Model Predictive Control via On-Policy Imitation Learning

Kwangjun Ahn KJAHN@MIT.EDU
Zakaria Mhammedi MHAMMEDI@MIT.EDU
Horia Mania HMANIA@MIT.EDU
Zhang-Wei Hong ZWHONG@MIT.EDU
Ali Jadbabaie JADBABAI@MIT.EDU
Massachusetts Institute of Technology

Editors: N. Matni, M. Morari, G. J. Pappas

Abstract

In this paper, we leverage the rapid advances in imitation learning, a topic of intense recent focus in the Reinforcement Learning (RL) literature, to develop new sample complexity results and performance guarantees for data-driven Model Predictive Control (MPC) for constrained linear systems. In its simplest form, imitation learning is an approach that tries to learn an expert policy by querying samples from an expert. Recent approaches to data-driven MPC have used the simplest form of imitation learning known as behavior cloning to learn controllers that mimic the performance of MPC by online sampling of the trajectories of the closed-loop MPC system. Behavior cloning, however, is a method that is known to be data inefficient and suffer from distribution shifts. As an alternative, we develop a variant of the forward training algorithm which is an on-policy imitation learning method proposed by Ross and Bagnell (2010). Our algorithm uses the structure of constrained linear MPC, and our analysis uses the properties of the explicit MPC solution to theoretically bound the number of online MPC trajectories needed to achieve optimal performance. We validate our results through simulations and show that the forward training algorithm is indeed superior to behavior cloning when applied to MPC. The full version of this paper can be found at https://arxiv.org/abs/2210.09206.

Keywords: model predictive control, data-driven control, imitation learning, machine learning

1. Introduction

Optimization-based control methods such as model predictive control (MPC) have been among the most versatile techniques in feedback control design for more than 40 years. Such techniques have been successfully applied to control of dynamic systems in a variety of domains such as autonomous vehicles (Murray et al., 2003; Jadbabaie and Hauser, 2002; Falcone et al., 2007; Rosolia and Borrelli, 2017), chemical plants (Qin and Badgwell, 2003), humanoid robots (Kuindersma et al., 2016), and many others. Nonetheless, MPC's versatility comes at a cost. Having to solve optimization problems online makes it difficult to deploy MPC on high-dimensional systems that have strict latency requirements and limited computational or energy resources. To mitigate this issue, considerable effort went into developing faster, tailored optimization methods for MPC (Boyd et al., 2011; Giselsson et al., 2013; Jerez et al., 2014; Kögel and Findeisen, 2011; Lucia et al., 2016, 2018; Mattingley and Boyd, 2012; Richter et al., 2012; Zometa et al., 2013).

Instead of following these approaches, we pursue a data-driven methodology. We propose and study a scheme to collect data interactively from a dynamical system in feedback with an MPC

controller and in order to learn an explicit controller that maps states to inputs. Such approaches are known in the reinforcement learning literature as *imitation learning* (Pomerleau, 1988; Schaal, 1999) and they are well suited for MPC because one can query MPC for the next input at any desired state; all that is needed is to solve the corresponding optimization problem. Nonetheless, in order to learn controllers that are guaranteed to stabilize dynamical systems, to satisfy state and action constraints, and to obtain low cost, we would need to exploit several properties of MPC. Our goal of obtaining an explicit map from states to inputs that encapsulates an MPC controller falls under the purview of *explicit MPC* (Bemporad et al., 2002), which aims to pre-compute and store the solutions of the optimization problems that might be encountered at runtime (Alessio and Bemporad, 2009).

In general, explicit MPC aims to pre-compute an exact representation of the MPC controller while we aim to learn a controller that performs as well as MPC with high probability. In the same vain, Hertneck et al. (2018) and Karg and Lucia (2020) suggest learning a controller from data. However, their approaches collect all the trajectory data using MPC before any learning occurs and do not interact with the dynamics further. The lack of interaction in imitation learning is known to lead to sub-optimal performance because of the issue called *distribution shift* (see Section 2 for details), leading to error compounding. Our proposed approach completely avoids this issue. To this end, our contributions in this paper can be summarized as follows:

- We start by analyzing the imitation learning method known as the forward training algorithm (FORWARD) in the setting of control affine systems (Ross and Bagnell, 2010).
- We modify FORWARD to make it suitable for MPC applications with constraints. Firstly, FORWARD learns a different controller for each distinct time step and hence it cannot be applied straightforwardly to problems with long or infinite horizons. Fortunately, after sufficiently many times steps, the MPC controller applied to time invariant linear systems becomes equivalent to the classical linear quadratic regulator (LQR) (Sznaier and Damborg, 1987). We exploit this property; we modify FORWARD to switch to LQR after a number of time steps estimated from data. We refer to our modified method as FORWARD-SWITCH.
- We theoretically guarantee that a controller learned with FORWARD-SWITCH stabilizes linear systems and satisfies their constraints as long as certain amount of data is available. Moreover, we bound the cost suboptimality of the learned controller, showing that it approaches optimal performance as more data becomes available. None of the previous works on imitating MPC included such guarantees. We also provide theoretical sample complexity bounds using state of the art tools of high dimensional statistics and statistical learning theory.
- We validate the efficacy of the modified forward training algorithm on simulated MPC problems, showing that it surpasses non-interactive approaches for unstable systems.

2. The Forward Training Algorithm for Control

In this section, we present the imitation learning method FORWARD (Ross and Bagnell, 2010) and provide its performance guarantees when the dynamics are control-affine. More specifically, we consider control-affine dynamical systems with constraints:

$$x_{t+1} = f(x_t) + g(x_t)u_t\,, \quad x_t \in \mathcal{X} \text{ and } u_t \in \mathcal{U}\,, \tag{2.1}$$

where $\mathcal{X} \subset \mathbb{R}^{d_x}$ is the state space and $\mathcal{U} \subset \mathbb{R}^{d_u}$ is the input space. We also find it useful to denote $\varphi_t(x_0, \{u_t\}_{t\geqslant 0})$ the state x_t that evolves according to $x_{t+1} = f(x_t) + g(x_t)u_t$ and starts at x_0.

When the dynamics evolve according to a time-varying feedback controller $\pi = \pi_{0:t-1}$ (i.e. π_0 is used at time 0, π_1 at time 1, etc.) we denote the state at time t by $\varphi_t(x_0; \pi_{0:t-1})$. If the controller π is time-invariant, we simply write $\varphi_t(x_0; \pi)$.

