Robotics: Science and Systems 2019
Freiburg im Breisgau, June 22-26, 2019

An Online Learning Approach to Model Predictive Control

Nolan Wagener,*# Ching-An Cheng,*# Jacob Sacks,† and Byron Boots*
Georgia Institute of Technology
{nolan.wagener, cacheng, jsacks}@gatech.edu, bboots@cc.gatech.edu

***Abstract*—Model predictive control (MPC) is a powerful technique for solving dynamic control tasks. In this paper, we show that there exists a close connection between MPC and online learning, an abstract theoretical framework for analyzing online decision making in the optimization literature. This new perspective provides a foundation for leveraging powerful online learning algorithms to design MPC algorithms. Specifically, we propose a new algorithm based on dynamic mirror descent (DMD), an online learning algorithm that is designed for non-stationary setups. Our algorithm, Dynamic Mirror Descent Model Predictive Control (DMD-MPC), represents a general family of MPC algorithms that includes many existing techniques as special instances. DMD-MPC also provides a fresh perspective on previous heuristics used in MPC and suggests a principled way to design new MPC algorithms. In the experimental section of this paper, we demonstrate the flexibility of DMD-MPC, presenting a set of new MPC algorithms on a simple simulated cartpole and a simulated and real-world aggressive driving task. A video of the real-world experiment can be found at https://youtu.be/vZST3v0_S9w.**

I. Introduction

Model predictive control (MPC) [20] is an effective tool for control tasks involving dynamic environments, such as helicopter aerobatics [1] and aggressive driving [30]. One reason for its success is the pragmatic principle it adopts in choosing controls: rather than wasting computational power to optimize a complicated controller for the full-scale problem (which may be difficult to accurately model), MPC instead optimizes a simple controller (e.g., an open-loop control sequence) over a shorter planning horizon that is just sufficient to make a sensible decision at the current moment. By alternating between optimizing the simple controller and applying its corresponding control on the real system, MPC results in a closed-loop policy that can handle modeling errors and dynamic changes in the environment.

Various MPC algorithms have been proposed, using tools ranging from constrained optimization techniques [7, 20, 27] to sampling-based techniques [30]. In this paper, we show that, while these algorithms were originally designed differently, if we view them through the lens of *online learning* [16], many of them actually follow the same general update rule. Online learning is an abstract theoretical framework for analyzing online decision making. Formally, it concerns iterative interactions between a learner and an environment over T rounds. At round t, the learner makes a decision $\tilde{\boldsymbol{\theta}}_t$ from some decision set $\boldsymbol{\Theta}$. The environment then chooses a loss function ℓ_t based on the learner's decision, and the learner suffers a cost $\ell_t(\tilde{\boldsymbol{\theta}}_t)$. In addition to seeing the decision's cost, the learner may be given additional information about the loss function (e.g., its gradient evaluated at $\tilde{\boldsymbol{\theta}}_t$) to aid in choosing the next decision $\tilde{\boldsymbol{\theta}}_{t+1}$. The learner's goal is to minimize the accumulated costs $\sum_{t=1}^{T} \ell_t(\tilde{\boldsymbol{\theta}}_t)$, e.g., by minimizing regret [16].

We find that the MPC process bears a strong similarity with online learning. At time t (i.e., round t), an MPC algorithm optimizes a controller (i.e., the decision) over some cost function (i.e., the per-round loss). To do so, it observes the cost of the initial controller (i.e., $\ell_t(\tilde{\boldsymbol{\theta}}_t)$), improves the controller, and executes a control based on the improved controller in the environment to get to the next state (which in turn defines the next per-round loss) with a new controller $\tilde{\boldsymbol{\theta}}_{t+1}$.

In view of this connection, we propose a generic framework, *DMD-MPC* (Dynamic Mirror Descent Model Predictive Control), for synthesizing MPC algorithms. DMD-MPC is based on a first-order online learning algorithm called dynamic mirror descent (DMD) [14], a generalization of mirror descent [4] for dynamic comparators. We show that several existing MPC algorithms [31, 32] are special cases of DMD-MPC, given specific choices of step sizes, loss functions, and regularization. Furthermore, we demonstrate how new MPC algorithms can be derived systematically from DMD-MPC with only mild assumptions on the regularity of the cost function. This allows us to even work with discontinuous cost functions (like indicators) and discrete controls. Thus, DMD-MPC offers a spectrum from which practitioners can easily customize new algorithms for their applications.

In the experiments, we apply DMD-MPC to design a range of MPC algorithms and study their empirical performance. Our results indicate the extra design flexibility offered by DMD-MPC does make a difference in practice; by properly selecting hyperparameters which are obscured in the previous approaches, we are able to improve the performance of existing algorithms. Finally, we apply DMD-MPC on a real-world AutoRally car platform [13] for autonomous driving tasks and show it can achieve competent performance.

Notation: As our discussions will involve planning horizons, for clarity, we use lightface to denote variables that are meant for a single time step, and boldface to denote the variables congregated across the MPC planning horizon. For example,

*Institute for Robotics and Intelligent Machines
†School of Electrical and Computer Engineering
#Equal contribution

we use $\hat{u}_t$ to denote the planned control at time t and $\hat{\boldsymbol{u}}_t \triangleq (\hat{u}_t, \dots, \hat{u}_{t+H-1})$ to denote an H-step planned control sequence starting from time t. We use a subscript to extract elements from a congregated variable; e.g., we use $\hat{u}_{t,h}$ to the denote the h^{th} element in $\hat{\boldsymbol{u}}_t$ (the subscript index starts from zero). All the variables in this paper are finite-dimensional.

II. An Online Learning Perspective on MPC

A. The MPC Problem Setup

Let $n,\ m \in \mathbb{N}_+$ be finite. We consider the problem of controlling a discrete-time stochastic dynamical system

$$x_{t+1} \sim f(x_t, u_t) \tag{1}$$

for some stochastic transition map $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$. At time t, the system is in state $x_t \in \mathbb{R}^n$. Upon the execution of control $u_t \in \mathbb{R}^m$, the system randomly transitions to the next state x_{t+1}, and an instantaneous cost $c(x_t, u_t)$ is incurred. Our goal is to design a state-feedback control law (i.e., a rule of choosing u_t based on x_t) such that the system exhibits good performance (e.g., accumulating low costs over T time steps).

