Proximal Policy Optimization with MPC-based policy

Hien Bui
University of Pennsylvania
xuanhien@seas.upenn.edu

Haoxiang You
University of Pennsylvania
youhaox@seas.upenn.edu

Ye Duan
University of Pennsylvania
duanye@seas.upenn.edu

***Abstract*—Reinforcement learning, particularly model-free algorithms, has achieved remarkable success in complex locomotion and manipulation tasks in recent years. The control policies in these algorithms are often parameterized by neural networks. Thus, in order to successfully train those policies, a large number of interactions between the system and the environment are required, which may be costly or even impossible in real-world settings. The need for data grows substantially as the dynamics and tasks become more complex, such as in a hybrid system with contacts. In this work, we resolve that issue by imposing contact-dynamics-related structure to the control policy. More specifically, our method attempts to parameterize the policy as a MPC controller, with the dynamics model represented by a learnable Linear Complementarity System. We validate our method on an underactuated multi-contact system, and our results show that the amount of required data is roughly 15 times less than that of the state-of-the-art model-free algorithm, Proximal Policy Optimization, while achieving comparable performance.**

I. Introduction

For many robotics tasks such as manipulation and locomotion, robots frequently need to make and break contact with the environment. Yet, it is challenging to find explicit policies and/or trajectories that can exploit the interaction of the robot with its environment in order to enable robust, stable motion. Thus, with the great expressiveness of deep neural policies, modern model-free reinforcement learning algorithms, particularly Proximal Policy Optimization (PPO) [1], have grown in popularity in recent years due to the capability to generate interesting and efficient control policies to accomplish desired tasks. However, because of their data inefficiency, carrying out experiments on real robotic systems is always resource-intensive and time-consuming.

In the past few years, numerous research works have tried to improve the data efficiency of those model-free algorithms. An unifying theme across much of those prior works is utilizing interaction data to learn the underlying dynamics models, which can then be used for planning or generating more artificial data. That paradigm is commonly referred to as model-based reinforcement learning.

Some prominent works learn global dynamics models with the use of deep neural networks, then perform online planning with those models by using cross-entropy method (CEM) [2], [3] or Model Predictive Path Integral (MPPI) [4], [5]. In contrast, [6], [7] combine iLQR and locally approximated linear dynamics models to guide the search of optimal policies, but this approach could get stuck in local minima fairly easily. Furthermore, few other works such as [8]–[13] use the learned dynamics models to produce imagined data for policy training, hence drastically reduce the number of real interactions with environments.

The traditional MPC is a classic online planning tool, which can achieve great performance with the long horizon and accurate environment models [14]. However, MPC also suffers from several disadvantages, including high reliability on the model acquisition, and high cost to plan over a long horizon. But it can be potentially more sample efficient than model-free methods through model learning. Inspired by recent works about differentiable MPC [15], [16], our method seek to directly parameterize the policy of any model-free algorithms, PPO for instance, as a MPC instead of a deep neural network. By so doing, the policy is highly structured and eventually consumes less interaction data. Moreover, preserving the stochastic nature of PPO policy also improves its performance and robustness.

The focus of this work is two-fold: the first objective is to accomplish the desired task with less sampling data required and the second objective is to learn a simplified dynamics model of the system. We also demonstrate the effectiveness of our method on an underactuated multi-contact system.

II. Background

A. PPO Framework

PPO is currently the state-of-the-art model-free reinforcement learning algorithms due to the advantages of quick convergence, fewer hyper-parameters to tune and the ability of running in parallel. Generally, PPO is a policy gradient algorithm that concerns: using current data, how can we take the biggest possible improvement step on a policy, without stepping so far that we accidentally cause performance collapse. PPO consists of two main components: the value function V_ϕ and policy π_θ, both are parameterized by deep neural networks. The value function aids the policy optimization by estimating total rewards that could potentially be yielded by the current policy starting at a given state. The overall architecture of PPO framework is shown in Figure 1.

