Data-Driven Differential Dynamic Programming Using Gaussian Processes

Yunpeng Pan and Evangelos Theodorou

***Abstract*—We present a data-driven trajectory optimization framework for general unknown systems based on Gaussian processes (GPs), called Gaussian progresses differential dynamic programming (GPDDP). Rooted in the Dynamic Programming(DP) principle and second-order local approximations of the value function, GPDDP learns time-varying local optimal controls from sampled data. Based on this framework, we propose two algorithms for implementation. We demonstrate the effectiveness and efficiency of the proposed framework using three numerical examples.**

I. Introduction

Differential dynamic programming (DDP) is a powerful trajectory optimization approach. Originally introduced in [4], DDP generates both open and closed local optimal control policies along with an optimal state trajectory. Compared to global optimal control approaches, the local optimal DDP shows superior applicability to high-dimensional problems. In the last decade, variations of DDP have been proposed in both control and machine learning communities [11][8][10]. A recent study demonstrated the applicability of DDP to high-dimensional robotic systems [9].

The principle limitation of DDP is that it relies on accurate models of dynamical systems. Modeling a complex dynamical system is generally a challenging task because of system's uncertainty and stochasticity. Various approaches have been developed to address these issues. In [6], a minimax formulation of DDP was employed to correct modeling errors and unknown disturbances. In [1], a complex system was parametrized and learned from data, and DDP was applied on the parametric model. Despite the broad applicability of DDP on either parametrically learned or a-priori known dynamics, there are case where the interdependencies between various elements (electromechanical, pneumatic, hydraulic) of autonomous systems are far from being easily represented by a parametric models. To overcome this limitation, nonparametric models, such as Gaussian processes (GPs), have gained more and more attention in a theoretical and applications level.

GP is a Bayesian nonparametric approach and data-driven, generative model. The model uncertainty can be quantified due to its explicit probabilistic formulation. GP regression operates in the function space and is completely defined by a mean and a covariance function. GPs have demonstrated promising applicability in modeling complex dynamical systems [5][3][7]. In this paper, we use GP to represent unknown dynamics, and incorporate GP inference into the DDP framework. Compared to DDP/iLQG [11], which generally perform linearization of a given dynamics using finite differences, GPDDP computes all partial derivatives analytically. For higher dimensional systems, the analytic computation is significantly more efficient than finite differences. Furthermore, we develop both online and offline algorithms for different types of control tasks based on GPDDP.

The rest of this paper is organized as follows: In Section II, we introduce the GP representation of dynamics. Section III shows the explicit form of optimal control. In Section IV, we propose two algorithms based on GPDDP and discuss the computational complexity of the framework. Numerical results are provided and discussed in Section V. Finally Section VI concludes this paper.

II. Dynamical model learning using Gaussian Processes

We consider a general unknown nonlinear system described by the following differential equation

$$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, \mathbf{u})\mathrm{d}t + \mathbf{C}(\mathbf{x}, \mathbf{u})\mathrm{d}\omega, \quad \mathbf{x}(t_0) = \mathbf{x}_0, \quad (1)$$

The authors are with The Daniel Guggenheim School of Aerospace Engineering, Georgia Institute of Technology, 270 Ferst Drive Atlanta GA 30332-0150, USA ypan37@gatech.edu etheodorou3@mail.gatech.edu

with state $\mathbf{x} \in \mathbb{R}^n$, control $\mathbf{u} \in \mathbb{R}^m$, and standard Brownian motion noise $\omega \in \mathbb{R}^p$ such that $\mathrm{d}\omega \sim \mathcal{N}(0, \mathbf{\Sigma}_\omega)$. The continuous functional mapping from state-control pair $\tilde{\mathbf{x}} = (\mathbf{x}, \mathbf{u}) \in \mathbb{R}^{n+m}$ to state transition $\mathrm{d}\mathbf{x}$ can be viewed as an inference with the goal of inferring state transition $\mathrm{d}\mathbf{x}$ given $(\mathbf{x}, \mathbf{u})$. We view this inference as a nonlinear regression problem. In this section, we introduce the Gaussian processes (GP) approach to learning dynamical model (1).

A GP is defined as a collection of random variables, any finite number subset of which have a joint Gaussian distribution. A GP defines a zero-mean Gaussian prior distribution over outputs. Given a sequence of state-control pair $\tilde{\mathbf{X}} = \{(\mathbf{x}_1, \mathbf{u}_1), \dots (\mathbf{x}_N, \mathbf{u}_N)\}$, and the corresponding state transition $\mathrm{d}\mathbf{X} = \{\mathrm{d}\mathbf{x}_1, \dots, \mathrm{d}\mathbf{x}_N\}$, a GP is completely defined by a mean function and a covariance function. The joint distribution of the observed output and the output corresponding to a given test state-control pair $\tilde{\mathbf{X}}^* = (\mathbf{x}^*, \mathbf{u}^*)$ can be written as

$$\mathrm{p}\begin{pmatrix} \mathrm{d}\mathbf{X} \\ \mathrm{d}\mathbf{x}^* \end{pmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} \mathbf{K}(\tilde{\mathbf{X}}, \tilde{\mathbf{X}}) + \sigma_n \mathbf{I} & \mathbf{K}(\tilde{\mathbf{X}}, \tilde{\mathbf{x}}^*) \\ \mathbf{K}(\tilde{\mathbf{x}}^*, \tilde{\mathbf{X}}) & \mathbf{K}(\tilde{\mathbf{x}}^*, \tilde{\mathbf{x}}^*) \end{bmatrix}\right).$$