In this paper, we adopt the MPC approach to choosing u_t: at state x_t, we imagine controlling a stochastic dynamics model $\hat{f}$ (which approximates our system f) for H time steps into the future. Our planned controls come from a control distribution $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ that is parameterized by some vector $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, where $\boldsymbol{\Theta}$ is the feasible parameter set. In each simulation (i.e., rollout), we sample[1] a control sequence $\hat{\boldsymbol{u}}_t$ from the control distribution $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ and recursively apply it to $\hat{f}$ to generate a predicted state trajectory $\hat{\boldsymbol{x}}_t \triangleq (\hat{x}_t, \hat{x}_{t+1}, \dots, \hat{x}_{t+H})$: let $\hat{x}_t = x_t$; for $\tau = t, \dots, t+H-1$, we set $\hat{x}_{\tau+1} \sim \hat{f}(\hat{x}_\tau, \hat{u}_\tau)$. More compactly, we can write the simulation process as

$$\hat{\boldsymbol{x}}_t \sim \hat{\boldsymbol{f}}(x_t, \hat{\boldsymbol{u}}_t) \tag{2}$$

in terms of some $\hat{\boldsymbol{f}}$ that is defined naturally according to the above recursion. Through these simulations, we desire to select a parameter $\boldsymbol{\theta}_t \in \boldsymbol{\Theta}$ that minimizes an MPC objective $\hat{J}(\boldsymbol{\pi}_{\boldsymbol{\theta}}; x_t)$, which aims to predict the performance of the system if we were to apply the control distribution $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ starting from x_t.[2] In other words, we wish to find the $\boldsymbol{\theta}_t$ that solves

$$\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \hat{J}(\boldsymbol{\pi}_{\boldsymbol{\theta}}; x_t). \tag{3}$$

Once $\boldsymbol{\theta}_t$ is decided, we then sample[3] $\hat{\boldsymbol{u}}_t$ from $\boldsymbol{\pi}_{\boldsymbol{\theta}_t}$, extract the first control $\hat{u}_t$, and apply it on the real dynamical system f in (1) (i.e., set $u_t = \hat{u}_t$) to go to the next state x_{t+1}. Because $\boldsymbol{\theta}_t$ is determined based on x_t, MPC is effectively state-feedback.

The motivation behind MPC is to use the MPC objective $\hat{J}$ to reason about the controls required to achieve desirable long-term behaviors. Consider the statistic

$$C(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t) \triangleq \textstyle\sum_{h=0}^{H-1} c(\hat{x}_{t+h}, \hat{u}_{t+h}) + c_{\text{end}}(\hat{x}_{t+H}), \tag{4}$$

[1] This can be sampled in either an open-loop or closed-loop fashion.

[2] $\hat{J}$ can be seen as a surrogate for the long-term performance of our controller. Typically, we set the planning horizon H to be much smaller than T to reduce the optimization difficulty and to mitigate modeling errors.

[3] This setup can also optimize deterministic policies, e.g., by defining $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ to be a Gaussian policy with the mean being the deterministic policy.

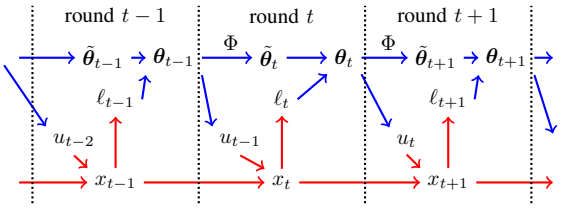


Fig. 1: Diagram of the online learning perspective, where blue and red denote the learner and the environment, respectively.

where c_{end} is a terminal cost function. A popular MPC objective is $\hat{J}(\boldsymbol{\pi}_{\boldsymbol{\theta}}; x_t) = \mathbb{E}[C(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t) \mid x_t, \boldsymbol{\pi}_{\boldsymbol{\theta}}, \hat{\boldsymbol{f}}]$, which estimates the expected H-step future costs. Later in Section III-A, we will discuss several MPC objectives and their properties.

Although the idea of MPC sounds intuitively promising, the optimization can only be approximated in practice (e.g., using an iterative algorithm like gradient descent), because (3) is often a stochastic program (like the example above) and the control command u_t needs to be computed at a high frequency. In consideration of this imperfection, it is common to heuristically *bootstrap* the previous approximate solution as the initialization to the current problem. Specifically, let $\boldsymbol{\theta}_{t-1}$ be the approximate solution to the previous problem and $\tilde{\boldsymbol{\theta}}_t$ denote the initial condition of $\boldsymbol{\theta}$ in solving (3). The bootstrapping step can then written as

$$\tilde{\boldsymbol{\theta}}_t = \Phi(\boldsymbol{\theta}_{t-1}) \tag{5}$$

by effectively defining a *shift operator* Φ (see Section A for details). Because the subproblems in (3) of two consecutive time steps share all control variables except for the first and the last ones, shifting the previous solution provides a warm start to (3) to amortize the computational complexity.

B. The Online Learning Perspective

As discussed, the iterative update process of MPC resembles the setup of online learning [16]. Here we provide the details to convert an MPC setup into an online learning problem. Recall from the introduction that online learning mainly consists of three components: the decision set, the learner's strategy for updating decisions, and the environment's strategy for updating per-round losses. We show the counterparts in MPC that correspond to each component below. Note that in this section we will overload the notation $\hat{J}(\boldsymbol{\theta}; x_t)$ to mean $\hat{J}(\boldsymbol{\pi}_{\boldsymbol{\theta}}; x_t)$.