First, we need to understand how PPO defines its control policy. Similar to most model-free RL algorithms, PPO models the policy as a stochastic policy, with the Gaussian distribution being the most common form. Given any state s_t, the action

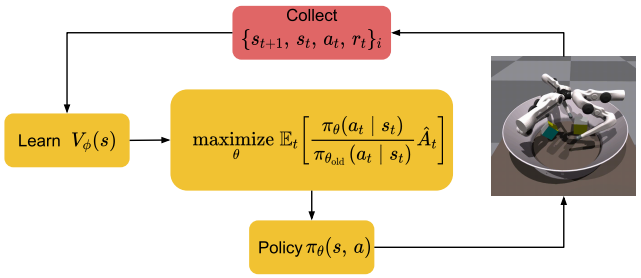

Fig. 1. The architecture of PPO framework.

a_t is obtained by simply sampling from the distribution $\mathcal{N}(\mu_\theta(s_t), \xi I_{n\times n})$ where n is the dimension of action.

Actually, $\pi_\theta(a_t|s_t)$ indicates the probability density of a distribution with a mean action value $\mu_\Theta(s_t)$, which is predicted by a neural network, and a learnable variance ξ.

$$\begin{aligned}\pi_\theta(a_t|s_t) &= \text{PDF}\ \left(\mathcal{N}(\mu_\Theta(s_t), \xi I_{n\times n})\right) \\ &= \frac{1}{(2\pi\xi)^{n/2}} \exp\left(-\frac{1}{2\xi}||a_t - \mu_\Theta(s_t)||^2\right)\end{aligned} \tag{1}$$

Next, the PPO loss function is defined as follows:

$$\begin{aligned}L_\theta^{CLIP} &= -\mathbb{E}_t\left[\min\left(h_t^\theta \hat{A}_t, \text{clip}\left(h_t^\theta, 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)\right] \\ &= -\frac{1}{|D|T}\sum_{\tau\in D}\sum_{t=0}^{T}\begin{cases}\max\left(h_t^\theta, 1-\epsilon\right)\hat{A}_t & \text{if } \hat{A}_t < 0 \\ \min\left(h_t^\theta, 1+\epsilon\right)\hat{A}_t & \text{if } \hat{A}_t \geq 0\end{cases}\end{aligned} \tag{2}$$

where D and $|D|$ are data buffer and its size, that buffer consists of rollout trajectories produced by the current policy; h_t^θ is the ratio of new policy over old policy; $\hat{A}_t$ is called the advantage function which measures how much is a certain action a good or bad decision given a certain state; and the scalar ϵ define the trust region of policy improvement.

The detailed expressions of h_t^θ and $\hat{A}_t$ are given below

$$h_t^\theta = \frac{\pi_\theta\left(a_t \mid s_t\right)}{\pi_{\theta_{\text{old}}}\left(a_t \mid s_t\right)} \tag{3}$$

$$\hat{A}_t = r_t + \gamma V_\phi\left(s_{t+1}\right) - V_\phi(s_t) \tag{4}$$

B. Discrete-time Linear Complementarity Systems

A discrete-time linear complementarity system (LCS) is a piecewise-linear system, where the state evolution is governed by a linear dynamics in (5a) and a linear complementarity problem (LCP) in (5b).

$$s_{t+1} = As_t + Ba_t + C\lambda_t + d \tag{5a}$$

$$0 \leq \lambda_t \perp Ds_t + Ea_t + F\lambda_t + c \geq 0 \tag{5b}$$

Here, $s_t \in \mathbb{R}^{n_x}$ and $a_t \in \mathbb{R}^{n_a}$ are the system state and action at time step t. And, $\lambda_t \in \mathbb{R}^{n_\lambda}$ is the complementarity variable at time step t.