The covariance of this multivariate Gaussian distribution is defined via a kernel matrix $\mathbf{K}(\mathbf{x}_i, \mathbf{x}_j)$. In particular, in this paper we consider the Gaussian kernel, which is most widely used kernel function

$$\mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) = \sigma_s^2 \exp(-\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_j)^{\mathrm{T}} \mathbf{W} (\mathbf{x}_i - \mathbf{x}_j)) + \sigma_n^2, \tag{2}$$

with $\sigma_s, \sigma_n, \mathbf{W}$ the hyper-parameters of the GP. The kernel function can be interpreted as a similarity measure of random variables. More specifically, if the training pairs $\tilde{\mathbf{X}}_i$ and $\tilde{\mathbf{X}}_j$ are close to each other in the kernel space, their outputs $\mathrm{d}\mathbf{x}_i$ and $\mathrm{d}\mathbf{x}_j$ are highly correlated. The posterior distribution, which is also a Gaussian distribution, can be obtained by constraining the joint distribution to contain the output $\mathrm{d}\mathbf{x}^*$ that is consistent with the observations. Assuming independent outputs (no correlation between each output dimension) and given test input $\tilde{\mathbf{x}}_t = [\mathbf{x}_t, \mathbf{u}_t]$ at time t the one-step prediction of the dynamics based on GP can be evaluated as $\mathrm{p}(\mathrm{d}\mathbf{x}_t | \tilde{\mathbf{x}}_t) \sim \mathcal{N}(\mathrm{d}\boldsymbol{\mu}_t, \mathrm{d}\mathbf{\Sigma}_t)$, where the terms $\mathrm{d}\boldsymbol{\mu}_t$ and $\mathrm{d}\mathbf{\Sigma}_t$ are the predictive mean and variance of the state transition and they are specified as

$$\begin{aligned} \mathrm{d}\boldsymbol{\mu}_t &= \mathbb{E}[\mathrm{d}\mathbf{x}_t] = \mathbf{K}(\tilde{\mathbf{x}}_t, \tilde{\mathbf{X}})(\mathbf{K}(\tilde{\mathbf{X}}, \tilde{\mathbf{X}}) + \sigma_n \mathbf{I})^{-1} \mathrm{d}\mathbf{X}, \\ \mathrm{d}\mathbf{\Sigma}_t &= \mathrm{Var}[\mathrm{d}\mathbf{x}_t] = \mathbf{K}(\tilde{\mathbf{x}}_t, \tilde{\mathbf{x}}_t) - \mathbf{K}(\tilde{\mathbf{x}}_t, \tilde{\mathbf{X}})(\mathbf{K}(\tilde{\mathbf{X}}, \tilde{\mathbf{X}}) \\ &\quad + \sigma_n \mathbf{I})^{-1} \mathbf{K}(\tilde{\mathbf{X}}, \tilde{\mathbf{x}}_t). \end{aligned} \tag{3}$$

A toy example of GP regression is shown in Fig. 1.

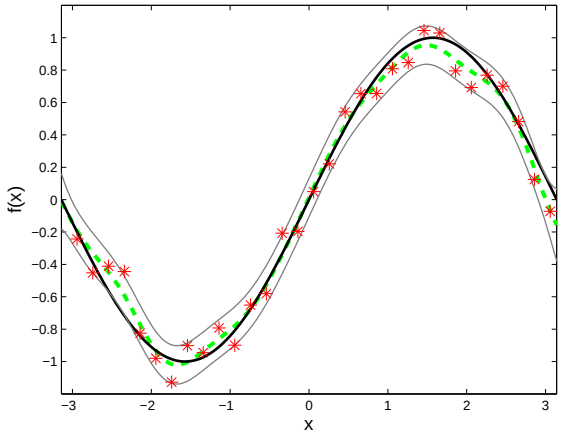


Fig. 1: Simple one-dimensional GP regression, where $\mathbf{f}(\mathbf{x}) = \sin \mathbf{x}$. Red stars are noisy samples drawn from $\mathbf{f}(\mathbf{x})$; the black line is the function $\mathbf{f}(\mathbf{x})$; green dash line is the GP predictive mean, and gray lines are GP predictive variance.

The state distribution at $t = 1$ is $\mathrm{p}(\mathbf{x}_1) \sim \mathcal{N}(\boldsymbol{\mu}_1, \mathbf{\Sigma}_1)$ where the state mean an variance are: $\boldsymbol{\mu}_1 = \mathbf{x}_0 + \mathrm{d}\boldsymbol{\mu}_0, \mathbf{\Sigma}_1 = \mathrm{d}\mathbf{\Sigma}_0$. When propagating the GP-based dynamics over a trajectory of time horizon H, the input state $\mathbf{x}_t$ becomes uncertain with Gaussian distribution, where $t = 1, ..., H$ (the initial state $\mathbf{x}_0$ is deterministic). Thus the distribution over state transition can be computed as

$$\mathrm{p}(\mathrm{d}\mathbf{x}_t | \tilde{\mathbf{x}}_t) = \int \int \mathrm{p}(\mathrm{d}\mathbf{x}_t | \mathbf{x}_t, \mathbf{u}_t) \mathrm{p}(\mathbf{x}_t, \mathbf{u}_t) \mathrm{d}\mathbf{x}_t \mathrm{d}\mathbf{u}_t.$$

Generally, the above integration cannot be computed analytically because the nonlinear mapping of input Gaussian distributions lead to non-Gaussian predictive distributions. However, the predictive distribution can be approximated by a Gaussian in (4). Thus the state distribution at $t+1$ is also a Gaussian

$$\begin{aligned} \boldsymbol{\mu}_{t+1} &= \boldsymbol{\mu}_t + \mathrm{d}\boldsymbol{\mu}_t, \\ \mathbf{\Sigma}_{t+1} &= \mathbf{\Sigma}_t + \mathrm{d}\mathbf{\Sigma}_t + \mathrm{Cov}[\mathbf{x}_t, \mathrm{d}\mathbf{x}_t] + \mathrm{Cov}[\mathrm{d}\mathbf{x}_t, \mathbf{x}_t]. \end{aligned} \tag{4}$$

In order to obtain the distribution over state $\mathcal{N}(\boldsymbol{\mu}_{t+1}, \boldsymbol{\Sigma}_{t+1})$ we compute the joint distribution over state-control pair $\mathrm{p}(\tilde{\mathbf{x}}_t) = \mathrm{p}(\mathbf{x}_t, \mathbf{u}_t)$ as follows

$$\mathrm{p}\begin{pmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{pmatrix} \sim \mathcal{N}\left(\begin{matrix} \boldsymbol{\mu}_t \\ \mathbb{E}[\mathbf{u}_t] \end{matrix}, \begin{bmatrix} \boldsymbol{\Sigma}_t & \mathrm{Cov}[\mathbf{x}_t, \mathbf{u}_t] \\ \mathrm{Cov}[\mathbf{u}_t, \mathbf{x}_t] & \mathrm{Cov}[\mathbf{u}_t] \end{bmatrix} \right), \tag{5}$$