Imitation Learning: Imitation learning aims to learn from demonstrations a controller $\hat{\pi}$ that imitates the behavior of a target controller $\pi^\star$, called *expert policy* or simply *expert* in the reinforcement learning literature. Imitation learning is valuable when $\pi^\star$ lacks a closed-form expression or is expensive to query in general. For instance, $\pi^\star$ could be a human performing a task or a MPC controller. More formally, in imitation learning it is assumed that for a state x we can access the input $\pi^\star(x)$. Then, the aim is to use data $\{x_i, \pi^\star(x_i)\}$ to learn a controller $\hat{\pi}$ such that $\hat{\pi}(x) \approx \pi^\star(x)$.

Behavior cloning (BC) is the simplest imitation learning method. It consists of collecting m independent trajectories $\varphi_t(x_0^{(i)}; \pi^\star)$ with initial states $x_0^{(1)}, x_0^{(2)}, \ldots, x_0^{(n)}$ sampled randomly from an initial distribution $\mathcal{D}$. Then, BC produces a controller $\hat{\pi}_{\mathsf{BC}}$ through Empirical Risk Minimization:

$$\hat{\pi}_{\mathsf{BC}} \in \underset{\pi \in \Pi}{\text{minimize}} \sum_{i=1}^{n} \sum_{t=0}^{T-1} \left\| \pi(\varphi_t(x_0^{(i)}; \pi^\star)) - \pi^\star(\varphi_t(x_0^{(i)}; \pi^\star)) \right\| ,$$

where Π is a class of models that map the state space to the input space and $\|\cdot\|$ is any norm (although it could be replaced by a more general loss function). All our results assume that $\pi^\star \in \Pi$.

Distribution Shift: The states collected using the expert $\pi^\star$ have a particular distribution $\mathcal{D}^\star$. BC produces a controller $\hat{\pi}_{\mathsf{BC}}$ that, when evaluated on samples from $\mathcal{D}^\star$, behaves similarly to the expert $\pi^\star$. However, $\hat{\pi}_{\mathsf{BC}}$ is not a perfect copy of the expert and hence the states encountered during its deployment have a different distribution than $\mathcal{D}^\star$. This discrepancy is well known and leads to errors compounding in practice (Pomerleau, 1988; Ross and Bagnell, 2010; Ross et al., 2011). More explicitly, consider an initial state x_0 sampled from $\mathcal{D}$. Then, at the first time step $\hat{\pi}_{\mathsf{BC}}$ and $\pi^\star$ perform similarly since $\hat{\pi}_{\mathsf{BC}}$ was trained using data sampled from $\mathcal{D}$. However, at the second time step the distributions over states produced by $\hat{\pi}_{\mathsf{BC}}$ and $\pi^\star$ are different, which means that at the second time step $\hat{\pi}_{\mathsf{BC}}$ would be evaluated on a distribution different than the one on which it was trained. Hence, with each time step, $\hat{\pi}_{\mathsf{BC}}$ can take the dynamical system to parts of the state space that are less and less covered by the training trajectories resulting in error compounding.

Since BC does not account for the intrinsic distribution shift in imitation learning, the number of training trajectories it requires to guarantee a good learned controller can be large (e.g. exponential in the number of time steps or dimension). The methods for learning a MPC controller due to Hertneck et al. (2018), and Karg and Lucia (2020) are variants of behavior cloning and hence also suffer from the presence of distribution shift. Instead, we use and theoretically analyze the forward training algorithm that was initially used by Ross and Bagnell (2010) for the tabular MDP setting.

Forward Training Algorithm: FORWARD learns a time-varying feedback controller $\hat{\pi}_{0:T-1}$ in an inductive fashion: during stage 0, it obtains $\hat{\pi}_0$ from the Empirical Risk Minimizer (ERM) (2.2). The controller $\hat{\pi}_0$ is used in the dynamical system just at the initial time step. Then, given already learned controllers $\hat{\pi}_0, \cdots, \hat{\pi}_{t-1}$, to learn the policy $\hat{\pi}_t$ for time step t, FORWARD samples states $\hat{x}_t^{(i)} = \varphi_t(x_0^{(i)}; \hat{\pi}_{0:t-1})$, where $x_0^{(1)}, x_0^{(2)}, \cdots$ are sampled i.i.d. from the initial state distribution $\mathcal{D}$.

The advantage of this method is that at time step t during deployment the controller $\hat{\pi}_t$ would be evaluated on the same distribution as that on which it was trained. A similar idea has been employed in other recent works (Mhammedi et al., 2020; Sun et al., 2019).

FORWARD TRAINING ALGORITHM *[Ross and Bagnell (2010)] Given n and T, a time-varying policy $\hat{\pi}_{0:T-1}$ is computed iteratively according to the following procedure:*

Stage $t = 0, \dots, T-1$**:** *Sample new initial states $x_0^{(1)}, \cdots, x_0^{(n_t)} \sim \mathcal{D}$, where $n_t := cnt\lceil \ln^2(t+1) \rceil + n$ and $c := \sum_{t=1}^{\infty} 1/(t \ln^2(t+1))$, then evaluate the states $\hat{x}_t^{(i)} := \varphi_t(x_0^{(i)}; \hat{\pi}_{0:t-1})$ using the controllers $\hat{\pi}_{0:t-1}$ learned in previous stages. Then, select $\hat{\pi}_t$ s.t.*

$$\hat{\pi}_t \in \operatorname*{argmin}_{\pi \in \Pi} \frac{1}{n_t} \sum_{i=1}^{n_t} \|\pi^\star(\hat{x}_t^{(i)}) - \pi(\hat{x}_t^{(i)})\|. \tag{2.2}$$

Since $\pi^\star$ is only defined on $\mathcal{X}$ and since $\hat{x}_t^{(i)}$ could lie outside $\mathcal{X}$, we define $\pi^\star(x) = \pi^\star(\operatorname{proj}_{\mathcal{X}} x)$.
Output: *The time-varying controller $\hat{\pi} = \hat{\pi}_{0:T-1}$.*

2.1. The Sample Complexity of Learning a Controller with FORWARD

In this section, we discuss our statistical guarantees of the controllers produced by FORWARD. In this section we consider the setting without state constraints, i.e., $\mathcal{X} = \mathbb{R}^{d_x}$; the setting with state constraints will be considered in Section 4. Before we can state the main results of this section, we need to make an assumption on the class of controllers Π used by FORWARD.

Assumption 1 *The model class Π is a finite and contains $\pi^\star$. Moreover, for any $\pi \in \Pi$ and any $x \in \mathcal{X}$ we have $\pi(x) \in \mathcal{U}$.*

The second part of the assumption just guarantees that Π enforces the input constraints. Any controller class can be modified to satisfy this property by projecting the outputs of the controllers onto $\mathcal{U}$. We assume that the controller class Π is finite for simplicity. In this case, our sample complexity guarantees scale with $\ln|\Pi|$—a quantity that arises through a standard generalization bound. When Π is not finite, one can replace $\ln|\Pi|$ by learning-theoretic complexity measures such as the Rademacher complexity. Finally, in the MPC application we care about, the assumption $\pi^\star \in \Pi$ is easily satisfied. In the case of constrained linear dynamics with quadratic costs the optimal MPC controller is piecewise affine and it can be expressed as a neural network with ReLU activations as extensively discussed by Karg and Lucia (2020, Section I-D).