The matrix $A \in \mathbb{R}^{n_x\times n_x}$ defines the autonomous dynamics and matrix $B \in \mathbb{R}^{n_x\times n_a}$ captures the effect of the action on the state. In addition, the matrix $C \in \mathbb{R}^{n_x\times n_\lambda}$ and $d \in \mathbb{R}^{n_x}$ describes the effect of the contact forces and the constant forces acting on the state respectively. Other matrices $D \in \mathbb{R}^{n_\lambda\times n_x}, E \in \mathbb{R}^{n_\lambda\times n_a}, F \in \mathbb{R}^{n_\lambda\times n_\lambda}$ and $c \in \mathbb{R}^{n_\lambda}$ altogether capture the relationship between states, actions, and contact forces.

III. Methodology

A. MPC-based PPO Policy

While keeping the PPO policy as a stochastic policy (1), we impose a structure on its mean value by parameterizing it as a MPC problem instead of a neural network. In addition, we choose LCS model as in (5) to represent the simplified dynamics model. LCS is known to sufficiently capture the contact dynamics that frequently arise in manipulation tasks.

$$\begin{aligned}\mu_\Theta(s) = \underset{a}{\text{argmax}} &\sum_{k=0}^{H-1} r\left(s_k, a_k\right) + V\left(s_H\right) \\ \text{s.t. } & s_{k+1} = As_k + Ba_k + C\lambda_k + d \\ & Ds_k + Ea_k + F\lambda_k + c \geq 0 \\ & \lambda_k \geq 0 \\ & \lambda_k^T\left(Ds_k + Ea_k + F\lambda_k + c\right) = 0 \\ & \text{for } k = 0, \ldots, H-1, \text{ given } s_0,\end{aligned} \tag{6}$$

where H is the planning horizon, $\Theta = (A, B, C, d, D, E, F, c)$ are parameters of the MPC-based policy.

Given any initial state s, we first use similar method in [17] to solve the optimization problem (6) to find a sequence of optimal actions $\left[a_0^*, a_1^*, \ldots, a_{H-1}^*\right]$, then only pick the first optimal action as the mean action, i.e. $\mu_\Theta(s) = a_0^*$. Subsequently, we sample the true action, the one we will apply on robots, from the Gaussian distribution $a \sim \mathcal{N}(\mu_\Theta(s), \xi I)$. Here, the set of parameters extends to $\theta = [\Theta, \xi]$.

B. Gradient of PPO Loss over MPC Parameters

In order to optimize the parameters of MPC-based policy, we have to compute the gradient of the PPO loss function with respect to those parameters. Given the explicit form of PPO loss in (2), using chain rule, the gradient with respect to θ can be computed as follows:

$$\begin{aligned}\frac{dL_\theta^{CLIP}}{d\theta} &= \frac{1}{|D|T}\times \\ &\sum_{\tau\in D}\sum_{t=0}^{T}\begin{cases}\frac{dh_t^\theta}{d\pi_\theta}\frac{d\pi_\theta}{d\theta}\hat{A}_t & \text{if } h_t^\theta \geq 1-\epsilon \text{ and } \hat{A}_t < 0 \\ 0 & \text{if } h_t^\theta < 1-\epsilon \text{ and } \hat{A}_t < 0 \\ \frac{dh_t^\theta}{d\pi_\theta}\frac{d\pi_\theta}{d\theta}\hat{A}_t & \text{if } h_t^\theta \leq 1+\epsilon \text{ and } \hat{A}_t \geq 0 \\ 0 & \text{if } h_t^\theta > 1+\epsilon \text{ and } \hat{A}_t \geq 0\end{cases}\end{aligned} \tag{7}$$

Due to clipping effect of PPO loss, the gradients are zero when the improvement steps of the PPO policy h_t^θ are outside of trust region $[1-\epsilon, 1+\epsilon]$. Hence, we are left to compute

the gradients if h_t^θ stays within the trust region. Specifically, to compute (7), ones requires to compute $\frac{dh_t^\theta}{d\pi_\theta}$ and $\frac{d\pi_\theta}{d\theta}$.