where $\mathbb{E}[\mathbf{u}_t]$ and $\mathrm{Cov}[\mathbf{u}_t]$ are mean and covariance of the distribution over control policy $\mathrm{p}(\mathbf{u}_t)$. To simplify notation, we denote the mean and covariance of the above distribution as $\mathrm{p}(\tilde{\mathbf{x}}_t) \sim \mathcal{N}(\tilde{\boldsymbol{\mu}}_t, \tilde{\boldsymbol{\Sigma}}_t)$. Since the control policy is a linear function of the state in this paper, this joint distribution is a Gaussian. Given the input joint distribution $\mathrm{p}(\tilde{\mathbf{x}}_t)$, we will compute the predictive distribution of state transition $\mathrm{p}(\mathrm{d}\mathbf{x}_t|\tilde{\mathbf{x}}_t)$. The predictive mean $\mathbb{E}(\mathrm{d}\mathbf{x}_t|\tilde{\mathbf{x}}_t)$ can be computed using the law of iterated expectations

$$\begin{aligned} \mathbb{E}(\mathrm{d}\mathbf{x}_t|\tilde{\mathbf{x}}_t) &= \int\int\int \mathbf{f}(\mathbf{x}_t, \mathbf{u}_t)\mathrm{p}(\mathbf{x}_t, \mathbf{u}_t)\mathrm{d}\mathbf{f}\mathrm{d}\mathbf{x}_t\mathrm{d}\mathbf{u}_t\mathrm{d}t \\ &= \underbrace{\left(\mathbf{K}(\tilde{\mathbf{X}}, \tilde{\mathbf{X}}) + \sigma_n^2\mathbf{I})^{-1}\mathrm{d}\mathbf{X}\right)^{\mathrm{T}}\mathrm{d}t}_{\Psi} \\ &\quad \underbrace{\int \mathbf{K}(\tilde{\mathbf{X}}, \tilde{\mathbf{x}}_t)\mathcal{N}\big(\tilde{\mathbf{x}}_t|\tilde{\boldsymbol{\mu}}_t, \tilde{\boldsymbol{\Sigma}}_t\big)\mathrm{d}\tilde{\mathbf{x}}_t}_{\mathbf{q}}, \end{aligned} \tag{6}$$

where $\Psi \in \mathbb{R}^{N\times n}$ and $\mathbf{q}_t = [q_{t1}, \ldots, q_{tN}]^{\mathrm{T}} \in \mathbb{R}^N$. Next, we compute the predictive covariance matrix

$$\mathrm{Cov}\big(\mathrm{d}\mathbf{x}_t|\tilde{\mathbf{x}}_t\big) = \begin{bmatrix} \mathrm{Var}(\mathrm{d}\mathbf{x}_{t1}) & \ldots & \mathrm{Cov}(\mathrm{d}\mathbf{x}_{tn}, \mathrm{d}\mathbf{x}_{t1}) \\ \vdots & \ddots & \vdots \\ \mathrm{Cov}(\mathrm{d}\mathbf{x}_{t1}, \mathrm{d}\mathbf{x}_{tn}) & \ldots & \mathrm{Var}(\mathrm{d}\mathbf{x}_{tn}) \end{bmatrix} \tag{7}$$

where the variance terms can be obtained as

$$\begin{aligned} &\mathrm{Var}[\mathrm{d}\mathbf{x}_t] = \\ &\mathbb{E}_{\tilde{\mathbf{x}}_t}\big[\mathrm{Var}(\mathbf{f}(\tilde{\mathbf{x}}_t)|\tilde{\boldsymbol{\mu}}_t, \tilde{\boldsymbol{\Sigma}}_t)\big] + \mathrm{Var}\Big(\mathbb{E}_{\mathbf{f}}\big[\mathbf{f}(\tilde{\mathbf{x}}_t)|\tilde{\boldsymbol{\mu}}_t, \tilde{\boldsymbol{\Sigma}}_t\big]\Big) \\ &= \sigma_s^2 - \big(\Psi^{\mathrm{T}}\mathbf{q}_t\big)^2 - \mathrm{tr}\Bigg(\big(\mathbf{K}(\tilde{\mathbf{X}}, \tilde{\mathbf{X}}) + \sigma_n^2\mathbf{I}\big)^{-1} \\ &\int \Big(\mathbf{K}(\tilde{\mathbf{X}}, \tilde{\mathbf{x}}_t)\mathbf{K}(\tilde{\mathbf{x}}_t, \tilde{\mathbf{X}})\Big)\mathrm{p}(\tilde{\mathbf{x}}_t)\mathrm{d}\tilde{\mathbf{x}}_t\Bigg) + \Psi^{\mathrm{T}}\Phi_t\Psi \\ &\Phi_t = \int \mathbf{K}(\tilde{\mathbf{X}}, \tilde{\mathbf{x}}_t)\mathbf{K}(\tilde{\mathbf{x}}_t, \tilde{\mathbf{X}})\mathrm{p}(\tilde{\mathbf{x}}_t)\mathrm{d}\tilde{\mathbf{x}}_t \end{aligned} \tag{8}$$

The cross covariance terms can be obtained by

$$\begin{aligned} &\mathrm{Cov}(\mathrm{d}\mathbf{x}_{ti}, \mathrm{d}\mathbf{x}_{tj}) = \\ &\mathbb{E}_{\tilde{\mathbf{x}}_t}\big[\mathrm{d}\mathbf{x}_{ti}\mathrm{d}\mathbf{x}_{tj}\big] - \mathbb{E}_{\tilde{\mathbf{x}}_t}\big[\mathrm{d}\mathbf{x}_{ti}\big]\mathbb{E}_{\tilde{\mathbf{x}}_t}\big[\mathrm{d}\mathbf{x}_{tj}\big], \end{aligned} \tag{9}$$

Similarly, it can be found that the first term in (9) is

$$\mathbb{E}_{\tilde{\mathbf{x}}_t}\big[\mathrm{d}\mathbf{x}_{ti}\mathrm{d}\mathbf{x}_{tj}\big] = \Psi_i^{\mathrm{T}}\Phi_t\Psi_j, \tag{10}$$

$$\text{with } \Psi_i = \mathbf{K}(\tilde{\mathbf{X}}, \tilde{\mathbf{X}}) + \sigma_n^2\mathbf{I})^{-1}\mathrm{d}\mathbf{X}_i, \ \forall i = 1, ...n. \tag{11}$$

