{R}^n$.

Again, we obtain σ_{t+1}^2 by following Fubini's law

$$\begin{aligned}\sigma_{t+1}^2 &= \mathbb{E}_{\mathbf{x}_t}[\sigma_t^2|\boldsymbol{\mu}_t,\boldsymbol{\Sigma}_t] \\ &+ (\mathbb{E}_{\mathbf{x}_t}[\mu_t^2|\boldsymbol{\mu}_t,\boldsymbol{\Sigma}_t] - \mathbb{E}_{\mathbf{x}_t}[\mu_t|\boldsymbol{\mu}_t,\boldsymbol{\Sigma}_t]^2) \\ &= \int \Phi_t^\top \mathbf{A}^{-1}\Phi_t p(\mathbf{x}_t)\mathrm{d}\mathbf{x}_t \\ &+ \int \Phi_t^\top \bar{\mathbf{w}}\bar{\mathbf{w}}^\top \Phi_t p(\mathbf{x}_t)\mathrm{d}\mathbf{x}_t - (\bar{\mathbf{w}}^\top \mathbf{q})^2,\end{aligned} \tag{22}$$

where $\mathbb{E}_{x_t}[\mu_t|\boldsymbol{\mu}_t,\boldsymbol{\Sigma}_t]^2$ follows Eq.(20). The desired predicted variance σ_{t+1}^2 with uncertain input $\mathbf{x}_t$ is

$$\begin{aligned}\sigma_{t+1}^2 &= \mathrm{tr}\left(\mathbf{A}^{-1}\int \Phi_t\Phi_t^\top p(\mathbf{x}_t)\mathrm{d}\mathbf{x}_t\right) \\ &+ \bar{\mathbf{w}}^\top \underbrace{\int \Phi_t\Phi_t^\top p(\mathbf{x}_t)\mathrm{d}\mathbf{x}_t}_{:=\mathbf{Q}} \bar{\mathbf{w}} - \mu_{t+1}^2 \\ &= \mathrm{tr}\left(\mathbf{A}^{-1}\mathbf{Q}\right) + \bar{\mathbf{w}}^\top \mathbf{Q}\bar{\mathbf{w}} - \mu_{t+1}^2.\end{aligned} \tag{23}$$

The inner products re-arranged to pull expressions that are independent to $\mathbf{x}_t$ in the integrals. $\mathbf{Q} \in \mathbb{R}^{M\times M}$ is given by

$$\mathbf{Q} = \int \Phi_t\Phi_t^\top p(\mathbf{x}_t)\mathrm{d}\mathbf{x}_t = \begin{bmatrix} 1 & \tilde{\mathbf{q}}^\top \\ \tilde{\mathbf{q}} & \begin{matrix}\mathbf{CC} & \mathbf{CS} \\ \mathbf{CS}^\top & \mathbf{SS}\end{matrix}\end{bmatrix}, \tag{24}$$

where $\mathbf{CC}, \mathbf{CS}, \mathbf{SS} \in \mathbb{R}^{n\times n}$ are calculated by Eq.(28-30).

The moment-matching prediction summarized as follow

$$\mu_{t+1} = f^\mu(\boldsymbol{\mu}_t,\boldsymbol{\Sigma}_t,\mathbf{u}_t) = \bar{\mathbf{w}}^\top \mathbf{q}, \tag{25}$$

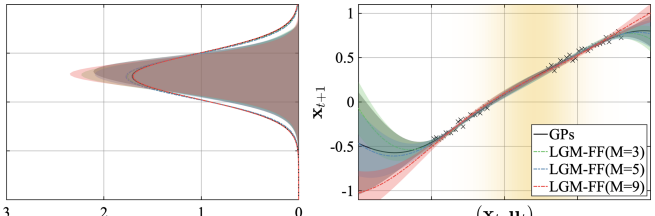
$$\begin{aligned}\sigma_{t+1}^2 &= f^\sigma(\boldsymbol{\mu}_t,\boldsymbol{\Sigma}_t,\mathbf{u}_t,\mu_{t+1}) \\ &= \mathrm{tr}\left(\mathbf{A}^{-1}\mathbf{Q}\right) + \bar{\mathbf{w}}^\top \mathbf{Q}\bar{\mathbf{w}} - \mu_{t+1}^2.\end{aligned} \tag{26}$$

Both $\bar{\mathbf{w}}$ and $\mathbf{A}^{-1}$ are pre-calculated in the training stage by Eq.(18). Thus, the computational cost of exploiting moment-matching with LGM-FF is $\mathcal{O}(M^2)$ in calculating $\mathbf{Q} \in \mathbb{R}^{M\times M}$, i.e., independent to training sample size N. It results in a semi-constant time consumption in predictions throughout all time steps, regardless of the growth of the training sample size.