Now we are ready to state the main result of this section. Its proof relies on the empirical Bernstein inequality (Maurer and Pontil, 2009) and is deferred to (Ahn et al., 2022).

Theorem 1 *Let $T \geqslant 1$ be the target time step, $\delta \in (0,1)$, $n \geqslant 2$, and $\mathfrak{B}_u := \sup_{u \in \mathcal{U}} \|u\|$. Let $\hat{x}_t = \varphi_t(x_0; \hat{\pi}_{0:t-1})$. When Assumption 1 holds, then under an event $\mathcal{E}$ of probability $\geqslant 1-\delta$ (over the randomness in the training), FORWARD produces $\hat{\pi}_{0:T-1}$ that satisfies*

$$\mathbb{E}\left[\|\pi^\star(\hat{x}_t) - \hat{\pi}_t(\hat{x}_t)\| \,\middle|\, \hat{\pi}_{0:T-1}\right] \leqslant \frac{7\mathfrak{B}_u \ln(2T|\Pi|/\delta)}{n_t}, \quad \forall t \geqslant 0, \tag{2.3}$$

where $n_t := cnt\lceil \ln^2(t+1) \rceil + n$, $c := \sum_{t=1}^{\infty} 1/(t\ln^2(t+1))$, and the expectation in (2.3) is with respect to the randomness in the initial state. Furthermore, under the same event,

$$\mathbb{P}\left[\forall t \geqslant 0, \;\; \|\pi^\star(\hat{x}_t) - \hat{\pi}_t(\hat{x}_t)\| \leqslant \frac{14\mathfrak{B}_u \ln(2T|\Pi|/\delta)}{n\delta} \,\middle|\, \hat{\pi}_{0:T-1}\right] \geqslant 1-\delta.$$

This result guarantees that the time-varying controller learned by FORWARD is close in expectation or with high probability to the optimal controller.

Infinite Model Classes: The results presented in this section assume Π is finite for simplicity. This assumption can be easily relaxed. For example, to get an analogue of the result of Theorem 1 for a infinite class Π, one can use the empirical Bernstein inequality (Maurer and Pontil, 2009, Lemma 6), which replaces $\ln|\Pi|$ by the logarithm of a "growth function" for the class Π. In the case where Π is a class of ReLU Neural Networks, the latter quantity can be bounded by $\widetilde{O}(N_{\mathsf{params}})$, where N_{params} is the number of parameters of the Neural Networks in Π.

Trajectory Guarantees: Theorem 1 guarantees that FORWARD produces a controller $\hat{\pi}$ that generates inputs to the system that are close to those outputted by $\pi^\star$. However, this result does not immediately imply that $\hat{\pi}$ and $\pi^\star$ follow similar trajectories (errors could compound over time, causing $\hat{\pi}$'s trajectories to diverge from those of $\pi^\star$). Following the main ideas of Tu et al. (2022) and Pfrommer et al. (2022), one can in fact show guarantees in terms of trajectories when the closed-loop system under $\pi^\star$ is robust in an appropriate sense. See (Ahn et al., 2022) for the details.

In subsequent sections, we refine the results presented so far to the case of MPC.

3. Brief Background on Model Predictive Control

In this section, we provide a brief background on the control of linear systems, with a focus on MPC. We refer readers to textbooks (Morari and Lee, 1999; Rawlings et al., 2017; Borrelli et al., 2017) for extensive background. We consider the constrained linear dynamical system

$$x_{t+1} = Ax_t + Bu_t, \quad x_t \in \mathcal{X} \text{ and } u_t \in \mathcal{U}. \tag{3.1}$$

which is a special case of the control-affine setting (2.1) with $f(x_t) = Ax_t$ and $g(x_t) = B$.

Suppose we wish to design a controller that minimizes the quadratic cost $\sum_{t=0}^{\infty} x_t^\top Q x_t + u_t^\top R u_t$ (where $R \succ 0$ and $Q \succcurlyeq 0$) under the dynamics (3.1) such that $x_t \in \mathcal{X}$ and $u_t \in \mathcal{U}$ for all $t \geqslant 0$. When the problem is unconstrained, i.e., $\mathcal{X} = \mathbb{R}^{d_x}$ and $\mathcal{U} = \mathbb{R}^{d_u}$, it is well known that the optimal controller is a stationary policy given by the LQR controller defined as

$$\pi^{\mathsf{lqr}}(x) = K^{\mathsf{lqr}} x, \quad \text{where } K^{\mathsf{lqr}} = -(R + B^\top P^{\mathsf{lqr}} B)^{-1} B^\top P^{\mathsf{lqr}} A\,, \tag{3.2}$$

where P^{lqr} be the unique positive definite solution to the discrete algebraic Riccati equation $P = A^\top P A - A^\top P B(R + B^\top P B)^{-1} B^\top P A + Q$. One important property of π^{lqr} is that the closed-loop system induced by the controller is stable, i.e., $A_{K^{\mathsf{lqr}}} := A + BK^{\mathsf{lqr}}$ satisfies $\rho(A_{K^{\mathsf{lqr}}}) < 1$. However, solving an infinite horizon problem under the presence of constraints is computationally challenging. Moreover, it is not sufficient to find an optimal sequence of inputs $\{u_t\}$ because open-loop control is brittle in the presence of noise.

Model Predictive Control (MPC): MPC precisely resolves these issues by designing a **feedback controller** using the following **finite horizon** N-step optimal control problem. For any given initial state $x \in \mathcal{X}$ and a sequence of control inputs $\boldsymbol{u} = (u_0, u_1, \cdots, u_{N-1})$,

$$V_N(x, \boldsymbol{u}) := \sum_{k=0}^{N-1} x_k^\top Q x_k + u_k^\top R u_k + x_N^\top P_{\mathsf{f}} x_N\,, \quad \text{s.t. } x_{t+1} = Ax_t + Bu_t,\ x_0 = x,$$

where $P_{\mathsf{f}}, Q \succcurlyeq 0$ and $R \succ 0$. Then, the finite horizon problem is:

$$\underset{\boldsymbol{u}=(u_0,u_1,\cdots,u_{N-1})}{\text{minimize}} \{V_N(x,\boldsymbol{u}) \mid x_t \in \mathcal{X}, u_t \in \mathcal{U}\ \forall t, \text{ and } x_N \in \mathcal{X}_{\mathsf{f}}\}\ , \tag{MPC}$$

where $\mathcal{X}, \mathcal{U}, \mathcal{X}_{\mathsf{f}}$ are constraint sets that are closed and contain the origin. Given this, the MPC feedback controller is defined as

$$\pi^{\mathsf{mpc}}(x_0) \coloneqq u_0^\star(x_0), \quad \text{where } u_0^\star(x), \ldots, u_{N-1}^\star(x) \text{ are the optimal solutions to (MPC).}$$