Therefore

$$\mathrm{Cov}[\mathrm{d}\mathbf{x}_{ti}, \mathrm{d}\mathbf{x}_{tj}] = \Psi_i^{\mathrm{T}}\Phi_t\Psi_j - (\Psi_i^{\mathrm{T}}\mathbf{q}_t)^{\mathrm{T}}(\Psi_j^{\mathrm{T}}\mathbf{q}_t). \tag{12}$$

The input-output cross-covariances can be obtained by

$$\begin{aligned} \mathrm{Cov}[\mathbf{x}_t, \mathrm{d}\mathbf{x}_t] &= \mathbb{E}[\mathbf{x}_t\mathrm{d}\mathbf{x}_t] - \mathbb{E}[\mathbf{x}_t]\mathbb{E}[\mathrm{d}\mathbf{x}_t] \\ &= \mathbb{E}[\mathbf{x}_t\mathbf{f}(\mathbf{x}_t, \mathbf{u}_t)] - \boldsymbol{\mu}_t\tilde{\boldsymbol{\mu}}_t. \end{aligned} \tag{13}$$

The kernel or hyper-parameter $\Theta = (\sigma_n, \sigma_s, \mathbf{W})$ can be learned by maximizing the log-likelihood of the training outputs given the inputs

$$\Theta^* = \underset{\Theta}{\mathrm{argmax}}\left\{\log\left(\mathrm{p}\Big(\mathrm{d}\mathbf{X}|\tilde{\mathbf{X}}, \Theta\Big)\right)\right\}, \tag{14}$$

where

$$\begin{aligned} &\log\left(\mathrm{p}\Big(\mathrm{d}\mathbf{X}|\tilde{\mathbf{X}}, \Theta\Big)\right) = -\frac{1}{2}\mathrm{d}\mathbf{X}^{\mathrm{T}}\Big(\mathbf{K}(\tilde{\mathbf{X}}, \tilde{\mathbf{X}}) + \sigma_n^2\mathbf{I}\Big)^{-1} \\ &\mathrm{d}\mathbf{X} - \frac{1}{2}\log\Big|\mathbf{K}(\tilde{\mathbf{X}}, \tilde{\mathbf{X}}) + \sigma_n^2\mathbf{I}\Big| - \frac{N}{2}\log 2\pi. \end{aligned} \tag{15}$$

The optimization problem can be solved using numerical methods such as conjugate gradient [12].

III. Gaussian Process Differential Dynamic Programming

We consider the optimal control problem with cost:

$$J^{\pi}(\mathbf{x}, t) = \mathbb{E}\left[h\Big(\mathbf{x}(t_f)\Big) + \int_{t_0}^{t_f} \mathcal{L}\Big(\mathbf{x}(t), \pi(\mathbf{x}, t), t\Big)dt\right] \tag{16}$$

where $h(\mathbf{x}(t_f))$ is the terminal cost, $\mathcal{L}(\mathbf{x}(t), \pi(\mathbf{x}, t), t)$ is the instantaneous cost rate which is a function of state $\mathbf{x}$ and control policy $\pi(\mathbf{x}(t), t)$. The cost $J^{\pi}(\mathbf{x}, t)$ is defined as the expectation accumulated over the time horizon $(t_0, \ldots, t_f)$ starting from the initial state $\mathbf{x}(t_0)$ to final state $\mathbf{x}(t_f)$.

A. Linearization of the GP-based dynamics

As discussed in the last section, the states of the dynamical systems can be approximated by the predictive mean of the GP regression $\boldsymbol{\mu}_t$ in (4). It is worth mentioning that the covariances have also been taken into account in the propagation of the dynamics (details in section II). Given a nominal trajectory $(\bar{\mathbf{x}}_t, \bar{\mathbf{u}}_t)$ of states and controls, we linearize the dynamics around $(\bar{\mathbf{x}}_t, \bar{\mathbf{u}}_t)$. Here we define the state and control deviations $\delta\mathbf{x}_t = \boldsymbol{\mu}_t - \bar{\mathbf{x}}_t, \delta\mathbf{u}_t = \mathbf{u}_t - \bar{\mathbf{u}}_t$ and compute the first-order expansion of the dynamics specified as

$$\mathrm{d}\delta\mathbf{x}_t = \mathbf{f}_{\mu_t}\delta\mathbf{x}_t\mathrm{d}t + \mathbf{f}_{\mathbf{u}_t}\delta\mathbf{u}_t\mathrm{d}t, \tag{17}$$

where $\mathbf{f}_{\mu_t} = \nabla_\mu \mathbf{f}_t$ and $\mathbf{f}_{\mathbf{u}_t} = \nabla_\mathbf{u}\mathbf{f}_t$ the gradients that can be computed as

$$\nabla_\mu \mathbf{f}_t = \frac{\partial \mathbb{E}(\mathrm{d}\mathbf{x}_t|\tilde{\mathbf{x}}_t)}{\partial \boldsymbol{\mu}_t} = \frac{\partial \Psi^{\mathrm{T}}\mathbf{q}_t}{\partial \boldsymbol{\mu}_t} = \frac{\partial \Psi^{\mathrm{T}}\mathbf{q}_t}{\partial \tilde{\boldsymbol{\mu}}_t}\frac{\partial \tilde{\boldsymbol{\mu}}_t}{\partial \boldsymbol{\mu}_t}. \tag{18}$$

Next we show the computation of $\frac{\partial \Psi^{\mathrm{T}}\mathbf{q}_t}{\partial \tilde{\boldsymbol{\mu}}_t}$. In particular for each output dimension j, the partial derivative can be computed analytically

$$\begin{aligned}\frac{\partial \Psi_i^{\mathrm{T}}\mathbf{q}_{ti}}{\partial \tilde{\boldsymbol{\mu}}_t} &= \sum_{j=1}^{N}\Psi_{ij}\frac{\partial \mathbf{q}_{tij}}{\partial \tilde{\boldsymbol{\mu}}_t} \\ &= \sum_{j=1}^{N}\Psi_{ij}\mathbf{q}_{tij}(\tilde{\mathbf{x}}_t - \tilde{\boldsymbol{\mu}}_t)^{\mathrm{T}}(\tilde{\boldsymbol{\Sigma}}_t + \mathbf{W}_j)^{-1}.\end{aligned}$$