A single dimension prediction task shows the performance comparison of exploiting moment-matching of standard GPs and LGM-FF. Focusing on prediction accuracy of LGM-FF with various numbers of total features, $M = [3, 5, 9]$. All four models learn the system dynamics from 50 samples drawn from an underlying linear function with noise, Figure 1. Given a probabilistic state-input $\tilde{\mathbf{x}}_t = \mathcal{N}(\mu_t = 0.2, \sigma_t^2 = 0.2^2)$, we obtain four state predictions by exploiting moment-matching on each model. Estimated by KL-divergence, moment-matching with LGM-FF accurately approximates its GPs counterpart with limited numbers of features, Table I. This shows the computation-efficient nature of the moment-matching scheme with LGM-FF.

IV. Experiment Results

In this section, we focus on comparing the performance of SCP-MPC with SPMPC [7] in simulation. We further extend SCP-MPC to real-time tracking with a real robot. In both



	Computational Cost
Standard GPs	$\mathcal{O}(N^3)$
LGM-FF	$\mathcal{O}(M^3)$

Fig. 1: Exploiting moment-matching with GP and LGM-FF and corresponding computational cost. Bottom-right: state distribution at time state t. Upper-right: probabilistic dynamics learned from observed samples (black cross), with corresponding variance (shade). Upper-left: true predictive distributions (shade) and its' approximated Gaussian distributions (line).

TABLE I: KL-divergence between predictive distributions of moment-matching with LGM-FF and GPs with corresponding mean and variance.

	GPs	LGM-FF		
		M=3	M=5	M=9
mean	0.2637	0.2616	0.2600	0.2634
variance	0.2333	0.2269	0.2250	0.2347
KL-divergence	0	0.0008	0.0014	0.0004

experiments, cost functions are defined as follow:

$$\begin{aligned}\ell(\mathbf{x}_t,\mathbf{u}_t) &= \mathbf{k}_{\boldsymbol{s}}\|\boldsymbol{s}^{ref} - \boldsymbol{s}_t\|_2 + \mathbf{k}_{\mathbf{u}}\|\mathbf{u}_t\|_1 \\ \boldsymbol{s}_t &= \mathcal{FK}(\mathbf{x}_t)\end{aligned} \tag{27}$$

where $\boldsymbol{s}^{ref}$ is the reference target position in task space; $\boldsymbol{s}_t$ is the end-effector position of the arm calculated via forward kinematics $\mathcal{FK}$. The weight variables are set as $\mathbf{k}_{\boldsymbol{s}} = 1.0$ and $\mathbf{k}_{\mathbf{u}} = 0.0$.

All computation are done by a computer with specs: Intel Core i9-9900K, 64GB RAM, Nvidia GTX1060 graphics.

A. Simulation Experiments

We setup a simple two-DoF(Degrees of Freedom) arm to demonstrate the raw performance difference between our approach (SCP-MPC) and SPMPC [7].

1) Details of Test Bench Setup: The k-DoF arm simulator is constructed with k actuators and the first actuator fixed at the origin. All actuators are linked together by k bars in same plane, with total length l and each bar length l/k. The goal is to bring the arm tip to target position by MPC.

Due to periodic behavior of rotation, we define the state with the complex representation of angular position as $\mathbf{x}_t = \left[\sin(\Theta_t), \cos(\Theta_t)\right]$, where $\Theta_t = \left[\theta_{1,t}, ..., \theta_{k,t}\right]^\top$ and Θ_t is directly observed by the actuator encoder feedback. The corresponding control signal is defined as $\mathbf{U}_t = \left[\sin(\mathbf{u}_t), \cos(\mathbf{u}_t)\right]$, where $\mathbf{u}_t = \left[u_{1,t}, ..., u_{k,t}\right]^\top$. $\theta_{i,t}$ and $u_{i,t}$ denotes the angular position and MPC control signal

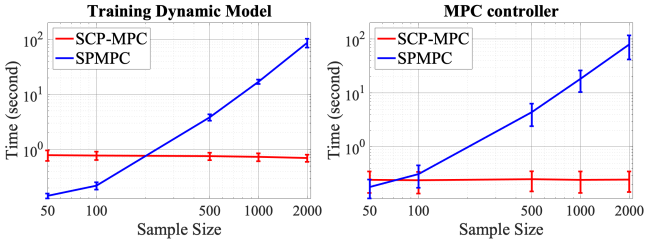


Fig. 2: Average time consumption to train the system dynamics model (left) and MPC control sequence planning in each step (right).