We use the concept of per-round loss in online learning as a mechanism to measure the decision uncertainty in MPC, and propose the following identification (shown in Fig. 1) for the MPC setup described in the previous section: we set the rounds in online learning to synchronize with the time steps of our control system, set the decision set $\boldsymbol{\Theta}$ as the space of feasible parameters of the control distribution $\boldsymbol{\pi}_{\boldsymbol{\theta}}$, set the learner as the MPC algorithm which in round t outputs the decision $\tilde{\boldsymbol{\theta}}_t \in \boldsymbol{\Theta}$ and side information u_{t-1}, and set the per-round loss as

$$\ell_t(\cdot) = \hat{J}(\cdot\,; x_t). \tag{6}$$

In other words, in round t of this online learning setup, the learner plays a decision $\tilde{\boldsymbol{\theta}}_t$ along with a side information

u_{t-1} (based on the optimized solution $\boldsymbol{\theta}_{t-1}$ and the shift operator in (5)), the environment selects the per-round loss $\ell_t(\cdot) = \hat{J}(\cdot; x_t)$ (by applying u_{t-1} to the real dynamical system in (1) to transit the state to x_t), and finally the learner receives ℓ_t and incurs cost $\ell_t(\tilde{\boldsymbol{\theta}}_t)$ (which measures the sub-optimality of the future plan made by the MPC algorithm).

This online learning setup differs slightly from the standard setup in its separation of the decision $\tilde{\boldsymbol{\theta}}_t$ and the side information u_{t-1}; while our setup can be converted into a standard one that treats $\boldsymbol{\theta}_{t-1}$ as the sole decision played in round t, we adopt this explicit separation in order to emphasize that the variable part of the incurred cost $\ell_t(\tilde{\boldsymbol{\theta}}_t)$ pertains to only $\tilde{\boldsymbol{\theta}}_t$. That is, the learner cannot go back and revert the previous control u_{t-1} already applied on the system, but only uses ℓ_t to update the current and future controls $\hat{u}_t, \dots, \hat{u}_{t+H-1}$.

The performance of the learner in online learning (which by our identification is the MPC algorithm) is measured in terms of the accumulated costs $\sum_{t=1}^{T} \ell_t(\tilde{\boldsymbol{\theta}}_t)$. For problems in non-stationary setups, a normalized way to describe the accumulated costs in the online learning literature is through the concept of *dynamic regret* [14, 34], which is defined as

$$\text{D-Regret} = \textstyle\sum_{t=1}^{T} \ell_t(\tilde{\boldsymbol{\theta}}_t) - \sum_{t=1}^{T} \ell_t(\boldsymbol{\theta}_t^\star), \tag{7}$$

where $\boldsymbol{\theta}_t^\star \in \arg\min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \ell_t(\boldsymbol{\theta})$. Dynamic regret quantifies how suboptimal the played decisions $\tilde{\boldsymbol{\theta}}_1, \dots, \tilde{\boldsymbol{\theta}}_T$ are on the corresponding loss functions. In our proposed problem setup, the optimality concept associated with dynamic regret conveys a *consistency* criterion desirable for MPC: we would like to make a decision $\boldsymbol{\theta}_{t-1}$ at state x_{t-1} such that, after applying control u_{t-1} and entering the new state x_t, its shifted plan $\tilde{\boldsymbol{\theta}}_t$ remains close to optimal with respect to the new loss function ℓ_t. If the dynamics model $\hat{f}$ is accurate and the MPC algorithm is ideally solving (3), we can expect that bootstrapping the previous solution $\boldsymbol{\theta}_{t-1}$ through (5) into $\tilde{\boldsymbol{\theta}}_t$ would result in a small instantaneous gap $\ell_t(\tilde{\boldsymbol{\theta}}_t) - \ell_t(\boldsymbol{\theta}_t^\star)$ which is solely due to unpredictable future information (such as the stochasticity in the dynamical system). In other words, an online learning algorithm with small dynamic regret, if applied to our online learning setup, would produce a consistently optimal MPC algorithm with regard to the solution concept discussed above. However, we note that having small dynamic regret here does not directly imply good absolute performance on the control system, because the overall performance of the MPC algorithm is largely dependent on the form of the MPC objective $\hat{J}$ (e.g., through choice of H and accuracy of $\hat{f}$). Small dynamic regret more precisely means whether the plan produced by an MPC algorithm is consistent with the given MPC objective.

III. A Family of MPC Algorithms Based on Dynamic Mirror Descent

The online learning perspective on MPC suggests that good MPC algorithms can be designed from online learning algorithms that achieve small dynamic regret. This is indeed the case. We will show that a range of existing MPC algorithms are in essence applications of a classical online learning algorithm called dynamic mirror descent (DMD) [14]. DMD is a generalization of mirror descent [4] to problems involving dynamic comparators (in this case, the $\{\boldsymbol{\theta}_t^\star\}$ in dynamic regret in (7)). In round t, DMD applies the following update rule:

$$\boldsymbol{\theta}_t = \arg\min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \langle \gamma_t \boldsymbol{g}_t, \boldsymbol{\theta}\rangle + D_\psi(\boldsymbol{\theta} \| \tilde{\boldsymbol{\theta}}_t), \quad \tilde{\boldsymbol{\theta}}_{t+1} = \Phi(\boldsymbol{\theta}_t) \tag{8}$$

where $\boldsymbol{g}_t = \nabla \ell_t(\tilde{\boldsymbol{\theta}}_t)$ (which can be replaced by unbiased sampling if $\nabla \ell_t(\tilde{\boldsymbol{\theta}}_t)$ is an expectation), Φ is called the *shift model*,[4] $\gamma_t > 0$ is the step size, and for some $\boldsymbol{\theta}, \boldsymbol{\theta}' \in \boldsymbol{\Theta}$, $D_\psi(\boldsymbol{\theta} \| \boldsymbol{\theta}') \triangleq \psi(\boldsymbol{\theta}) - \psi(\boldsymbol{\theta}') - \langle \nabla\psi(\boldsymbol{\theta}'), \boldsymbol{\theta} - \boldsymbol{\theta}'\rangle$ is the Bregman divergence generated by a strictly convex function ψ on $\boldsymbol{\Theta}$.

The first step of DMD in (8) is reminiscent of the proximal update in the usual mirror descent algorithm. It can be thought of as an optimization step where the Bregman divergence acts as a regularization to keep $\boldsymbol{\theta}$ close to $\tilde{\boldsymbol{\theta}}_t$. Although $D_\psi(\boldsymbol{\theta} \| \boldsymbol{\theta}')$ is not necessarily a metric (since it may not be symmetric), it is still useful to view it as a distance between $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$. Indeed, familiar examples of the Bregman divergence include the squared Euclidean distance and KL divergence[5] [3].