Similarly we can obtain $\nabla_\mathbf{u}\mathbf{f}_t = \frac{\partial \Psi^{\mathrm{T}}\mathbf{q}_t}{\partial \tilde{\boldsymbol{\mu}}_t}\frac{\partial \tilde{\boldsymbol{\mu}}_t}{\partial \mathbf{u}_t}$. The partial derivatives $\frac{\partial \tilde{\boldsymbol{\mu}}_t}{\partial \boldsymbol{\mu}_t}$ and $\frac{\partial \tilde{\boldsymbol{\mu}}_t}{\partial \mathbf{u}_t}$ can be computed analytically as well. The linearized dynamics (17) can be rewritten in discrete form as follows,

$$\delta\mathbf{x}_{t+1} = \underbrace{(I_{n\times n} + \mathbf{f}_{\mu_t}\mathrm{d}t)}_{\mathbf{A}_t}\delta\mathbf{x}_t + \underbrace{(\mathbf{f}_{\mathbf{u}_t}\mathrm{d}t)}_{\mathbf{B}_t}\delta\mathbf{u}_t, \tag{19}$$

After specifying the forms of the state transition and control transition matrices $\mathbf{A}_t$ and $\mathbf{B}_t$, next we discuss the DDP algorithm based on first order expansion of dynamics.

B. Value function approximation

The Bellman equation for the cost-to-go function is defined as

$$V(\mathbf{x}_t, t) = \min_{\mathbf{u}_t}\mathbb{E}\Big[\mathcal{L}(\mathbf{x}_t, \mathbf{u}_t)\mathrm{d}t + V(\mathbf{x}_t, t)\Big|\mathbf{x}_t\Big]. \tag{20}$$

As in the classical DDP, we expand the cost-to-go function up to the second order around a given nominal trajectory $(\bar{\mathbf{x}},\bar{\mathbf{u}})$. First we expand the running cost:

$$\begin{aligned}\mathcal{L}(\mathbf{x}_t, \mathbf{u}_t)\mathrm{d}t &= \mathcal{L}\Big(\bar{\mathbf{x}}_t + \delta\mathbf{x}_t, \bar{\mathbf{u}}_t + \delta\mathbf{u}_t\Big)\mathrm{d}t \\ &= q_t + \mathbf{q}_t^{\mathrm{T}}\delta\mathbf{x}_t + \mathbf{r}_t^{\mathrm{T}}\delta\mathbf{u}_t \\ &+ \frac{1}{2}\delta\mathbf{x}_t^{\mathrm{T}}\mathbf{Q}_t\delta\mathbf{x}_t + \frac{1}{2}\delta\mathbf{u}_t^{\mathrm{T}}\mathbf{R}_t\delta\mathbf{u}_t \\ &+ \frac{1}{2}\delta\mathbf{u}_t^{\mathrm{T}}\mathbf{N}_t\delta\mathbf{x}_t + \frac{1}{2}\delta\mathbf{x}^{\mathrm{T}}\mathbf{M}_t\delta\mathbf{u}_t,\end{aligned}$$

where

$$\begin{aligned}&q_t \equiv \mathcal{L}(\mathbf{x}_t, \mathbf{u}_t)\mathrm{d}t, \mathbf{q}(t_k) \equiv \mathcal{L}_\mathbf{x}(\mathbf{x}_t, \mathbf{u}_t)\mathrm{d}t, \mathcal{L}_\mathbf{u}(\mathbf{x}_t, \mathbf{u}_t)\mathrm{d}t \\ &\mathbf{Q}_t \equiv \ \mathcal{L}_{\mathbf{xx}}(\mathbf{x}_t, \mathbf{u}_t)\mathrm{d}t, \quad \mathbf{R}_t \equiv \mathcal{L}_{\mathbf{uu}}(\mathbf{x}_t, \mathbf{u}_t)\mathrm{d}t, \\ &\mathbf{N}_t \equiv \mathcal{L}_{\mathbf{xu}}(\mathbf{x}_t, \mathbf{u}_t)\mathrm{d}t, \quad \mathbf{M}_t \equiv \mathcal{L}_{\mathbf{xu}}(\mathbf{x}_t, \mathbf{u}_t)\mathrm{d}t.\end{aligned} \tag{21}$$

Subscripts of $\mathcal{L}(\cdot)$ indicate derivatives, which can be computed analytically. We will use this subscript rule for $V(\cdot)$ as well. The detailed formulations are not given due to space limit. In addition the term $V(\mathbf{x}_t, t)$ is also expanded

$$\begin{aligned}V(\bar{\mathbf{x}}_{t+1} + \delta\mathbf{x}_{t+1}, t+1) &= V(\bar{\mathbf{x}}_{t+1}, t+1) + V_\mathbf{x}^{\mathrm{T}}\delta\mathbf{x}_{t+1} \\ &+ \frac{1}{2}\delta\mathbf{x}_{t+1}^{\mathrm{T}}V_{\mathbf{xx}}\delta\mathbf{x}_{t+1}.\end{aligned} \tag{22}$$

By taking the expectation we have:

$$\begin{aligned}&\mathbb{E}\Big[V(\bar{\mathbf{x}}_{t+1} + \delta\mathbf{x}_{t+1}, t+1)\Big] = V(\bar{\mathbf{x}}_{t+1}, t+1) \\ &+ V_{\mathbf{x}_{t+1}}^{\mathrm{T}}\mathbf{A}_t\delta\mathbf{x}_t + V_{\mathbf{x}_{t+1}}^{\mathrm{T}}\mathbf{B}_t\delta\mathbf{u}_t \\ &+ \frac{1}{2}\delta\mathbf{u}_t^{\mathrm{T}}\mathbf{B}_t^{\mathrm{T}}V_{\mathbf{x}\mathbf{x}_{t+1}}\mathbf{B}_t\delta\mathbf{u}_t + \frac{1}{2}\delta\mathbf{x}_t^{\mathrm{T}}\mathbf{A}_t^{\mathrm{T}}V_{\mathbf{x}\mathbf{x}_{t+1}}\mathbf{A}_t\delta\mathbf{x}_t \\ &+ \frac{1}{2}\delta\mathbf{u}_t^{\mathrm{T}}\mathbf{B}_t^{\mathrm{T}}V_{\mathbf{x}\mathbf{x}_{t+1}}\mathbf{A}_t\delta\mathbf{x}_t + \frac{1}{2}\delta\mathbf{x}_t^{\mathrm{T}}\mathbf{A}_t^{\mathrm{T}}V_{\mathbf{x}\mathbf{x}_{t+1}}\mathbf{B}_T\delta\mathbf{u}_t.\end{aligned}$$