In other words, instead of deploying all open-loop solutions $u_0^\star(x), u_1^\star(x), \ldots, u_{N-1}^\star(x)$, MPC only deploys the first input $u_0^\star(x)$, observes the next state of the system, and solves another N-step problem starting at the new state. The **feasible domain** of the MPC controller π^{mpc} w.r.t. $(\mathcal{X}, \mathcal{U}, \mathcal{X}_{\mathsf{f}})$ is denoted by $\mathcal{X}_0$, and is recursively defined as $\mathcal{X}_N = \mathcal{X}_{\mathsf{f}}$, and $\mathcal{X}_i \coloneqq \{x \in \mathcal{X} : \exists u \in \mathcal{U} \text{ s.t. } Ax + Bu \in \mathcal{X}_{i+1}\}$, for $i = N-1, \ldots, 0$.

It turns out in order to have the MPC controller successfully stabilize the system to the origin, one needs to carefully choose the terminal cost p_{f} and the terminal set $\mathcal{X}_{\mathsf{f}}$. For concreteness, in the remaining of the paper, we focus on the standard choice

$$P_{\mathsf{f}} = P^{\mathsf{lqr}} \quad \text{and} \quad \mathcal{X}_{\mathsf{f}} = \mathcal{O}_\infty^{\mathsf{lqr}}(\mathcal{X}, \mathcal{U})\,, \tag{3.3}$$

where $\mathcal{O}_\infty^{\mathsf{lqr}}(\mathcal{X}, \mathcal{U})$ is the *maximal positively invariant* set w.r.t. the LQR controller. More formally, $\mathcal{O}_\infty^{\mathsf{lqr}}(\mathcal{X}, \mathcal{U})$ is a maximal subset such that if $\mathcal{O} \subseteq \mathcal{X}$ and whenever $x_0 \in \mathcal{O}$, then $x_t \in \mathcal{O}$ and $\pi^{\mathsf{lqr}}(x_t) = K^{\mathsf{lqr}} x_t \in \mathcal{U}$, for all $t \geqslant 0$, where $x_{t+1} = (A + BK^{\mathsf{lqr}})x_t$. As detailed in Borrelli et al. (2017, Sec. 10.2), the maximal positively invariant set $\mathcal{O}_\infty^{\mathsf{lqr}}(\mathcal{X}, \mathcal{U})$ (or its polytopic inner approximations) can be computed using polytopic computations. With the choice (3.3), we also have the following useful property that we use later:

$$\pi^{\mathsf{mpc}}(x) = \pi^{\mathsf{lqr}}(x) \quad \text{whenever } x \in \mathcal{O}_\infty^{\mathsf{lqr}}\,. \tag{3.4}$$

See, e.g., (Rawlings et al., 2017, Sec. 2.5.4) for details. At a high level, when $x \in \mathcal{O}_\infty^{\mathsf{lqr}}$, the inputs produced by the LQR controller π^{lqr} correspond to the optimal solution of (MPC) since they are the optimal solution to the unconstrained objective V_N.

Robust MPC: In our main result, we use robust MPC proposed by Mayne et al. (2005) as the expert policy instead of the 'vanilla' MPC. We summarize the properties of robust MPC that are necessary to present our main result, while deferring details to (Ahn et al., 2022). We begin with notations: for two arbitrary sets $\mathcal{X}$ and $\mathcal{Y}$, $\mathcal{X} \ominus \mathcal{Y}$ is defined as $\mathcal{X} \ominus \mathcal{Y} \coloneqq \{x \in \mathcal{X} \,|\, x + \mathcal{Y} \subseteq \mathcal{X}\}$, and $\mathsf{B}(r)$ denotes the Euclidean ball of radius $r > 0$.

Robust MPC stabilizes the constrained linear system with disturbances w_t s.t. $\|w_t\| \in \mathsf{B}(\varepsilon)$:

$$x_{t+1} \coloneqq Ax_t + Bu_t + w_t\,, \quad x_0 = x,\ x_t \in \mathcal{X},\ u_t \in \mathcal{U},\ w_t \in \mathsf{B}(\varepsilon)\,. \tag{3.5}$$

In order to attain this robustness property, robust MPC shrinks the constraint sets $\mathcal{X}$ and $\mathcal{U}$ using the *disturbance invariant set* (Kolmanovsky and Gilbert, 1998). Given a compact disturbance set $\mathcal{W}$, we say that a neighborhood around the origin $\Delta_{\mathcal{W}}$ is a disturbance invariant set if it satisfies $A_{K^{\mathsf{lqr}}} \Delta_{\mathcal{W}} + \mathcal{W} \subseteq \Delta_{\mathcal{W}}$, where $A_{K^{\mathsf{lqr}}} \coloneqq A + BK^{\mathsf{lqr}}$. In our case where $\mathcal{W} = \mathsf{B}(\varepsilon)$, one can show that $\Delta_{\mathcal{W}} \subseteq \mathsf{B}(\kappa \cdot \varepsilon)$, where κ is a constant depending on the system parameters. Then, robust MPC

uses the shrunk constraint sets $\bar{\mathcal{X}} \coloneqq \mathcal{X} \ominus \Delta_{\mathcal{W}}$ and $\bar{\mathcal{U}} \coloneqq \mathcal{U} \ominus K^{\mathsf{lqr}}\Delta_{\mathcal{W}}$. The terminal set is also shrunk accordingly $\bar{\mathcal{X}}_{\mathsf{f}} = \bar{\mathcal{O}}_{\infty}^{\mathsf{lqr}} \coloneqq \mathcal{O}_{\infty}^{\mathsf{lqr}}(\bar{\mathcal{X}}, \bar{\mathcal{U}})$, and let $\bar{\mathcal{X}}_0$ is the feasible domain w.r.t. $(\bar{\mathcal{X}}, \bar{\mathcal{U}}, \bar{\mathcal{X}}_{\mathsf{f}})$. Let $\bar{\pi}^{\mathsf{mpc}}$ be the MPC controller defined by these shrunk constraint sets and terminal set.