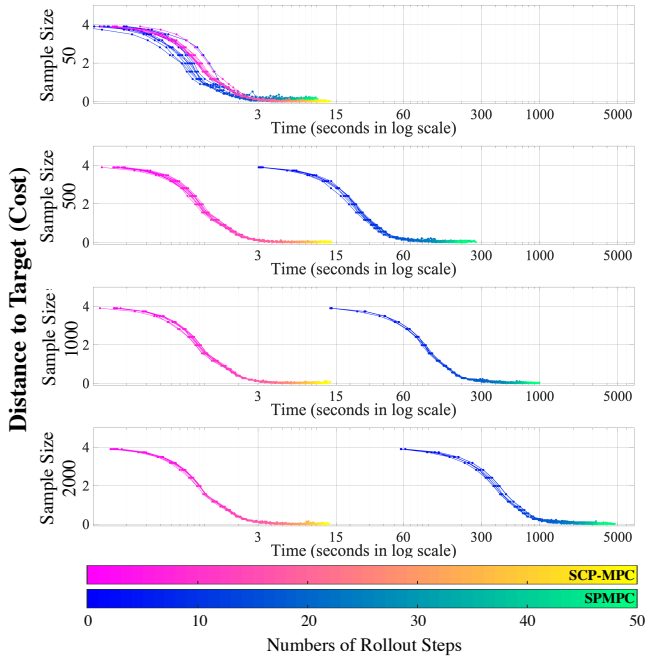


Fig. 3: The comparison of MPC performance with various sample size pre-trained system dynamics (10 trials). Color gradients represent steps in each rollout.

(angular position translation) for i^{th} actuator at time step t. For each element in $\mathbf{x}_t$, a latent GPs model $f_a(\mathbf{x}_t, \mathbf{U}_t)$ is trained to predict corresponding element in $\mathbf{x}_{t+1}$.

Experiment vairables is set as followed: two-DoF arm, $k = 2$. Rollout for each trial $L_{rollout} = 50$. MPC prediction horizon $H = 3$. Constraints of control signal $u_{k,t} \in [-10, 10]^\circ$. Total length of k-DoF arm $l = 2$. Initial position: $\theta_{i,1} = 0$ for all $i = 1, ..., k$. Target position: $(l\cos 170^\circ, l\sin 170^\circ)$. Number of features $M = 161$.

2) MPC Raw Performance Comparison: We compare the raw performance of MPC prediction between SCP-MPC and SPMPC. The dynamics models are pre-trained with various of pre-collected random sample sets, with sample size of $[50, 100, 500, 1000, 2000]$. For each training sample size, we run $N_{trial} = 10$ trials.

a) Computational Cost: In both system dynamics training and MPC control sequence optimizing, time consumption of the SCP-MPC stays semi-constant regardless of the training sample size, Fig. 2. Oppositely, the SPMPC severely suffers from the larger training sample size.

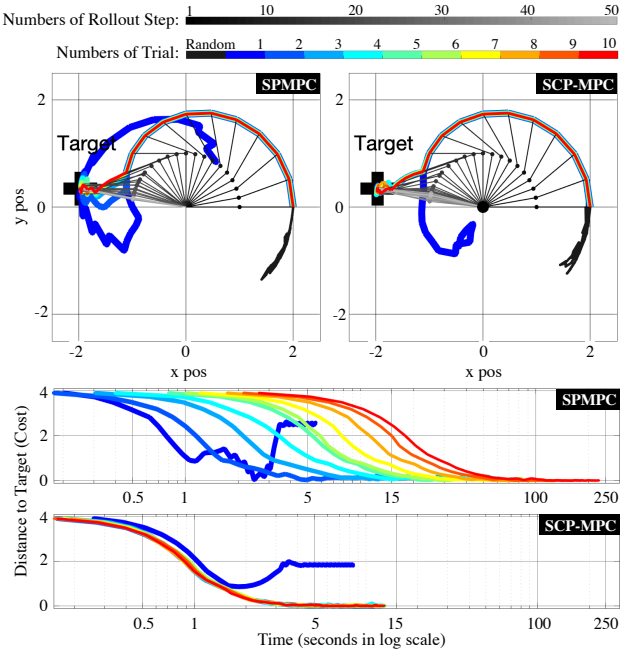


Fig. 4: (Upper) Path of each trial of RL with MPC and initial random sample generating trial (black path). (Lower) Distance to target regarding time consumption. Colors represent each trial. Gray-scale represent postures in the final trial.