C. Computing the local optimal control

To find the optimal control, we compute the local variations in control $\delta\mathbf{u}^*$ that maximize the value function

$$\begin{aligned}\delta\mathbf{u}_t^* =& \arg\max_{\delta\mathbf{u}}(\mathcal{L}(\mathbf{x}_t,\mathbf{u}_t)\mathrm{d}t + V(\mathbf{x}_{t+1},t+1)\\ =&\mathbf{l}_t + \mathbf{L}_t\delta\mathbf{x}_t,\end{aligned} \tag{23}$$

where

$$\mathbf{l}_t = -\mathbf{H}_t^{-1}\mathbf{g}_t, \qquad \mathbf{L}_t = -\mathbf{H}_t^{-1}\mathbf{G}_t. \tag{24}$$

The terms $\mathbf{g},\mathbf{G},\mathbf{H}$ are specified as

$$\begin{aligned}\mathbf{g}_t =&\mathbf{r}_t + \mathbf{B}_t^{\mathrm{T}}V_{\mathbf{x}_{t+1}}, \quad \mathbf{H} = \mathbf{R}_t + \mathbf{B}_t^{\mathrm{T}}V_{\mathbf{x}\mathbf{x}_{t+1}}\mathbf{B}_{t+1},\\ \mathbf{G}_t =&\frac{1}{2}\mathbf{N}_t + \frac{1}{2}\mathbf{M}_t + \mathbf{B}_t^{\mathrm{T}}V_{\mathbf{x}\mathbf{x}_{t+1}}\mathbf{A}_t.\end{aligned} \tag{25}$$

Therefore, the local optimal control $\mathbf{u}^*$ can be computed as

$$\mathbf{u}_t^* = \bar{\mathbf{u}}_t + \alpha\delta\mathbf{u}_t^*, \tag{26}$$

where α is a weighting factor. By plugging the optimal control back into the value function, splitting terms into zero, first and second order in $\delta\mathbf{x}$ we have:

$$V(\mathbf{x}+\delta\mathbf{x},t) = V_t + V_{\mathbf{x}_t}^{\mathrm{T}}\delta\mathbf{x} + \frac{1}{2}\delta\mathbf{x}^{\mathrm{T}}V_{\mathbf{x}\mathbf{x}_t}\delta\mathbf{x}, \tag{27}$$

where V_t, $V_{\mathbf{x}_t}$, $V_{\mathbf{x}\mathbf{x}_t}$ can be computed backward in time

$$\begin{aligned}V_t &= V_{t+1} + q_t + \mathbf{g}_t^{\mathrm{T}}\mathbf{l}_t + \frac{1}{2}\mathbf{l}_t^{\mathrm{T}}\mathbf{H}_t\mathbf{l}_t,\\ V_{\mathbf{x}_t} &= \mathbf{q}_t + \mathbf{L}_t^{\mathrm{T}}\mathbf{g}_t + \mathbf{L}_t^T\mathbf{H}_t\mathbf{l}_t + \mathbf{G}_t^{\mathrm{T}}\mathbf{l}_t + \mathbf{A}_t^{\mathrm{T}}V_{\mathbf{x}_{t+1}},\\ V_{\mathbf{x}\mathbf{x}_t} &= \mathbf{Q}_t + \mathbf{L}_t^T\mathbf{H}_t + \mathbf{L}_t^T\left(\mathbf{N}_t + \mathbf{B}_t^{\mathrm{T}}V_{\mathbf{x}\mathbf{x}_{t+1}}\mathbf{A}_t\right)\\ &+ \left(\mathbf{M}_t + \mathbf{A}_t^{\mathrm{T}}V_{\mathbf{x}\mathbf{x}_{t+1}}\mathbf{B}_t^{\mathrm{T}}\right)\mathbf{L}_t + \mathbf{A}_t^{\mathrm{T}}V_{\mathbf{x}\mathbf{x}_{t+1}}\mathbf{A}_t.\end{aligned} \tag{28}$$

Therefore, the second order approximation of the value function will be propagated backward in time iteratively. The optimal control sequence will be applied to generate a local optimal state trajectory forward in time. Next, we will discuss two algorithms based on the GP-DDP framework.

IV. Summary of Algorithms

In this section, we present two algorithms based on the GPDDP framework. The proposed algorithms are summarized in **Algorithm 1** and **2** . They are designed for different purposes. The online algorithm requires interaction with the physical system at every iteration. In particular once an local optimal control sequence is computed at every iteration, it is applied to the physical system to generate trajectories to learn the GP model. Therefore, in the online version a GP dynamic model is learned at every iteration, based on the trajectories/rollouts generated by applying the corresponding controller to the real dynamics.

In the off-line version, the GP model of the system dynamics is learned based on state and control trajectories collected initially. Then the GPDDP is applied to find the optimal control without interacting with the physical system. The offline algorithm collects training data as initialization and applies the optimal control sequence to the real system only after convergence. Therefore, the offline version of GP-DDP is suitable for tasks such as autonomous flight where forward propagating dynamics online is unsafe.

Both algorithms contain the following tasks:

Dynamical model learning: Given a nominal trajectory in states and controls, the nonlinear dynamics can be approximated locally by the linearized GP model as introduced in Section III-A. The approximated model is Bayesian nonparametric and takes into account uncertainties/stochaticities of the training data. In this paper, we use trajectory sampling to generate the training data that consist of pairs $(\{\mathbf{x}(t_k),\mathbf{u}(t_k)\},\{\mathbf{x}(t_{k+1}) - \mathbf{x}(t_k)\})$. The trajectories can be obtained by using variations of $\bar{\mathbf{u}}$ (sample from the stochastic dynamics) and with forward propagation.

Back-propagation: Compute the local optimal control law $\delta\mathbf{u}^*(t_k)$ as in (23) and cost to go function $V(\mathbf{x}+\delta\mathbf{x},t_k)$ as in (27) in a backward pass through time. The details about this computation have been shown in Section III-C.

Forward propagation: Apply the local optimal control obtained from back-propagation to the dynamics (either the stochastic dynamics or the GP approximation, depend on the choice of algorithm).

This step will give us a local optimal trajectory, which will be used as the new nominal trajectory to linearize the dynamics in the next iteration.