Now, the key idea of Mayne et al. (2005) is to include the initial point x_0 as a parameter of the optimization problem: given $x \in (\bar{\mathcal{X}}_0 \oplus \Delta_{\mathcal{W}}) \cap \mathcal{X}$,

$$\underset{x_0, \boldsymbol{u}}{\text{minimize}} \left\{ V_N(x_0, \boldsymbol{u}) \mid x_t \in \bar{\mathcal{X}},\ u_t \in \bar{\mathcal{U}}\ , \forall t,\ x_N \in \bar{\mathcal{X}}_{\mathsf{f}} \text{ and } x \in x_0 \oplus \Delta_{\mathcal{W}} \right\} . \tag{3.6}$$

Letting $\bar{x}_0(x), \bar{u}_0(x), \bar{u}_1(x), \ldots, \bar{u}_{N-1}(x)$ be the optimal solution to (3.6), the robust MPC controller is defined as $\boldsymbol{\pi}(x) \coloneqq \bar{u}_0(x) + K^{\mathsf{lqr}}(x - \bar{x}_0(x)) = \bar{\pi}^{\mathsf{mpc}}(\bar{x}_0(x)) + K^{\mathsf{lqr}}(x - \bar{x}_0(x))$. The following result establishes a key property of the robust MPC controller $\boldsymbol{\pi}$.

Proposition 2 (Mayne et al. (2005)) *For any $x \in (\bar{\mathcal{X}}_0 \oplus \Delta_{\mathcal{W}}) \cap \mathcal{X}$, the robust MPC controller $\boldsymbol{\pi}$ robustly stabilizes the system with disturbances (3.5) in the sense that there exists a constant $\zeta \in (0, 1)$ s.t. $\|\bar{x}_0(x_t)\| = O(\zeta^t \|\bar{x}_0(x)\|)$ for all $t \in \mathbb{N}$.*

Note that the conclusion $\|\bar{x}_0(x_t)\| = O(\zeta^t \|\bar{x}_0(x)\|)$ holds for any choices of disturbances $\{w_t\}$ as long as $w_t \in \mathcal{W}$, for all $t \in [T]$. Hence, for any initial state, there must exist a time step after which the state lies in $\bar{\mathcal{O}}_{\infty}^{\mathsf{lqr}}$.

Definition 3 (Time Step to Reach Positive Invariance) *Given a initial distribution $\mathcal{D}$ whose support is almost surely bounded, let $\tau_{\infty}^{\star}$ be the time step such that $\bar{x}_0(x_{\tau_{\infty}^{\star}}) \in \bar{\mathcal{O}}_{\infty}^{\mathsf{lqr}}$ almost surely. (Note that $\tau_{\infty}^{\star}$ depends solely on the system parameters and can be regarded as an absolute constant.)*

From (3.4), we know $\bar{\pi}^{\mathsf{mpc}}(x) = \pi^{\mathsf{lqr}}(x)$ for $x \in \bar{\mathcal{O}}_{\infty}^{\mathsf{lqr}}$, Hence, it holds that

$$\forall t \geqslant \tau_{\infty}^{\star},\ \ \boldsymbol{\pi}(x_t) = \bar{\pi}^{\mathsf{mpc}}(\bar{x}_0(x_t)) + K^{\mathsf{lqr}}(x_t - \bar{x}_0(x_t)) = \pi^{\mathsf{lqr}}(x_t)\,. \tag{3.7}$$

4. On-policy Imitation Learning for MPC

In this section, we discuss how to adapt FORWARD to imitate robust MPC and we offer refined guarantees on the performance of the modified FORWARD method. We consider the constrained linear dynamical system (3.1). Our result can handle the case with state constraints, i.e., $\mathcal{X} \subsetneq \mathbb{R}^{d_x}$.

Limitations of FORWARD: The algorithm FORWARD learns a time-varying controller, which allows it to elude the challenge of distribution shift. However, as explained by Ross and Bagnell (2010), learning a time-varying controller implies that the sample complexity grows linearly with the number of stages T. Therefore, when the target time step T is too large, a straightforward application of FORWARD would be computationally prohibited.

Our Approach: To address this, we use the property (3.7), namely, after using robust MPC for a certain a number of time steps the state of the dynamics reaches a region where robust MPC agrees with the infinite horizon LQR, at which point it is safe to switch to π^{lqr}. Our algorithm, FORWARD-SWITCH, estimates the number of steps the learned controller $\hat{\pi}$ requires to drive the state to $\bar{\mathcal{O}}_{\infty}^{\mathsf{lqr}}$. Given that FORWARD learns the controller incrementally for each time step, one can estimate the number of steps t by checking if all the states at stage t from the generated trajectories have reached $\bar{\mathcal{O}}_{\infty}^{\mathsf{lqr}}$. The next theorem justifies the step (4.1) of FORWARD-SWITCH that switches the learned controller $\hat{\pi}_t$ to π^{lqr} for $t \geqslant \hat{\tau}_{\infty}$: we show that with high probability, it holds that $\hat{\tau}_{\infty} \leqslant \tau_{\infty}^{\star}$ and $\hat{x}_{\hat{\tau}_{\infty}}$ lies in $\bar{\mathcal{O}}_{\infty}^{\mathsf{lqr}}$ (in which the expert policy is indeed π^{lqr}). The proof relies on Theorem 2 and Theorem 1; see Ahn et al. (2022) for full details.

FORWARD-SWITCH $T =$ *Imitation learning times steps.* $\hat{\tau}_\infty$ *is initialized as* $\hat{\tau}_\infty = T$. *We use* FORWARD *with* $\pi^\star$ *chosen as the robust MPC controller* $\boldsymbol{\pi}$ *to learn* $\tilde{\pi}_{0:T-1}$ *as follows:*

Forward training until positive invariance: *At the beginning of each stage* t *of* FORWARD*, we sample* ℓ *trajectories according to our learned controller* $\hat{\pi}_{0:t-1}$ *to generate* $\hat{x}_t^{(i)}$, $i \in [\ell]$.

- *If* $\hat{x}_t^{(i)} \in \bar{\mathcal{O}}_\infty^{\mathsf{lqr}}$ *for all* $i = 1, 2, \ldots, \ell$, *then we terminate* FORWARD *early and set* $\hat{\tau}_\infty = t$.
- *Otherwise, proceed to the next stage.*

Output policy: *With* π^{lqr} *as in (3.2) Output a time-varying policy* $\tilde{\pi}_{0:T-1}$ *defined as*

$$\tilde{\pi}_t \coloneqq \begin{cases} \hat{\pi}_t, & \text{if } t < \hat{\tau}_\infty, \\ \pi^{\mathsf{lqr}}, & \text{if } \hat{\tau}_\infty \leqslant t \leqslant T-1\,. \end{cases} \tag{4.1}$$