The second step of DMD in (8) uses the shift model Φ to anticipate the optimal decision for the next round. In the context of MPC, a natural choice for the shift model is the shift operator in (5) defined previously in Section II-A (hence the same notation), because the per-round losses in two consecutive rounds here concern problems with shifted time indices. Hall and Willett [14] show that the dynamic regret of DMD scales with how much the optimal decision sequence $\{\boldsymbol{\theta}_t^\star\}$ deviates from Φ (i.e., $\sum_t \|\boldsymbol{\theta}_{t+1}^\star - \Phi(\boldsymbol{\theta}_t^\star)\|$), which is proportional to the unpredictable elements of the problem.

Algorithm 1: Dynamic Mirror Descent MPC (DMD-MPC)

for $t = 1, 2, \dots, T$ **do**
- $\ell_t(\cdot) = \hat{J}(\cdot; x_t)$
- $\boldsymbol{\theta}_t = \arg\min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \langle \gamma_t \nabla \ell_t(\tilde{\boldsymbol{\theta}}_t), \boldsymbol{\theta}\rangle + D_\psi(\boldsymbol{\theta} \| \tilde{\boldsymbol{\theta}}_t)$
- Sample $\hat{\boldsymbol{u}}_t \sim \boldsymbol{\pi}_{\boldsymbol{\theta}_t}$ and set $u_t = \hat{u}_t$
- Sample $x_{t+1} \sim f(x_t, u_t)$
- $\tilde{\boldsymbol{\theta}}_{t+1} = \Phi(\boldsymbol{\theta}_t)$

end

Applying DMD in (8) to the online learning problem described in Section II-B leads to an MPC algorithm shown in Algorithm 1, which we call *DMD-MPC*. More precisely, DMD-MPC represents a family of MPC algorithms in which a specific instance is defined by a choice of:

1) the MPC objective $\hat{J}$ in (6),
2) the form of the control distribution $\boldsymbol{\pi}_{\boldsymbol{\theta}}$, and
3) the Bregman divergence D_ψ in (8).

Thus, we can use DMD-MPC as a generic strategy for synthesizing MPC algorithms. In the following, we use this recipe to recreate several existing MPC algorithms and demonstrate new MPC algorithms that naturally arise from this framework.

[4] In [14], Φ is called a *dynamical model*, but it is not the same as the dynamics of our control system. We therefore rename it to avoid confusion.

[5] For probability distributions p and q over a random variable x, the KL divergence is defined as $\mathrm{KL}(p \,\|\, q) \triangleq \mathbb{E}_{x\sim p}[\log(p(x)/q(x))]$.

A. Loss Functions

We discuss several definitions of the per-round loss ℓ_t, which all result from the formulation in (6) but with different $\hat{J}$. These loss functions are based on the statistic $C(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t)$ defined in (4) which measures the H-step accumulated cost of a given trajectory. For transparency of exposition, we will suppose henceforth that the control distribution $\boldsymbol{\pi_\theta}$ is open-loop[6]; similar derivations follow naturally for closed-loop control distributions. For convenience of practitioners, we also provide expressions of their gradients in terms of the likelihood-ratio derivative[7] [12]. For some function $L_t(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t)$, all these gradients shall have the form

$$\nabla \ell_t(\boldsymbol{\theta}) = \mathbb{E}_{\hat{\boldsymbol{u}}_t \sim \boldsymbol{\pi_\theta}} \mathbb{E}_{\hat{\boldsymbol{x}}_t \sim \hat{\boldsymbol{f}}(x_t, \hat{\boldsymbol{u}}_t)} [L_t(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t) \nabla_{\boldsymbol{\theta}} \log \boldsymbol{\pi_\theta}(\hat{\boldsymbol{u}}_t)]. \quad (9)$$

In short, we will denote $\mathbb{E}_{\hat{\boldsymbol{u}}_t \sim \boldsymbol{\pi_\theta}} \mathbb{E}_{\hat{\boldsymbol{x}}_t \sim \hat{\boldsymbol{f}}(x_t, \hat{\boldsymbol{u}}_t)}$ as $\mathbb{E}_{\boldsymbol{\pi_\theta}, \hat{\boldsymbol{f}}}$. These gradients in practice are approximated by finite samples.

1) Expected Cost: The most commonly used MPC objective is the H-step expected accumulated cost function under model dynamics, because it directly estimates the expected long-term behavior when the dynamics model $\hat{f}$ is accurate and H is large enough. Its per-round loss function is[8]

$$\ell_t(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\pi_\theta}, \hat{\boldsymbol{f}}} [C(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t)] \quad (10)$$

$$\nabla \ell_t(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\pi_\theta}, \hat{\boldsymbol{f}}} [C(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t) \nabla_{\boldsymbol{\theta}} \log \boldsymbol{\pi_\theta}(\hat{\boldsymbol{u}}_t)]. \quad (11)$$

2) Expected Utility: Instead of optimizing for average cost, we may care to optimize for some preference related to the trajectory cost C, such as having the cost be below some threshold. This idea can be formulated as a *utility* that returns a normalized score related to the preference for a given trajectory cost $C(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t)$. Specifically, suppose that C is lower bounded by zero[9] and at some round t define the utility $U_t : \mathbb{R}_+ \to [0, 1]$ (i.e., $U_t : C(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t) \mapsto U_t(C(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t))$) to be a function with the following properties: $U_t(0) = 1$, U_t is monotonically decreasing, and $\lim_{z \to +\infty} U_t(z) = 0$. These are sensible properties since we attain maximum utility when we have zero cost, the utility never increases with the cost, and the utility approaches zero as the cost increases without bound. We then define the per-round loss as

$$\ell_t(\boldsymbol{\theta}) = -\log \mathbb{E}_{\boldsymbol{\pi_\theta}, \hat{\boldsymbol{f}}} [U_t(C(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t))] \quad (12)$$

$$\nabla \ell_t(\boldsymbol{\theta}) = -\frac{\mathbb{E}_{\boldsymbol{\pi_\theta}, \hat{\boldsymbol{f}}} [U_t(C(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t)) \nabla_{\boldsymbol{\theta}} \log \boldsymbol{\pi_\theta}(\hat{\boldsymbol{u}}_t)]}{\mathbb{E}_{\boldsymbol{\pi_\theta}, \hat{\boldsymbol{f}}} [U_t(C(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t))]}. \quad (13)$$