First, it is straightforward to evaluate $\frac{dh_t^\theta}{d\pi_\theta}$ in (7) as follows:

$$\frac{dh_t^\theta}{d\pi_\theta} = \frac{d}{d\pi_\theta}\left(\frac{\pi_\theta\left(a_t \mid s_t\right)}{\pi_{\theta_{\text{old}}}\left(a_t \mid s_t\right)}\right) = \frac{1}{\pi_{\theta_{\text{old}}}\left(a_t \mid s_t\right)} \tag{8}$$

Next, since θ is the joint vector of the MPC parameters Θ and the variance of the stochastic policy ξ, it is easier to evaluate $\frac{d\pi_\theta}{d\Theta}$ and $\frac{d\pi_\theta}{d\xi}$ separately and then assemble them back to obtain $\frac{d\pi_\theta}{d\theta}$.

$$\frac{d\pi_\theta}{d\theta} = \frac{d\pi_\theta}{d([\Theta,\xi])} = \left[\frac{d\pi_\theta}{d\Theta}, \frac{d\pi_\theta}{d\xi}\right] \tag{9}$$

Now, taking gradient of the explicit form of π_θ as in (1) with respect to Θ and ξ, we obtain

$$\begin{aligned}\frac{d\pi_\theta}{d\Theta} &= \frac{d}{d\Theta}\left(\frac{1}{(2\pi\xi)^{n/2}}\exp\left(-\frac{1}{2\xi}||a_t-\mu_\Theta(s_t)||^2\right)\right)\\ &= \frac{1}{(2\pi)^{n/2}\ \xi^{n/2+1}}\exp\left(-\frac{1}{2\xi}||a_t-\mu_\Theta(s_t)||^2\right)\times\\ &\quad (a_t-\mu_\theta(s_t))^T\frac{d\mu_\Theta(s_t)}{d\Theta}\end{aligned} \tag{10}$$

$$\begin{aligned}\frac{d\pi_\theta}{d\xi} &= \frac{d}{d\xi}\left(\frac{1}{(2\pi\xi)^{n/2}}\exp\left(-\frac{1}{2\xi}||a_t-\mu_\Theta(s_t)||^2\right)\right)\\ &= \frac{1}{2(\pi)^{n/2}\ \xi^{n/2+2}}\exp\left(-\frac{1}{2\xi}||a_t-\mu_\Theta(s_t)||^2\right)\times\\ &\quad \frac{1}{2}\left(||a_t-\mu_\theta(s_t))||^2-n\xi\right)\end{aligned} \tag{11}$$

Here, we follow [16] to compute the gradient of the MPC's otpimal action with respect to MPC parameters $\frac{d\mu_\Theta(s_t)}{d\Theta}$ in (10).

C. Prediction Loss for LCS Model

Because PPO loss only aims to improve the policy such that it could yield higher rewards as possible, there is no mechanism to ensure the correspondence between the learned dynamics model, here is LCS model, with the true dynamics model. In other words, given a current state, the dynamics model may predict the future state incorrectly, hence influencing the MPC's optimal action.

In this work, we use a simple prediction loss

$$L_\theta^{PRED} = \frac{1}{|D|T}\sum_{\tau\in D}\sum_{t=0}^{T}||s_t'-f_\Theta(s_t,a_t)||_2^2 \tag{12}$$

where $f_\Theta(s_t,a_t)$ is the LCS model (5)