Theorem 4 *Let* $\delta, \varepsilon \in (0,1)$ *and* $\mathfrak{B}_u \coloneqq \sup_{u \in \mathcal{U}} \|u\|$. *Suppose Assumption 1 holds and that the support of* $\mathcal{D}$ *is almost surely bounded. Choose* $n \geqslant \frac{14\ell\|B\|\mathfrak{B}_u \ln(2T\ell|\Pi|/\delta)}{\varepsilon\delta}$ *and* ℓ *such that* $\ell \geqslant \frac{10\ln(T/\delta)}{\delta}$. *Then, under an event* $\bar{\mathcal{E}}$ *of probability at least* $1 - 3\delta$ *(over the randomness in the training process), the stopping time* $\hat{\tau}_\infty$ *in* FORWARD-SWITCH *satisfies* $\hat{\tau}_\infty \leqslant \tau_\infty^\star$ *and*

$$\begin{aligned} \mathbb{P}[\hat{x}_{\hat{\tau}_\infty} \in \bar{\mathcal{O}}_\infty^{\mathsf{lqr}} \mid \hat{\tau}_\infty, \hat{\pi}] &\geqslant 1 - \delta, \\ \mathbb{P}\left[\forall t = 0, \ldots, \hat{\tau}_\infty - 1,\ \|\boldsymbol{\pi}(\hat{x}_t) - \hat{\pi}_t(\hat{x}_t)\| \leqslant \varepsilon/\|B\| \mid \hat{\tau}_\infty, \hat{\pi}\right] &\geqslant 1 - \delta, \end{aligned} \tag{4.2}$$

Further, under $\bar{\mathcal{E}}$ *and the events in (4.2), the controller* $\tilde{\pi}_{0:T-1}$ *does not violate any constraints.*

Remark 5 (Sample Complexity of FORWARD-SWITCH**)** *Note that under the setting of Theorem 4, the total number of expert demonstrations required by* FORWARD-SWITCH *is upper bounded by* $\widetilde{O}(n\hat{\tau}_\infty) = \widetilde{O}(\frac{\ell\hat{\tau}_\infty}{\varepsilon\delta}) = \widetilde{O}(\frac{\hat{\tau}_\infty}{\varepsilon\delta^2})$, *where* $\widetilde{O}$ *hides polylog factors in* T, δ, *and* $|\Pi|$. *Since Theorem 4 guarantees* $\hat{\tau}_\infty \leqslant \tau_\infty^\star$, *the total number of expert demonstrations is thus upper bounded by* $\widetilde{O}(\frac{\tau_\infty^\star \wedge T}{\epsilon\delta^2})$. *Crucially, for large enough imitation learning horizon* T *(in particular, for* $T \geqslant \tau_\infty^\star$*), the number of required trajectories depends only logarithmically on the horizon* T.

Performance Guarantees. We are left to quantify the cost achieved by the learned controller. For the theoretical analysis, we first define the Q-function of the MPC controller $\bar{\pi}^{\mathsf{mpc}}$ w.r.t. the shrunk constraint sets $\bar{\mathcal{X}}, \bar{\mathcal{U}}$ and terminal set $\bar{\mathcal{X}}_{\mathsf{f}}$. Let $\bar{J}_t : \bar{\mathcal{X}}_0 \to \mathbb{R}$ be the t-step cost function of $\bar{\pi}^{\mathsf{mpc}}$, i.e., for $x_0 \in \bar{\mathcal{X}}_0$ and $\bar{x}_s \coloneqq \varphi_s(x_0; \bar{\pi}^{\mathsf{mpc}})$, $\bar{J}_t(x_0) \coloneqq \sum_{t=0}^{t-1} \ell(\bar{x}_s, \bar{\pi}^{\mathsf{mpc}}(\bar{x}_s))$, where $\ell(x,u) = x^\top Q x + u^\top R u$ denotes the stage cost. For $x \in \bar{\mathcal{X}}_0$, and an input $u \in \bar{\mathcal{U}}$ such that $Ax + Bu \in \bar{\mathcal{X}}_0$, define $\bar{Q}_t(x,u) \coloneqq \ell(x,u) + \bar{J}_{T-t-1}(Ax + Bu)$. Let $\tilde{J}_t$ be the t-step cost of the learned policy $\tilde{\pi}$ from FORWARD-SWITCH (defined in a similar way to $\bar{J}_t$). We now state the performance guarantee of FORWARD-SWITCH (proof in (Ahn et al., 2022)).

Theorem 6 (Performance Guarantee) *Let* $\delta, \varepsilon \in (0,1)$ *and assume the same conditions as Theorem 4. Then, under the same event* $\bar{\mathcal{E}}$ *as Theorem 4 and for any* x_0 *satisfying the events in (4.2), we have* $\tilde{J}_T(x_0) - \bar{J}_T(x_0) \leqslant O(\tau_\infty^\star \varepsilon)$, *where* $O(\cdot)$ *hides an absolute constant that depends on the system parameters and* $\tau_\infty^\star$ *is a system's constant independent of* T*—see Definition 3.*

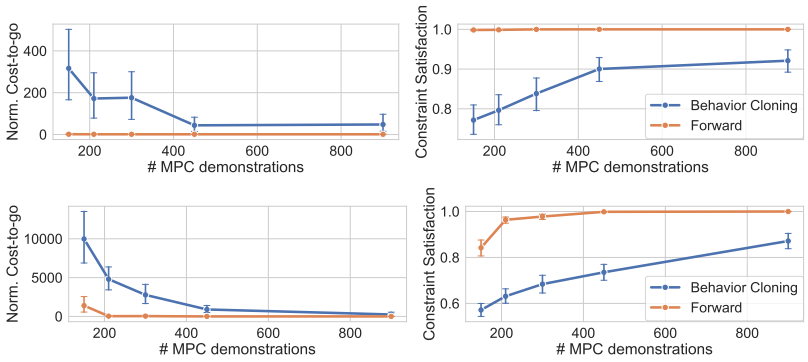

Figure 1: The results for the normalized cost-to-go and the constraint satisfaction ratio for the trajectory length $T = 30$. First row $\rightarrow$ results for $d = 3$; Second row $\rightarrow$ results for $d = 5$. For $d = 3$, the mean normalized cost of FORWARD is less than 1.2 for all settings, while that of BEHAVIOR CLONING is greater than 40 even with 900 MPC demonstrations. For $d = 5$, the normalized cost-to-go of FORWARD is less than 1.13 with 450 MPC demonstrations, while that of BEHAVIOR CLONING is higher than 240 even with 900 MPC demonstrations.

5. Experiments

In this section, we demonstrate our theoretical results for the MPC application through a set of experiments.

Experimental Setup. For $d \in \{3, 5\}$, we consider an open-loop unstable dynamical system $x_{t+1} = Ax_t + Bu_t$, where $A \in \mathbb{R}^{d\times d}$ is chosen as an upper triangular matrix whose diagonal entries are 1.1 and the upper diagonal entries are chosen from the uniform distribution over $[-2, 2]$, and $B \in \mathbb{R}^{d\times 1}$ is chosen as $[0\ 0\ \cdots\ 1]^\top$. We impose the constraints $x_t \in [-100, 100]^d$, $u_t \in [-10, 10]$ and choose the initial state distribution $\mathcal{D}$ as the uniform distribution over $[8, 10]^d$. We set the horizon N of MPC to be 20 and the number of imitation learning time steps T to be 30, and we use pyMPC (Forgione et al., 2020) for implementing MPC demonstrations. In the MPC optimization, we did not impose the terminal constraint.