b) MPC performance comparison: Focusing on the performance of accomplishing assigned task between SCP-MPC and SPMPC with various sample sizes pre-trained dynamics model. The the SCP-MPC results in comparative performance to SPMPC, but outperformed in computational cost, Fig. 3.

3) RL Performance Comparison: Both SCP-MPC and SPMPC follow Algorithm 1, to learn the target reaching task from scratch. Figure. 4 shows that both systems starts to converge since the second trial, where SCP-MPC out-stands in semi-constant time consumption. The result shows that, in simulation, the SCP-MPC results in a similar behavior to the SPMPC, but with a much lower computational cost.

B. Real Robot Experiments

In this section, we apply SCP-MPC to the real-time tracking moving target by RL with high-frequency MPC task on a real robot.

1) Environmental Setup: We set up a two-DoF robot arm identical to simulation environment with HEBI robotics hardware, Fig.5: two "X8-9" actuators and two 0.325m "X5" links, arm length $l = 0.65$m. We set 0.2 seconds fixed control period, real-time tracking a pattern 8 moving target with position $(0.9l\cos(2\pi t/T),\ 0.6l\sin(4\pi t/T)$, at time t in seconds. The cycle period is set to $T = 30$ seconds with $L_{rollout} = T/0.2$ for a full cycle target tracking. The

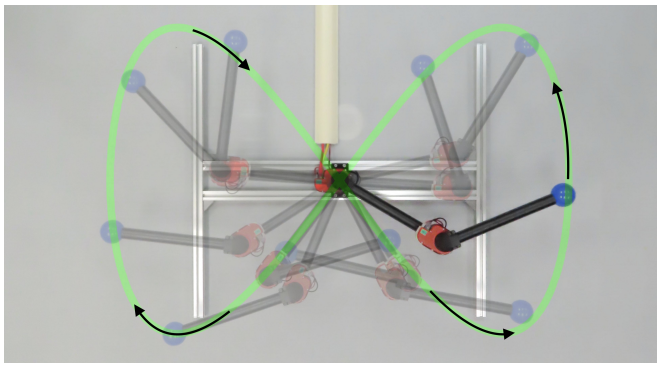


Fig. 5: HEBI robotics two-DoF arm setup with translucent arm tracking trajectory. Green path shows target trajectory.

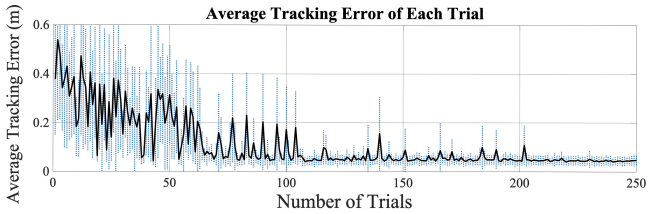


Fig. 6: Average tracking error of HEBI two-DoF arm in each trail. Blue zone represents standard deviation.

MPC objective is to plan a control sequence that minimize the distance between arm tip and the moving target. We define the state of the system as $\mathbf{x}_t = \left[\sin(\Theta_t), \cos(\Theta_t), \dot{\Theta}_t\right]$ where $\Theta_t = \left[\theta_{1,t}, \theta_{2,t}\right]^\top$, with corresponding discrete time angular acceleration control signal $\mathbf{u}_t = \left[\Delta\mathbf{u}_{1,t}, \Delta\mathbf{u}_{2,t}\right]^\top \in \left[-10, 10\right]^\circ/s$. Experiment variables as followed: Initial position $\theta_{1,t_0}, \theta_{2,t_0} = 0$, MPC predict horizon $H = 3$, number of features $M = 73$ and total trial $N_{trial} = 250$.