In addition, the gradient of this prediction loss with respect to $\theta=[\Theta,\xi]$ can be expressed as below

$$\begin{aligned}\frac{dL_\theta^{PRED}}{d\theta} &= \left[\frac{dL_\theta^{PRED}}{d\Theta}\ \frac{dL_\theta^{PRED}}{d\xi}\right]\\ &= \left[\frac{dL_\theta^{PRED}}{d\Theta}\quad 0\right]\end{aligned} \tag{13}$$

where

$$\frac{dL_\theta^{PRED}}{d\Theta} = \frac{1}{|D|T}\sum_{\tau\in D}\sum_{t=0}^{T}(s_t'-f_\Theta(s_t,a_t))^T\frac{df_\Theta(s_t,a_t)}{d\Theta} \tag{14}$$

Combining the PPO loss and the prediction loss, we would obtain new loss and new gradient

$$L_\theta = L_\theta^{CLIP} + \alpha L_\theta^{PRED} \tag{15}$$

$$\frac{dL_\theta}{d\theta} = \frac{dL_\theta^{CLIP}}{d\theta} + \alpha\frac{dL_\theta^{PRED}}{d\theta} \tag{16}$$

where the hyper-parameter α indicates how significant the contribution of the prediction loss is.

D. Overall Training Algorithm

Based on the above development and the original PPO algorithm, we provide the entire algorithm for PPO with MPC-based policy in the following algorithm (1).

Algorithm 1: PPO with MPC-based Policy

Parametrization: MPC-based policy π_θ in (6) and value function V_ϕ

Hyper parameters: Total number of iterations K and number of policy improvement steps N; Learning rate η for the policy optimization; Weight α for prediction loss in (15, 16); and also the discount factor γ for computing the advantages and bootstrapping.

Initialization: θ_0 and ϕ_0

for $k=0,1,...,K$ **do**

Collect $D_k=\{\tau_i\}$ trajectories by running MPC-based policy π_{θ_k}

For each trajectory τ_i, compute the advantage function $\hat{A}_t$ as in (4) for $t=0,1,\ldots,T$

For each trajectory τ_i, compute the bootstrapped value of total reward $\hat{R}_t = r(s_t,a_t)+\gamma V_{\phi_k}(s_{t+1})$ for $t=0,1,\ldots,T$

`/* Improve policy using gradient descent */`

for $j\leftarrow 0$ **to** N **do**

Compute the gradient $\frac{dL_{\theta_j}}{d\theta_j}$ as in (16)

$\theta_{j+1}\leftarrow\theta_j-\eta\frac{dL_{\theta_j}}{d\theta_j}$

end

Fit value function V_ϕ by performing regression with mean-square error

$$\phi_{k+1}=\arg\min_\phi\frac{1}{|\mathcal{D}_k|\,T}\sum_{\tau\in\mathcal{D}_k}\sum_{t=0}^{T}\left(V_\phi\left(s_t\right)-\hat{R}_t\right)^2$$