The gradient in (13) is particularly appealing when estimated with samples. Suppose we sample N control sequences $\hat{\boldsymbol{u}}_t^1, \dots, \hat{\boldsymbol{u}}_t^N$ from $\boldsymbol{\pi_\theta}$ and (for the sake of compactness) sample one state trajectory from $\hat{\boldsymbol{f}}$ for each corresponding control sequence, resulting in $\hat{\boldsymbol{x}}_t^1, \dots, \hat{\boldsymbol{x}}_t^N$. Then the estimate of (13) is a convex combination of gradients:

$$\nabla \ell_t(\boldsymbol{\theta}) \approx -\textstyle\sum_{i=1}^N w_i \nabla_{\boldsymbol{\theta}} \log \boldsymbol{\pi_\theta}(\hat{\boldsymbol{u}}_t^i),$$

where $w_i = \frac{U_t(C_i)}{\sum_{j=1}^N U_t(C_j)}$ and $C_i = C(\hat{\boldsymbol{x}}_t^i, \hat{\boldsymbol{u}}_t^i)$, for $i = 1, \dots, N$. We see that each weight w_i is computed by considering the relative utility of its corresponding trajectory. A cost C_i with high relative utility will push its corresponding weight w_i closer to one, whereas a low relative utility will cause w_i to be close to zero, effectively rejecting the corresponding sample.

We give two examples of utilities and their related losses.

a) Probability of Low Cost: For example, we may care about the system being below some cost threshold as often as possible. To encode this preference, we can use the threshold utility $U_t(C) \triangleq \mathbf{1}\{C \leq C_{t,\max}\}$, where $\mathbf{1}\{\cdot\}$ is the indicator function and $C_{t,\max}$ is a threshold parameter. Under this choice, the loss and its gradient become

$$\begin{aligned} \ell_t(\boldsymbol{\theta}) &= -\log \mathbb{E}_{\boldsymbol{\pi_\theta}, \hat{\boldsymbol{f}}} [\mathbf{1}\{C(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t) \leq C_{t,\max}\}] \quad (14) \\ &= -\log \mathbb{P}_{\boldsymbol{\pi_\theta}, \hat{\boldsymbol{f}}} (C(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t) \leq C_{t,\max}) \\ \nabla \ell_t(\boldsymbol{\theta}) &= -\frac{\mathbb{E}_{\boldsymbol{\pi_\theta}, \hat{\boldsymbol{f}}} [\mathbf{1}\{C(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t) \leq C_{t,\max}\} \nabla_{\boldsymbol{\theta}} \log \boldsymbol{\pi_\theta}(\hat{\boldsymbol{u}}_t)]}{\mathbb{E}_{\boldsymbol{\pi_\theta}, \hat{\boldsymbol{f}}} [\mathbf{1}\{C(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t) \leq C_{t,\max}\}]}. \quad (15) \end{aligned}$$

As we can see, this loss function also gives the probability of achieving cost below some threshold. As a result (Fig. 2a), costs below $C_{t,\max}$ are treated the same in terms of the utility. This can potentially make optimization easier since we are trying to make good trajectories as likely as possible instead of finding the best trajectories as in (10).

However, if the threshold $C_{t,\max}$ is set too low and the gradient is estimated with samples, the gradient estimate may have high variance due to the large number of rejected samples. Because of this, in practice, the threshold is set adaptively, e.g., as the largest cost of the top *elite fraction* of the sampled trajectories with smallest costs [6]. This allows the controller to make the best sampled trajectories more likely and therefore improve the controller.

b) Exponential Utility: We can also opt for a continuous surrogate of the indicator function, in this case the exponential utility $U_t(C) \triangleq \exp(-\frac{1}{\lambda} C)$, where $\lambda > 0$ is a scaling parameter. Unlike the indicator function, the exponential utility provides nonzero feedback for any given cost and allows us to discriminate between costs (i.e., if $C_1 > C_2$, then $U_t(C_1) < U_t(C_2)$), as shown in Fig. 2b. Furthermore, λ acts as a continuous alternative to $C_{t,\max}$ and dictates how quickly or slowly U_t decays to zero, which in a soft way determines the cutoff point for rejecting given costs.

Under this choice, the loss and its gradient become

$$\begin{aligned} \ell_t(\boldsymbol{\theta}) &= -\log \mathbb{E}_{\boldsymbol{\pi_\theta}, \hat{\boldsymbol{f}}} \left[\exp\left(-\frac{1}{\lambda} C(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t) \right) \right] \quad (16) \\ \nabla \ell_t(\boldsymbol{\theta}) &= -\frac{\mathbb{E}_{\boldsymbol{\pi_\theta}, \hat{\boldsymbol{f}}} \left[\exp\left(-\frac{1}{\lambda} C(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t) \right) \nabla_{\boldsymbol{\theta}} \log \boldsymbol{\pi_\theta}(\hat{\boldsymbol{u}}_t) \right]}{\mathbb{E}_{\boldsymbol{\pi_\theta}, \hat{\boldsymbol{f}}} \left[\exp\left(-\frac{1}{\lambda} C(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t) \right) \right]}. \quad (17) \end{aligned}$$

The loss function in (16) is also known as the risk-seeking objective in optimal control [28]; this classical interpretation is based on a Taylor expansion of (16) showing

$$\lambda \ell_t(\boldsymbol{\theta}) \approx \mathbb{E}_{\boldsymbol{\pi_\theta}, \hat{\boldsymbol{f}}} [C(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t)] - \tfrac{1}{\lambda} \mathbb{V}_{\boldsymbol{\pi_\theta}, \hat{\boldsymbol{f}}} [C(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t)]$$

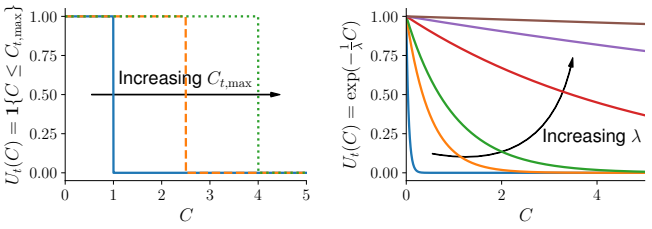
when λ is large, where $\mathbb{V}_{\boldsymbol{\pi_\theta}, \hat{\boldsymbol{f}}} [C(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t)]$ is the variance of $C(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t)$. Here we derive (16) from a different perspective

[6]Note again that even while using open-loop control distributions, the overall control law of MPC is state-feedback.