2) SCP-MPC Real-time Tracking: We define the tracking error as the distance between the arm tip and target position at each time step. At each time step, the MPC will observe the target position and plan an optimal control sequence to minimize the tracking error. The target stays moving during the MPC planning process; therefore, higher frequency control period returns better tracking performance, but sample collected per seconds also increases. The mean and standard deviation of the tracking error, Fig. 6 and corresponding tracking trajectory, Fig.7, shows the capability of SCP-MPC with high-frequency MPC control in the real world. At all 250^{th} trails, the SCP-MPC consistently maintains 0.2 seconds constant control period handling up to 37500 samples; this shows SCP-MPC is also capable of learning high complexity models which require a large training sample size to express its' dynamics.

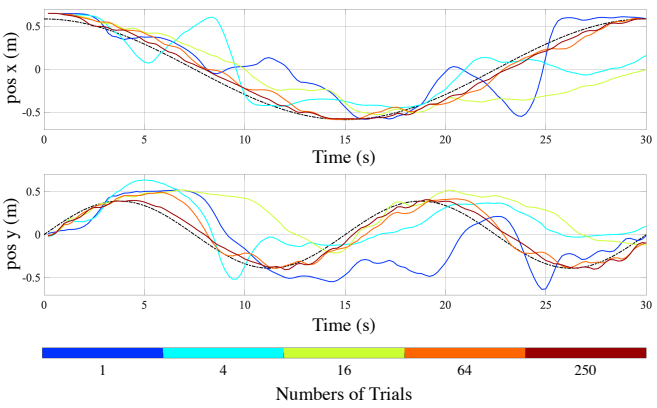


Fig. 7: Tracking position of x and y axis at each trails. Target trajectory in black.

V. Discussions and Future Work

A higher frequency control increases the collected sample per second but reduces the difference between each training input and target; therefore, it requires more samples to puzzle up the full system dynamics which will severely impact the computation efficiency in conventional MPC approaches [4, 5, 7, 16]. This work presents SCP-MPC which introduces MPC to high-frequency control scenarios based on its' sample-and-computation-efficient characteristic.

For the future work, we will compare the performance with a similar approach [17], where the author samples the features from RKS kernel expansion. Also, we will explore a multi-layer cascade SCP-MPC or expand SCP-MPC to higher complexity scenarios which require a large number of samples to express its' full dynamics, for example, legged robot locomotion or high-DoF robots.

Appendix

Math detail of Eq.(24) as follows

$$\begin{aligned}\mathbf{CC} &= n^{-1}\mathbb{E}_{\mathbf{x}_t}[\cos(\mathbf{V}\begin{bmatrix}\mathbf{x}_t\\ \mathbf{u}_t\end{bmatrix})\cos(\mathbf{V}\begin{bmatrix}\mathbf{x}_t\\ \mathbf{u}_t\end{bmatrix})^\top|\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t] \\ &= n^{-1}(\mathbf{G}\circ\cos(\mathbf{V}^{\text{sub}}) + \mathbf{H}\circ\cos(\mathbf{V}^{\text{add}})),\end{aligned} \tag{28}$$

$$\begin{aligned}\mathbf{CS} &= n^{-1}\mathbb{E}_{\mathbf{x}_t}[\cos(\mathbf{V}\begin{bmatrix}\mathbf{x}_t\\ \mathbf{u}_t\end{bmatrix})\sin(\mathbf{V}\begin{bmatrix}\mathbf{x}_t\\ \mathbf{u}_t\end{bmatrix})^\top|\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t] \\ &= n^{-1}(-\mathbf{G}\circ\sin(\mathbf{V}^{\text{sub}}) + \mathbf{H}\circ\sin(\mathbf{V}^{\text{add}})),\end{aligned} \tag{29}$$

$$\begin{aligned}\mathbf{SS} &= n^{-1}\mathbb{E}_{\mathbf{x}_t}[\sin(\mathbf{V}\begin{bmatrix}\mathbf{x}_t\\ \mathbf{u}_t\end{bmatrix})\sin(\mathbf{V}\begin{bmatrix}\mathbf{x}_t\\ \mathbf{u}_t\end{bmatrix})^\top|\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t] \\ &= n^{-1}(\mathbf{G}\circ\cos(\mathbf{V}^{\text{sub}}) - \mathbf{H}\circ\cos(\mathbf{V}^{\text{add}})),\end{aligned} \tag{30}$$