end

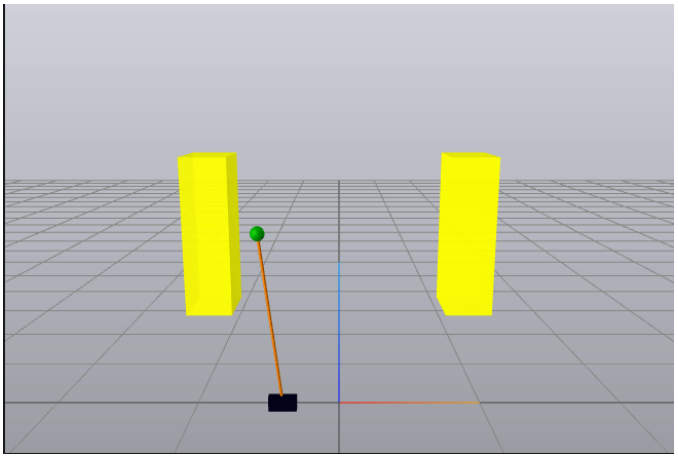

Fig. 2. Cartpole with soft walls

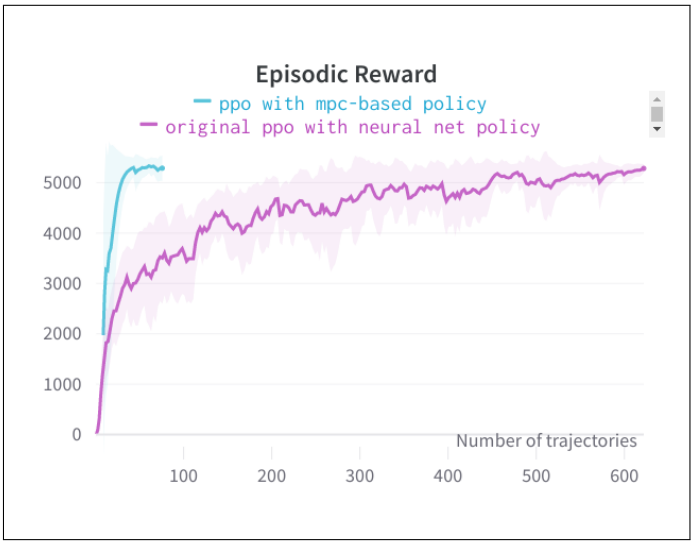

Fig. 3. The comparison of training performance of cartpole with soft walls task in two approaches: the original PPO with neural net policy and our PPO with MPC-based policy. The cyan and purple color lines are for the training with the MPC-based policy and the neural-net policy respectively.

IV. Experiments and Results

A. Experiment Settings

1) Cartpole with soft walls: We validate our proposed framework on a task that is stabilizing a cartpole system with two soft walls on both sides (Figure 2). This cartpole system was previously used in [18] and [19], and we re-implement it in Drake simulation [20].

The state s of this system is a 4-dimensional tuple $(x, \dot{x}, \theta, \dot{\theta})$, where x, $\dot{x}$, θ, and $\dot{\theta}$ are the linear position, velocity of the cart, the angular position and velocity of the pole respectively. The action a is the force applied on the cart, which has dimension of one, and $a \in [-100, 100]$ N. Two soft walls in yellow color are placed at a distance of $d = 0.35m$ from the origin and have stiffness of $700N/m$.

We design the reward function and termination conditions as follows:

- **Reward function**

$$r(s_t, a_t) = 15 - s_t^T \text{diag}(20, 20, 0.1, 0.1) s_t - 10^{-3} a_t^2$$

- **Termination condition**

$\theta \notin [\theta_{\min}, \theta_{\max}]$	The pole falls out of the safe range
$x \notin [x_{\min}, x_{\max}]$	The cart moves out of the safe range
$T > 400$	The episode length is longer than 400 steps, equivalent to 4 seconds

2) Training process: We run 5 training experiments with 5 different random seeds for both our proposed framework and the original PPO framework. We record the total rewards obtained by the learned policy with respect to the number of rollout trajectories.

B. Results

As seen in Figure (3), we observe that both the original PPO framework and our framework can learn good policies which have the same level of performance since the cartpole problem is fairly simple and low-dimensional. However, our proposed framework only requires approximately 40 rollout trajectories to learn the optimal policy while the original PPO framework needs around 600 trajectories, which is 15 times less data efficient than ours.

We create the video to compare the rollout trajectories generated by the original PPO, our approach, and also MPC with true dynamics model. The video is available at the following link https://youtu.be/mGEWMRvKkvo.

V. Conclusion and Future Work

In this paper, we present a novel reinforcement learning approach that combines the PPO framework with MPC-based policy. We validate the efficacy of our proposed approach using cartpole with soft walls experiments. The results are promising because data efficiency is higher than in the original PPO framework, while policy performance is comparable, at least in a simple and low-dimensional underactuated system.

We are planning to test this framework with more difficult and complex manipulation tasks such as object manipulation with a trifinger robot. We were unable to perform ablation studies or compare our framework to other model-based RL algorithms due to time constraints; these will be future work. In addition, it would be interesting to look into how to reuse the learned dynamics models for a family of similar tasks.