[7]We assume the control distribution is sufficiently regular with respect to its parameter so that the likelihood-ratio derivative rule holds.

[8]In experiments, we subtract the empirical average of the sampled costs from C in (11) to reduce the variance, at the cost of a small amount of bias.

[9]If this is not the case, let $c_{\min} \triangleq \inf_{\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t} C(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t)$, which we assume is finite. We can then replace C with $\tilde{C}(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t) \triangleq C(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t) - c_{\min}$.



(a) Threshold utility (b) Exponential utility

Fig. 2: Visualization of different utilities.

that treats it as a continuous approximation of (14). The use of exponential transformations to approximate indicators is a common machine-learning trick (like the Chernoff bound [8]).

B. Algorithms

We instantiate DMD-MPC with different choices of loss function, control distribution, and Bregman divergence as concrete examples to showcase the flexibility of our framework. In particular, we are able to recover well-known MPC algorithms as special cases of Algorithm 1.

Our discussions below are organized based on the class of Bregman divergences used in (8), and the following algorithms are derived assuming that the control distribution is a sequence of independent distributions. That is, we suppose $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ is a probability density/mass function that factorizes as

$$\boldsymbol{\pi}_{\boldsymbol{\theta}}(\hat{\boldsymbol{u}}_t) = \textstyle\prod_{h=0}^{H-1} \pi_{\theta_h}(\hat{u}_{t,h}), \tag{18}$$

and $\boldsymbol{\theta} = (\theta_0, \theta_1, \dots, \theta_{H-1})$ for some *basic control distribution* π_θ parameterized by $\theta \in \Theta$, where Θ denotes the feasible set for the basic control distribution. For control distributions in the form of (18), the shift operator Φ in (5) would set $\tilde{\boldsymbol{\theta}}_t$ by identifying $\tilde{\theta}_{t,h} = \theta_{t-1,h+1}$ for $h = 0, \dots, H-2$, and initializing the final parameter as either $\tilde{\theta}_{t,H-1} = \tilde{\theta}_{t,H-2}$ or $\tilde{\theta}_{t,H-1} = \bar{\theta}$ for some default parameter $\bar{\theta}$.

1) Quadratic Divergence: We start with perhaps the most common Bregman divergence: the quadratic divergence. That is, we suppose the Bregman divergence in (8) has a quadratic form [10] $D_\psi(\boldsymbol{\theta} \| \boldsymbol{\theta}') \triangleq \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}')^\mathsf{T} \boldsymbol{A} (\boldsymbol{\theta} - \boldsymbol{\theta}')$ for some positive-definite matrix $\boldsymbol{A}$. Below we discuss different choices of $\boldsymbol{A}$ and their corresponding update rules.

a) Projected Gradient Descent: This basic update rule is a special case when $\boldsymbol{A}$ is the identity matrix. Equivalently, the update can be written as $\boldsymbol{\theta}_t = \arg\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \|\boldsymbol{\theta} - (\tilde{\boldsymbol{\theta}}_t - \gamma_t \boldsymbol{g}_t)\|^2$.

b) Natural Gradient Descent: We can recover the natural gradient descent algorithm [2] by defining $\boldsymbol{A} = \mathcal{F}(\tilde{\boldsymbol{\theta}}_t)$ where

$$\mathcal{F}(\tilde{\boldsymbol{\theta}}_t) = \mathbb{E}_{\boldsymbol{\pi}_{\tilde{\boldsymbol{\theta}}_t}} \left[\nabla_{\tilde{\boldsymbol{\theta}}_t} \log \boldsymbol{\pi}_{\tilde{\boldsymbol{\theta}}_t}(\hat{\boldsymbol{u}}_t) \nabla_{\tilde{\boldsymbol{\theta}}_t} \log \boldsymbol{\pi}_{\tilde{\boldsymbol{\theta}}_t}(\hat{\boldsymbol{u}}_t)^\mathsf{T} \right]$$

is the Fisher information matrix. This rule uses the natural Riemannian metric of distributions to normalize the effects of different parameterizations of the same distribution [25].

c) Quadratic Problems: While the above two update rules are quite general, we can further specialize the Bregman divergence to achieve faster learning when the per-round loss function can be shown to be quadratic. This happens, for instance, when the MPC problem in (3) is an LQR or LEQR problem[11] [11]. That is, if

$$\ell_t(\boldsymbol{\theta}) = \tfrac{1}{2} \boldsymbol{\theta}^\mathsf{T} \boldsymbol{R}_t \boldsymbol{\theta} + \boldsymbol{r}_t^\mathsf{T} \boldsymbol{\theta} + \text{const.}$$

for some constant vector $\boldsymbol{r}_t$ and positive definite matrix $\boldsymbol{R}_t$, we can set $\boldsymbol{A} = \boldsymbol{R}_t$ and $\gamma_t = 1$, making $\boldsymbol{\theta}_t$ given by the first step of (8) correspond to the optimal solution to ℓ_t (i.e., the solution of LQR/LEQR). The particular values of $\boldsymbol{R}_t$ and $\boldsymbol{r}_t$ for each of LQR and LEQR are derived in Section D.

2) KL Divergence and the Exponential Family: We show that for control distributions in the exponential family [23], the Bregman divergence in (8) can be set to the KL divergence, which is a natural way to measure distances between distributions. Toward this end, we review the basics of the exponential family. We say a distribution p_η with natural parameter η of random variable u belongs to the *exponential family* if its probability density/mass function satisfies $p_\eta(u) = \rho(u) \exp(\langle \eta, \phi(u) \rangle - A(\eta))$, where $\phi(u)$ is the sufficient statistics, $\rho(u)$ is the carrier measure, and $A(\eta) = \log \int \rho(u) \exp(\langle \eta, \phi(u) \rangle) \, du$ is the log-partition function. The distribution p_η can also be described by its expectation parameter $\mu \triangleq \mathbb{E}_{p_\eta}[\phi(u)]$, and there is a duality between the two parameterizations: $\mu = \nabla A(\eta)$ and $\eta = \nabla A^*(\mu)$, where $A^*(\mu) = \sup_{\eta \in \mathcal{H}} \langle \eta, \mu \rangle - A(\eta)$ is the Legendre transformation of A and $\mathcal{H} = \{\eta : A(\eta) < +\infty\}$. That is, $\nabla A = (\nabla A^*)^{-1}$. The duality results in the property below.