To parametrize the policies we use a fully connected neural network with three hidden layers. Each layer has 50 neurons followed by ReLU activations. For optimization, we use the Adam optimizer with a learning rate of 0.001. We train the policies for 500 epochs.

Results. We first compare the performance of BEHAVIOR CLONING and FORWARD. For each algorithm, we measure the normalized cost $J^{\mathsf{algorithm}}(x_0)/J^{\mathsf{mpc}}(x_0)$ for 20 different test initial states x_0 sampled from $\mathcal{D}$. Moreover, we report the constraint satisfaction ratio along the test trajectories. We repeat each setting in the experiment for 50 times and report the 95% confidence intervals with error bars. The results are reported in Figure 1, and show that the performance of FORWARD is much higher than that of BEHAVIOR CLONING for this setting. In Figure 2, we visualize the extent of distribution shift by plotting the first two coordinates of the sample trajectories produced by each controller.

We now test the performance of FORWARD-SWITCH. For the same systems as before, we estimate $\hat{\tau}_\infty$ as per the procedure described in FORWARD-SWITCH, i.e., we check if the sample trajectory $\hat{x}_t^{(i)}$ lies in a subset of $\bar{\mathcal{O}}_\infty^{\mathsf{lqr}}$. Our estimated $\hat{\tau}_\infty$ for the $d = 5$ case is 12. In Figure 3, we report the mean normalized cost-to-go and the constraint satisfaction ratio of FORWARD-SWITCH. Our experiment indicates that FORWARD-SWITCH is indeed more sample-efficient in some situations.

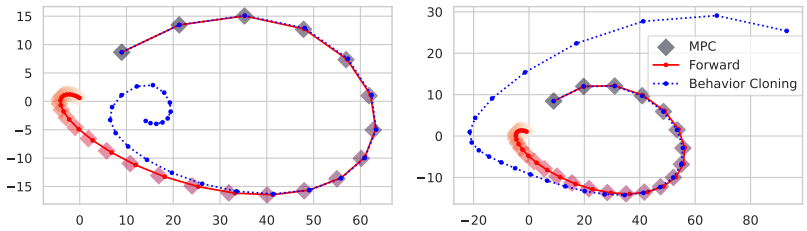

Figure 2: The first two coordinates of the sample trajectories produced by MPC, FORWARD, BEHAVIOR CLONING. The left plot is for $d = 3$, and the right column is for $d = 5$.

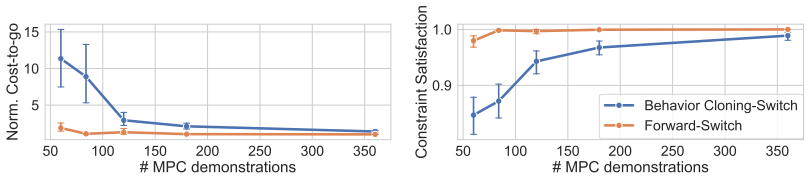

Figure 3: The performance comparison between FORWARD-SWITCH and its BEHAVIOR CLONING counterpart. The result is for $d = 5$. Here the estimated number of steps $\hat{\tau}_\infty$ to reach the positive invariant set $\bar{\mathcal{O}}_\infty^{\text{lqr}}$ is 12. Notably, FORWARD-SWITCH achieves the mean normalized cost-to-go of ≈ 1.034 with only 180 MPC demonstrations, while its BEHAVIOR CLONING counterpart achieves the mean normalized cost-to-go of ≈ 35 when trained using 210 MPC demonstrations.

We also compare the performance of FORWARD-SWITCH with its BEHAVIOR CLONING counterpart. For a fair comparison, we also train BEHAVIOR CLONING for $T = 12$ steps and then for time steps greater than 12, we employ the LQR controller. In Figure 3, we report the mean normalized cost-to-go and the constraint satisfaction ratio of FORWARD-SWITCH and its BEHAVIOR CLONING counterpart. Although $T = 12$ is smaller than the previous experiment setting where $T = 30$, we still see a noticeable difference in the performance between the two algorithms.

6. Conclusion

In this work, we leverage techniques from imitation learning to circumvent MPC's reliance on online optimization. More specifically, we adapt an interactive imitation learning algorithm called the forward training algorithm to take advantage of MPC's properties. When presented with a constrained linear system we show that our modified method learns a controller that stabilizes the dynamics, satisfies the state and input constraints, and achieves cost as good as that obtained by MPC. We validate our results through simulations and compare the modified forward training algorithm with other data-driven methods.

We conclude this paper with interesting future directions. An alternative approach to ours is to learn the value function instead of the policy. In particular, it is known that the MPC value function is convex and piecewise quadratic (Bemporad et al., 2002; Seron et al., 2003). It might be interesting to see whether such properties make the approach based on learning value functions more desirable. More broadly, whether the value of each expert demonstration can be used to improve performance of imitation learning algorithms would be of great interest. Lastly, combining our approach with a direct policy optimization approach (e.g., Chen et al. (2018)) would be of great practical interest, given that a direct policy optimization typically requires more samples.

Acknowledgement

Kwangjun Ahn, Zakaria Mhammedi, Horia Mania and Ali Jadbabaie were supported by the ONR grant (N00014-20-1-2394) and MIT-IBM Watson as well as a Vannevar Bush fellowship from Office of the Secretary of Defense. Zakaria Mhammedi was also supported by the ONR grant (N00014-20-1-2336). Zhang-Wei Hong was supported by the ONR MURI grant (N00014-22-1-2740). Kwangjun Ahn also acknowledges support from the Kwanjeong Educational Foundation.

Part of this work was done as Kwangjun Ahn's class project for 6.832: Underactuated Robotics at MIT, Spring 2022; Kwangjun Ahn thanks Russ Tedrake for constructive comments on the project. The authors thank Jack Umenberger and Sinho Chewi for very detailed comments regarding the theoretical results in the paper. The authors also thank Haoyuan Sun and Navid Azizan for fruitful discussions during the initial stage of this work.

References

Kwangjun Ahn, Zakaria Mhammedi, Horia Mania, Zhang-Wei Hong, and Ali Jadbabaie. Model predictive control via on-policy imitation learning. *arXiv preprint arXiv:2210.09206*, 2022.

Alessandro Alessio and Alberto Bemporad. A survey on explicit model predictive control. In *Nonlinear model predictive control*, pages 345–369. Springer, 2009.