with each element of $\mathbf{G}, \mathbf{H}, \mathbf{V}^{\text{sub}}, \mathbf{V}^{\text{add}} \in \mathbb{R}^{n\times n}$ as follow

$$G_{ij} = \exp(-0.5([\mathbf{V}_\mathbf{x}\boldsymbol{\Sigma}_t]_i - [\mathbf{V}_\mathbf{x}\boldsymbol{\Sigma}_t]_j)^2), \tag{31}$$

$$H_{ij} = \exp(-0.5([\mathbf{V}_\mathbf{x}\boldsymbol{\Sigma}_t]_i + [\mathbf{V}_\mathbf{x}\boldsymbol{\Sigma}_t]_j)^2), \tag{32}$$

$$V^{\text{sub}}_{ij} = [\mathbf{V}[\boldsymbol{\mu}_t, \mathbf{u}_t]^\top]_i - \left[\mathbf{V}[\boldsymbol{\mu}_t, \mathbf{u}_t]^\top\right]_j, \tag{33}$$

$$V^{\text{add}}_{ij} = [\mathbf{V}[\boldsymbol{\mu}_t, \mathbf{u}_t]^\top]_i + \left[\mathbf{V}[\boldsymbol{\mu}_t, \mathbf{u}_t]^\top\right]_j. \tag{34}$$

References

[1] A. S. Polydoros and L. Nalpantidis, "Survey of model-based reinforcement learning: Applications on robotics," *Journal of Intelligent & Robotic Systems*, vol. 86, no. 2, pp. 153–173, 2017.

[2] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2006.

[3] M. P. Deisenroth and C. E. Rasmussen, "PILCO: A Model-based and Data-efficient Approach to Policy Search," in *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pp. 465–472, 2011.
[4] G. Cao, E. M. Lai, and F. Alam, "Gaussian Process Model Predictive Control of an Unmanned Quadrotor," *Journal of Intelligent and Robotic Systems*, vol. 88, no. 1, pp. 147–162, 2017.
[5] S. Kamthe and M. P. Deisenroth, "Data-Efficient Reinforcement Learning with Probabilistic Model Predictive Control," in *International Conference on Artificial Intelligence and Statistics*, vol. 84, pp. 1701–1710, 2018.
[6] L. Hewing and M. N. Zeilinger, "Cautious Model Predictive Control using Gaussian Process Regression," *CoRR*, vol. abs/1705.10702, 2017.
[7] Y. Cui, S. Osaki, and T. Matsubara, "Reinforcement Learning Boat Autopilot: Sample-efficient and Model Predictive Control-based Approach," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2868–2875, 2019.
[8] M. P. Deisenroth, M. F. Huber, and U. D. Hanebeck, "Analytic Moment-based Gaussian Process Filtering," in *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 225–232, 2009.
[9] M. P. Deisenroth, *Efficient reinforcement learning using Gaussian processes*. KIT Scientific Publishing, 2010.
[10] A. Rahimi and B. Recht, "Random Features for Large-Scale Kernel Machines," in *Proceedings of the Advances in Neural Information Processing Systems 20*, pp. 1177–1184, 2008.
[11] Q. Le, T. Sarlós, and A. Smola, "Fastfood: Approximating Kernel Expansions in Loglinear Time," in *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, pp. 244–252, 2013.
[12] C. K. I. Williams, "Prediction with Gaussian Processes: From Linear Regression to Linear Prediction and Beyond," in *Learning in Graphical Models*, pp. 599–621, Springer Netherlands, 1998.
[13] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.
[14] J. Nocedal, "Sequential Quadratic Programming," in *Numerical Optimization*, pp. 529–562, Springer New York, 2006.
[15] Z. Yang, A. Wilson, A. Smola, and L. Song, "A la Carte – Learning Fast Kernels," in *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, pp. 1098–1106, 2015.
[16] A. S. K. Annamalai, R. Sutton, C. Yang, P. Culverhouse, and S. Sharma, "Robust Adaptive Control of an Uninhabited Surface Vehicle," *Journal of Intelligent & Robotic Systems*, vol. 78, no. 2, pp. 319–338, 2015.
[17] Y. Pan, X. Yan, E. A. Theodorou, and B. Boots, "Prediction under Uncertainty in Sparse Spectrum Gaussian Processes with Applications to Filtering and Control," in *Proceedings of the 34th International Conference on Machine Learning*, vol. 70, pp. 2760–2768, 2017.