Given: Nominal trajectory, target states
Goal : Optimized trajectory

for $i = 1$ **to** I_{max} **do**
  Generate M trajectories by sampling from stochastic dynamics and forward propagate the dynamics (1);
  Obtain training pairs $(\{\mathbf{x}(t_k), \mathbf{u}(t_k)\}, \{\mathbf{x}(t_{k+1}) - \mathbf{x}(t_k)\})$;
  Compute kernel matrices and linearize the GP dynamics around $(\bar{\mathbf{x}}, \bar{\mathbf{u}})$ according to Section III-A;
  Backpropagate to get the local optimal control $\delta\mathbf{u}^*$ and value function $V(\mathbf{x} + \delta\mathbf{x}, t)$ according to (23) (27);
  Forward propagate (1) by applying the optimal control $\mathbf{u}_t^* = \bar{\mathbf{u}}_t + \alpha\delta\mathbf{u}_t^*$, obtain a new trajectory $(\mathbf{x}^*, \mathbf{u}^*)$;
  if *Converge* **then** Break the **for** loop;
  Set $\bar{\mathbf{x}} = \mathbf{x}^*$ and $\bar{\mathbf{u}} = \mathbf{u}^*$;
end

Algorithm 1: On-line GP-DDP

Given: A nominal trajectory
Goal : The optimized trajectory

Generate M trajectories by sampling from the stochastic dynamics and forward propagate the dynamics (1);
Obtain training pairs $(\{\mathbf{x}(t_k), \mathbf{u}(t_k)\}, \{\mathbf{x}(t_{k+1}) - \mathbf{x}(t_k)\})$;
Compute kernel matrices;
for $i = 1$ **to** I_{max} **do**
  linearize the GP dynamics around $(\bar{\mathbf{x}}, \bar{\mathbf{u}})$ according to Section III-A;
  Backpropagate in time to get the local optimal control $\delta\mathbf{u}^*$ and value function $V(\mathbf{x} + \delta\mathbf{x}, t_k)$ according to (23) (27);
  Forward propagate the GP dynamics (II) by applying the optimal control $\mathbf{u}_t^* = \bar{\mathbf{u}}_t + \alpha\delta\mathbf{u}_t^*$, obtain a new trajectory $(\mathbf{x}^*, \mathbf{u}^*)$;
  if *Converge* **then** Break the **for** loop;
  Set $\bar{\mathbf{x}} = \mathbf{x}^*$ and $\bar{\mathbf{u}} = \mathbf{u}^*$;
end
Apply the optimal control $\mathbf{u}^*$ to (1), obtain the optimized trajectory.

Algorithm 2: Off-line GP-DDP

A. Computational complexity

For DDP, the heaviest computational burden comes from linearization of the given dynamics up to the second order. Generally the analytic forms of partial derivatives involved in linearization are not available, which are usually approximated using finite differences. For GPDDP, we compute all Jacobian matrices analytically. For systems with higher dimensions, this linearization scheme is much more efficient than finite differences, see Fig. 4 for example. For GPDDP, the major part of computational effort is devoted to GP inferences. In particular, the complexity of computing the inverse of kernel matrix $\mathbf{K}(\tilde{\mathbf{X}}, \tilde{\mathbf{X}})^{-1}$ is $\mathcal{O}(N^3)$ where N is the number of the training data. The computational efficiency can be improved in various ways. For the online case, instead of generating new training data at each iteration, we may use sparse GP [2] to add/delete data to reduce computational burden. For the offline case, local GP model [7] may be employed to compute local kernel matrices with smaller size, and approximate the predictive mean of GP by the weighted mean of each local model.

V. Numerical Examples

In this section, we provide three numerical examples to evaluate the proposed GPDDP framework. For comparison purpose, we apply DDP to the six link robotic arm with a given dynamical model. We perform all linearizations in DDP using finite differences.

A. Inverted pendulum

The dynamics of inverted pendulum can be described as follow:

$$\mathbf{f}(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} \mathbf{x}_2 \\ \frac{1}{I}\left(mgl\sin(\mathbf{x}_1) - b\mathbf{x}_2 + \mathbf{u}\right) \end{bmatrix}, \quad (29)$$

and $\mathbf{g} = [0.5; 1]$. Where $\mathbf{x}_1$ is the position and $\mathbf{x}_2$ is the velocity. The task is to bring the pendulum from initial position $[\frac{\pi}{2}\ 0]$ to the inverted position

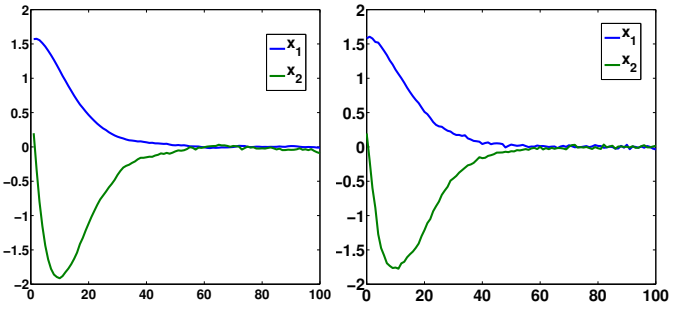
with zero velocity $[0\ 0]$. Both online and offline GP-DDP algorithms were applied to the system. For both methods, we sample 1 trajectories from (29). The results are shown in Fig. 2. Both algorithms are able to complete the task and perform similarly. It should be noticed that the offline algorithm requires more iterations to converge than the online one, since the online scheme feature more accurate approximation of the dynamics by sampling from (29) every iteration. By doing so, the online algorithm cost more computational effort at every iteration than the offline scheme.

B. Six-link robotic arm

The six-link arm model consist of six links of equal length and mass, connected in an open chain with resolute joints. The system has 6 degrees of freedom, and 12 state dimensions (angle and angular velocity for each joint). The goal for the first 3 joints is to move to $\frac{\pi}{4}$ and the rest 3 joints to $-\frac{\pi}{4}$. The desired velocities for all 6 joints are zeros. We apply the online GPDDP on this task. We sample 1 trajectory (M=1) from the true dynamics at each iteration. The optimized state trajectories are shown in Fig. 3 and a comparison is made in Fig. 4. As we mentioned in section IV-A, GPDDP linearize the learned GP model by computing Jacobian matrices analytically, which is significantly more efficient than finite-differences applied in DDP. GPDDP requires slightly more iterations than DDP to obtain the optimized trajectory since a local model is learned at each iteration.