Fact 1. [23] $\mathrm{KL}(p_\eta \,\|\, p_{\eta'}) = D_A(\eta' \| \eta) = D_{A^*}(\mu \| \mu')$.

We can use Fact 1 to define the Bregman divergence in (8) to optimize a control distribution $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ in the exponential family:

- if $\boldsymbol{\theta}$ is an expectation parameter, we can set $D_\psi(\boldsymbol{\theta} \| \tilde{\boldsymbol{\theta}}_t) \triangleq \mathrm{KL}(\boldsymbol{\pi}_{\boldsymbol{\theta}} \,\|\, \boldsymbol{\pi}_{\tilde{\boldsymbol{\theta}}_t})$, or
- if $\boldsymbol{\theta}$ is a natural parameter, we can set $D_\psi(\boldsymbol{\theta} \| \tilde{\boldsymbol{\theta}}_t) \triangleq \mathrm{KL}(\boldsymbol{\pi}_{\tilde{\boldsymbol{\theta}}_t} \,\|\, \boldsymbol{\pi}_{\boldsymbol{\theta}})$.

We demonstrate some examples using this idea below.

a) Expectation Parameters and Categorical Distributions: We first discuss the case where $\boldsymbol{\theta}$ is an expectation parameter and the first step in (8) is

$$\boldsymbol{\theta}_t = \arg\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \langle \gamma_t \boldsymbol{g}_t, \boldsymbol{\theta} \rangle + \mathrm{KL}(\boldsymbol{\pi}_{\boldsymbol{\theta}} \,\|\, \boldsymbol{\pi}_{\tilde{\boldsymbol{\theta}}_t}). \tag{19}$$

To illustrate, we consider an MPC problem with a *discrete* control space $\{1, 2, \dots, m\}$ and use the categorical distribution as the basic control distribution in (18), i.e., we set $\pi_{\theta_h} = \mathrm{Cat}(\theta_h)$, where $\theta_h \in \Delta^m$ is the probability of choosing each control among $\{1, 2, \dots, m\}$ at the h^{th} predicted time step and Δ^m denotes the probability simplex in $\mathbb{R}^m$. This parameterization choice makes $\boldsymbol{\theta}$ an expectation parameter of $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ that corresponds to sufficient statistics given by indicator functions. With the structure of (9), the update direction is

$$g_{t,h} = \mathbb{E}_{\boldsymbol{\pi}_{\tilde{\boldsymbol{\theta}}_t}, \hat{\boldsymbol{f}}} \left[L_t(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t) e_{\hat{u}_{t,h}} \oslash \tilde{\theta}_{t,h} \right] \quad (h = 0, 1, \dots, H-1)$$

where $\tilde{\theta}_{t,h}$ and $g_{t,h}$ are the h^{th} elements of $\tilde{\boldsymbol{\theta}}_t$ and $\boldsymbol{g}_t$, respectively, $e_{\hat{u}_{t,h}} \in \mathbb{R}^m$ has 0 for each element except at index $\hat{u}_{t,h}$

[10] This is generated by defining $\psi(\boldsymbol{\theta}) \triangleq \boldsymbol{\theta}^\mathsf{T} \boldsymbol{A} \boldsymbol{\theta} / 2$.

[11] The dynamics model $\hat{f}$ is linear, the step cost c is quadratic, the per-round loss ℓ_t is (10), and the basic control distribution is a Dirac-delta distribution.

where it is 1, and $\oslash$ denotes elementwise division. Update (19) then becomes the exponentiated gradient algorithm [16]:

$$\theta_{t,h} = \frac{1}{Z_{t,h}} \tilde{\theta}_{t,h} \odot \exp(-\gamma_t g_{t,h}) \quad (h = 0, 1, \ldots, H-1) \tag{20}$$

where $\theta_{t,h}$ is the h^{th} element of $\boldsymbol{\theta}_t$, $Z_{t,h}$ is the normalizer for $\theta_{t,h}$, and $\odot$ denotes elementwise multiplication. That is, instead of applying an additive gradient step to the parameters, the update in (19) exponentiates the gradient and performs elementwise multiplication. This does a better job of accounting for the geometry of the problem, and makes projection a simple operation of normalizing a distribution.

b) Natural Parameters and Gaussian Distributions: Alternatively, we can set $\boldsymbol{\theta}$ as a natural parameter and use

$$\boldsymbol{\theta}_t = \arg\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \langle \gamma_t \boldsymbol{g}_t, \boldsymbol{\theta} \rangle + \mathrm{KL}(\boldsymbol{\pi}_{\tilde{\boldsymbol{\theta}}_t} \,\|\, \boldsymbol{\pi}_{\boldsymbol{\theta}}) \tag{21}$$

as the first step in (8). In particular, we show that, with (21), the structure of the likelihood-ratio derivative in (9) can be leveraged to design an efficient update. The main idea follows from the observation that when the gradient is computed through (9) and $\tilde{\boldsymbol{\theta}}_t$ is the natural parameter, we can write

$$\boldsymbol{g}_t = \nabla \ell_t(\tilde{\boldsymbol{\theta}}_t) = \mathbb{E}_{\boldsymbol{\pi}_{\tilde{\boldsymbol{\theta}}_t}, \hat{\boldsymbol{f}}}[L_t(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t)(\phi(\hat{\boldsymbol{u}}_t) - \tilde{\boldsymbol{\mu}}_t)] \tag{22}$$

where $\tilde{\boldsymbol{\mu}}_t$ is the expectation parameter of $\tilde{\boldsymbol{\theta}}_t$ and ϕ is the sufficient statistics of the control distribution. We combine the factorization in (22) with a property of the proximal update below (proven in Section C) to derive our algorithm.