Acknowledgment

We would like to thank Professor Michael Posa and all the MEAM 5170 course staff for their constant support, Wanxin Jin and Yu-ming Chen for their helpful discussions related to solving and differentiating through MPC problems.

References

[1] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," Aug. 2017.

[2] Z. I. Botev, D. P. Kroese, R. Y. Rubinstein, and P. L'Ecuyer, "Chapter 3 - the cross-entropy method for optimization," in *Handbook of Statistics* (C. Rao and V. Govindaraju, eds.), vol. 31 of *Handbook of Statistics*, pp. 35–59, Elsevier, 2013.

[3] A. Nagabandi, K. Konoglie, S. Levine, and V. Kumar, "Deep Dynamics Models for Learning Dexterous Manipulation," Sept. 2019.

[4] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models," *arXiv:1805.12114 [cs, stat]*, Nov. 2018.

[5] N. Hansen, X. Wang, and H. Su, "Temporal Difference Learning for Model Predictive Control," July 2022.

[6] M. Zhang, S. Vikram, L. Smith, P. Abbeel, M. J. Johnson, and S. Levine, "SOLAR: Deep Structured Representations for Model-Based Reinforcement Learning," June 2019.

[7] W. H. Montgomery and S. Levine, "Guided Policy Search via Approximate Mirror Descent," in *Advances in Neural Information Processing Systems*, vol. 29, Curran Associates, Inc., 2016.

[8] R. S. Sutton, "Dyna, an integrated architecture for learning, planning, and reacting," *ACM SIGART Bulletin*, vol. 2, pp. 160–163, July 1991.

[9] R. S. Sutton, "Planning by Incremental Dynamic Programming," in *Machine Learning Proceedings 1991* (L. A. Birnbaum and G. C. Collins, eds.), pp. 353–357, San Francisco (CA): Morgan Kaufmann, Jan. 1991.

[10] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning," *arXiv:1708.02596 [cs]*, Dec. 2017.

[11] T. Kurutach, I. Clavera, Y. Duan, A. Tamar, and P. Abbeel, "Model-Ensemble Trust-Region Policy Optimization," Oct. 2018.

[12] I. Clavera, J. Rothfuss, J. Schulman, Y. Fujita, T. Asfour, and P. Abbeel, "Model-Based Reinforcement Learning via Meta-Policy Optimization," *ArXiv*, Sept. 2018.

[13] Y. Luo, H. Xu, Y. Li, Y. Tian, T. Darrell, and T. Ma, "Algorithmic Framework for Model-based Deep Reinforcement Learning with Theoretical Guarantees," Feb. 2021.

[14] C. E. García, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice—A survey," *Automatica*, vol. 25, pp. 335–348, May 1989.

[15] B. Amos, I. Jimenez, J. Sacks, B. Boots, and J. Z. Kolter, "Differentiable MPC for end-to-end planning and control," in *Advances in Neural Information Processing Systems* (S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), vol. 31, Curran Associates, Inc., 2018.

[16] W. Jin, S. Mou, and G. J. Pappas, "Safe Pontryagin Differentiable Programming," Oct. 2021.

[17] W. Jin and M. Posa, "Task-Driven Hybrid Model Reduction for Dexterous Manipulation," Nov. 2022.

[18] A. Aydinoglu, P. Sieg, V. M. Preciado, and M. Posa, "Stabilization of Complementarity Systems via Contact-Aware Controllers," *IEEE Transactions on Robotics*, vol. 38, pp. 1735–1754, June 2022.

[19] A. Aydinoglu and M. Posa, "Real-Time Multi-Contact Model Predictive Control via ADMM," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 3414–3421, May 2022.

[20] R. Tedrake and the Drake Development Team, "Drake: Model-based design and verification for robotics," 2019.