Alberto Bemporad, Manfred Morari, Vivek Dua, and Efstratios N Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, 2002.

Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

Steven Chen, Kelsey Saulnier, Nikolay Atanasov, Daniel D Lee, Vijay Kumar, George J Pappas, and Manfred Morari. Approximating explicit model predictive control using constrained neural networks. In *American Control Conference*, pages 1520–1527. IEEE, 2018.

Paolo Falcone, Francesco Borrelli, Jahan Asgari, Hongtei Eric Tseng, and Davor Hrovat. Predictive active steering control for autonomous vehicle systems. *IEEE Transactions on control systems technology*, 15(3):566–580, 2007.

Marco Forgione, Dario Piga, and Alberto Bemporad. Efficient calibration of embedded MPC. In *Proc. of the 21st IFAC World Congress 2020, Berlin, Germany, July 12-17 2020*, 2020.

Pontus Giselsson, Minh Dang Doan, Tamás Keviczky, Bart De Schutter, and Anders Rantzer. Accelerated gradient methods and dual decomposition in distributed model predictive control. *Automatica*, 49(3):829 – 833, 2013. ISSN 0005-1098. doi: https://doi.org/10.1016/j.automatica.2013.01.009.

Michael Hertneck, Johannes Köhler, Sebastian Trimpe, and Frank Allgöwer. Learning an approximate model predictive controller with guarantees. *IEEE Control Systems Letters*, 2(3):543–548, 2018.

Ali Jadbabaie and John Hauser. Control of a thrust-vectored flying wing: a receding horizon—lpv approach. *International Journal of Robust and Nonlinear Control*, 12(9):869–896, 2002.

Juan L Jerez, Paul J Goulart, Stefan Richter, George a Constantinides, Eric C Kerrigan, and Manfred Morari. Embedded Online Optimization for Model Predictive Control at Megahertz Rates. *IEEE Transactions on Automatic Control*, 59(12):3238–3251, 2014. ISSN 0018-9286. doi: 10.1109/TAC.2014.2351991.

Benjamin Karg and Sergio Lucia. Efficient representation and approximation of model predictive control laws via deep learning. *IEEE Transactions on Cybernetics*, 50(9):3866–3878, 2020.

M. Kögel and R. Findeisen. A fast gradient method for embedded linear predictive control. In *Proceedings of the 18th IFAC World Congress*, pages 1362–1367, 2011.

Ilya Kolmanovsky and Elmer G Gilbert. Theory and computation of disturbance invariant sets for discrete-time linear systems. *Mathematical problems in engineering*, 4(4):317–367, 1998.

Scott Kuindersma, Robin Deits, Maurice Fallon, Andrés Valenzuela, Hongkai Dai, Frank Permenter, Twan Koolen, Pat Marion, and Russ Tedrake. Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot. *Autonomous robots*, 40(3):429–455, 2016.

S Lucia, M. Kögel, P. Zometa, D. E. Quevedo, and R. Findeisen. Predictive control, embedded cyberphysical systems and systems of systems – A perspective. *Annual Reviews in Control*, 41:193–207, 2016.

S Lucia, D. Navarro, O. Lucia, P. Zometa, and R. Findeisen. Optimized FPGA implementation of model predictive control using high level synthesis tools. *IEEE Transactions on Industrial Informatics*, 14(1):137–145, 2018.

Jacob Mattingley and Stephen Boyd. CVXGEN: a code generator for embedded convex optimization. *Optimization and Engineering*, 13(1):1–27, 2012.

Andreas Maurer and Massimiliano Pontil. Empirical Bernstein bounds and sample variance penalization. In *Conference on Learning Theory, COLT 2009*, Montreal, Canada, 18–21 June 2009.

David Q Mayne, María M Seron, and SV Raković. Robust model predictive control of constrained linear systems with bounded disturbances. *Automatica*, 41(2):219–224, 2005.

Zakaria Mhammedi, Dylan J Foster, Max Simchowitz, Dipendra Misra, Wen Sun, Akshay Krishnamurthy, Alexander Rakhlin, and John Langford. Learning the linear quadratic regulator from nonlinear observations. *Advances in Neural Information Processing Systems*, 33:14532–14543, 2020.

Manfred Morari and Jay H Lee. Model predictive control: past, present and future. *Computers & Chemical Engineering*, 23(4-5):667–682, 1999.

Richard M Murray, John Hauser, Ali Jadbabaie, Mark B Milam, Nicolas Petit, William B Dunbar, and Ryan Franz. Online control customization via optimization-based control. In *Software-Enabled Control, Information technology for dynamical systems*, pages 149–174. Wiley Online Library, 2003.

Daniel Pfrommer, Thomas T.C.K. Zhang, Stephen Tu, and Nikolai Matni. TaSIL: Taylor series imitation learning. *arXiv preprint arXiv:2205.14812*, 2022.

Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.

S.Joe Qin and Thomas A. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7):733–764, 2003.

James Blake Rawlings, David Q Mayne, and Moritz Diehl. *Model predictive control: theory, computation, and design*, volume 2. Nob Hill Publishing Madison, WI, 2017.

Stefan Richter, Colin Neil Jones, and Manfred Morari. Computational complexity certification for real-time MPC with input constraints based on the fast gradient method. *IEEE Transactions on Automatic Control*, 57(6):1391–1403, 2012. ISSN 00189286. doi: 10.1109/TAC.2011.2176389.

Ugo Rosolia and Francesco Borrelli. Learning model predictive control for iterative tasks. a data-driven control framework. *IEEE Transactions on Automatic Control*, 63(7):1883–1896, 2017.

Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 661–668. JMLR Workshop and Conference Proceedings, 2010.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3 (6):233–242, 1999.

Maria M Seron, Graham C Goodwin, and José A De Doná. Characterisation of receding horizon control for constrained linear systems. *Asian Journal of Control*, 5(2):271–286, 2003.

Wen Sun, Anirudh Vemula, Byron Boots, and Drew Bagnell. Provably efficient imitation learning from observation alone. In *International conference on machine learning*, pages 6036–6045. PMLR, 2019.

Mario Sznaier and Mark J Damborg. Suboptimal control of linear systems with state and control inequality constraints. In *26th IEEE conference on decision and control*, volume 26, pages 761–762. IEEE, 1987.

Stephen Tu, Alexander Robey, Tingnan Zhang, and Nikolai Matni. On the sample complexity of stability constrained imitation learning. *Proceedings of the 4rd Conference on Learning for Dynamics and Control, arXiv preprint arXiv:2102.09161*, 2022.

P. Zometa, M. Kögel, and R. Findeisen. μAO-MPC: A free code generation tool for embedded real-time linear model predictive control. In *Proceedings of the American Control Conference*, pages 5320–5325, June 2013.