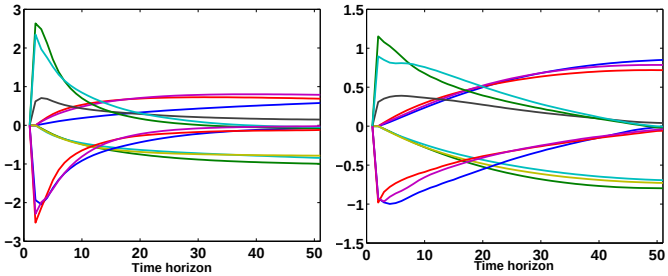
C. Helicopter hovering

The task of autonomous helicopter hovering is challenging because the helicopter systems are generally dynamically unstable. DDP has been used for autonomous helicopter tasks such as hovering and achieved state-of-the-art performances [1]. As most DDP applications, an accurate dynamical model of the helicopter system is essential. [1] assumes a parametrization of the helicopter dynamics and used flight data to estimate those parameters. The proposed GPDDP does not assume any parametrization of the dynamics and rely only on data. We applied the proposed GPDDP framework to the helicopter hovering task. The goal is to stabilize the helicopter from initial disturbances. Since sampling from dynamics online is not realistic, we use the offline scheme in this task. As in [1], the states can be described by positions $(\mathbf{x}, \mathbf{y}, \mathbf{z})$, velocities $(\mathbf{v}_x, \mathbf{v}_y, \mathbf{v}_z)$ and angular velocities $(\mathbf{w}_x, \mathbf{w}_y, \mathbf{w}_z)$. Control inputs are aileron, elevator, rudder, and collective pitch control, which are denoted by $\mathbf{u}_1$, $\mathbf{u}_2$, $\mathbf{u}_3$ and $\mathbf{u}_4$ respectively. The desired steady-states are all zeros. In our simulation, we generate trajectories from the parametric model (with stochastic terms) learned in [1] . Results are shown in Fig. 5. Notice that $\mathbf{u}_1, \mathbf{u}_2$ and $\mathbf{u}_3$ control the orientation of helicopter, $\mathbf{u}_4$ adjust the thrust of the main motor. Positive $\mathbf{u}_4$ force the main motor to blow air downward relative to the helicopter, and negative $\mathbf{u}_4$ blow air upward relative to the helicopter.



(a) (b)

Fig. 2: Optimized state trajectories for the inverted pendulum task. In all plots, the X-axis is the time horizon. (a) Online GPDDP. (b) Offline GPDDP.



(a) (b)

Fig. 3: Optimized state trajectories for six-link arm task. (a) DDP. (b) GPDDP.

VI. Conclusions

In this paper, we propose a data-driven differential dynamic programming (DDP) framework based

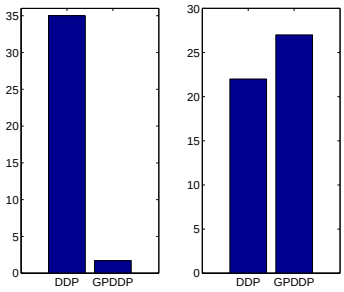
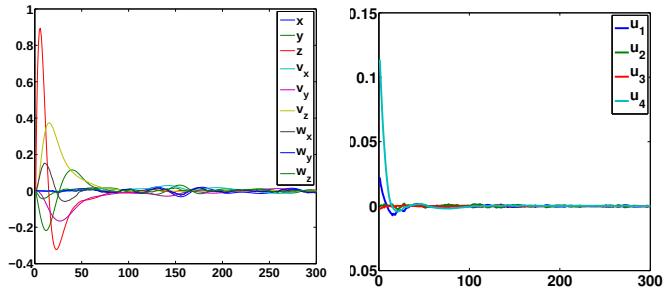


Fig. 4: Comparison of DDP and GPDDP for the six-link arm task. Left figure: time consumed at each iteration for linearization (sec). Right: number of iterations to obtain the optimized trajectory.



(a) (b)

Fig. 5: Simulation results for the helicopter hovering task. In all plots, the X-axis is the time horizon. (a) optimized state trajectories. (b) optimal controls.

on GPs, called GPDDP. The framework is model-based, and applicable to general unknown systems. Because of GP's Bayesian nonparametric nature, GPDDP takes into account model uncertainty and relies only on sampled data without any assumed structure of the system. Building on this framework, we also proposed online and offline algorithms for implementation. The framework was evaluated by three numerical examples.

Many interesting research avenues need in-depth exploration, which include utilizing the predictive variances for computing local optimal controls, and the improvement of the computational efficiency by using sparse GP, local model regression, as well as further applications to robotics and aerospace systems.

REFERENCES

[1] P. Abbeel, A. Coates, M. Quigley, and A. Y Ng. An application of reinforcement learning to aerobatic helicopter flight. *Advances in neural information processing systems*, 19:1, 2007.

[2] L. Csató and M. Opper. Sparse on-line gaussian processes. *Neural Computation*, 14(3): 641–668, 2002.

[3] M Deisenroth, D. Fox, and C Rasmussen. Gaussian processes for data-efficient learning in robotics and control. *IEEE Transsactions on Pattern Analysis and Machine Intelligence (PAMI)*, 27:75–90, 2014.

[4] D. Jacobson and D. Mayne. Differential dynamic programming. 1970.

[5] J. Ko and D. Fox. Gp-bayesfilters: Bayesian filtering using gaussian process prediction and observation models. *Autonomous Robots*, 27 (1):75–90, 2009.

[6] J. Morimoto, G. Zeglin, and C. G Atkeson. Minimax differential dynamic programming: Application to a biped walking robot. In *Intelligent Robots and Systems, 2003.(IROS 2003)*, volume 2, pages 1927–1932. IEEE, 2003.

[7] D. Nguyen-Tuong, M. Seeger, and J. Peters. Model learning with local gaussian process regression. *Advanced Robotics*, 23(15):2015–2034, 2009.

[8] Y. Tassa, T. Erez, and W. D. Smart. Receding horizon differential dynamic programming. In *NIPS*, 2007.

[9] Y. Tassa, N. Mansard, and E. Todorov. Control-limited differential dynamic programming. 2014.

[10] E. Theodorou, Y. Tassa, and E. Todorov. Stochastic differential dynamic programming. In *American Control Conference (ACC), 2010*, pages 1125–1132. IEEE, 2010.

[11] E. Todorov and W. Li. A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *American Control Conference*, pages 300–306, 2005.

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