Proposition 1. *Let g_t be an update direction. Let $\mathcal{M}$ be the image of $\mathcal{H}$ under ∇A. If $\mu_t - \gamma_t g_t \in \mathcal{M}$ and $\eta_{t+1} = \arg\min_{\eta \in \mathcal{H}} \langle \gamma_t g_t, \eta \rangle + D_A(\eta \| \eta_t)$, then $\mu_{t+1} = \mu_t - \gamma_t g_t$.*[12]

We find that, under the assumption[13] in Proposition 1, the update rule in (21) becomes

$$\boldsymbol{\mu}_{t+1} = (1 - \gamma_t)\tilde{\boldsymbol{\mu}}_t + \gamma_t \mathbb{E}_{\boldsymbol{\pi}_{\tilde{\boldsymbol{\theta}}_t}, \hat{\boldsymbol{f}}}[L_t(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t)\phi(\hat{\boldsymbol{u}}_t)]. \tag{23}$$

In other words, when $\gamma_t \in [0, 1]$, the update to the expectation parameter $\boldsymbol{\mu}_t$ in (8) is simply a convex combination of the sufficient statistics and the previous expectation parameter $\tilde{\boldsymbol{\mu}}_t$.

We provide a concrete example of an MPC algorithm that follows from (23). Let us consider a continuous control space and use the Gaussian distribution as the basic control distribution in (18), i.e., we set $\pi_{\theta_h}(\hat{u}_{t,h}) = \mathcal{N}(\hat{u}_{t,h}; m_h, \Sigma_h)$ for some mean vector m_h and covariance matrix Σ_h. For π_{θ_h}, we can choose sufficient statistics $\phi(\hat{u}_{t,h}) = (\hat{u}_{t,h}, \hat{u}_{t,h}\hat{u}_{t,h}^{\mathsf{T}})$, which results in the expectation parameter $\mu_h = (m_h, S_h)$ and the natural parameter $\eta_h = (\Sigma_h^{-1} m_h, -\frac{1}{2}\Sigma_h^{-1})$, where $S_h \triangleq \Sigma_h + m_h m_h^{\mathsf{T}}$. Let us set θ_h as the natural parameter. Then (21) is equivalent to the update rule for $h = 0, \ldots, H-1$:

$$\begin{aligned} m_{t,h} &= (1 - \gamma_t)\tilde{m}_{t,h} + \gamma_t \mathbb{E}_{\boldsymbol{\pi}_{\tilde{\boldsymbol{\theta}}_t}, \hat{\boldsymbol{f}}}[L_t(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t)\hat{u}_{t,h}] \\ S_{t,h} &= (1 - \gamma_t)\tilde{S}_{t,h} + \gamma_t \mathbb{E}_{\boldsymbol{\pi}_{\tilde{\boldsymbol{\theta}}_t}, \hat{\boldsymbol{f}}}\big[L_t(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t)\hat{u}_{t,h}\hat{u}_{t,h}^{\mathsf{T}}\big]. \end{aligned} \tag{24}$$

[12] A similar proposition can be found for (19).

[13] If $\mu_t - \gamma_t g_t$ is not in $\mathcal{M}$, the update in (21) needs to perform a projection, the form of which is algorithm dependent.

Several existing algorithms are special cases of (24).

- *Cross-entropy method (CEM)* [6]:
  If ℓ_t is set to (14) and $\gamma_t = 1$, then (24) becomes
  $$m_{t,h} = \frac{\mathbb{E}_{\boldsymbol{\pi}_{\tilde{\boldsymbol{\theta}}_t}, \hat{\boldsymbol{f}}}[\mathbf{1}\{C(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t) \le C_{t,\max}\}\hat{u}_{t,h}]}{\mathbb{E}_{\boldsymbol{\pi}_{\tilde{\boldsymbol{\theta}}_t}, \hat{\boldsymbol{f}}}[\mathbf{1}\{C(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t) \le C_{t,\max}\}]}$$
  $$S_{t,h} = \frac{\mathbb{E}_{\boldsymbol{\pi}_{\tilde{\boldsymbol{\theta}}_t}, \hat{\boldsymbol{f}}}\big[\mathbf{1}\{C(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t) \le C_{t,\max}\}\hat{u}_{t,h}\hat{u}_{t,h}^{\mathsf{T}}\big]}{\mathbb{E}_{\boldsymbol{\pi}_{\tilde{\boldsymbol{\theta}}_t}, \hat{\boldsymbol{f}}}[\mathbf{1}\{C(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t) \le C_{t,\max}\}]},$$
  which resembles the update rule of the cross-entropy method for Gaussian distributions [6]. The only difference is that the second-order moment matrix $S_{t,h}$ is updated instead of the covariance matrix $\Sigma_{t,h}$.
- *Model-predictive path integral (MPPI)* [31]:
  If we choose ℓ_t as the exponential utility, as in (16), and do not update the covariance, the update rule becomes
  $$m_{t,h} = (1 - \gamma_t)\tilde{m}_{t,h} + \gamma_t \frac{\mathbb{E}_{\boldsymbol{\pi}_{\tilde{\boldsymbol{\theta}}_t}, \hat{\boldsymbol{f}}}\big[e^{-\frac{1}{\lambda}C(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t)}\hat{u}_{t,h}\big]}{\mathbb{E}_{\boldsymbol{\pi}_{\tilde{\boldsymbol{\theta}}_t}, \hat{\boldsymbol{f}}}\big[e^{-\frac{1}{\lambda}C(\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t)}\big]}, \tag{25}$$
  which reduces to the MPPI update rule [31] for $\gamma_t = 1$. This connection is also noted in [24].

C. Extensions

In the previous sections, we discussed multiple instantiations of DMD-MPC, showing the flexibility of our framework. But they are by no means exhaustive. In Section B, we discuss variations of DMD-MPC, e.g., imposing constraints and different ways to approximate the expectation in (9).

IV. Related Work

Recent work on MPC has studied sampling-based approaches, which are flexible in that they do not require differentiability of a cost function. One such algorithm which can be used with general cost functions and dynamics is MPPI, which was proposed by Williams et al. [31] as a generalization of the control affine case [30]. The algorithm is derived by considering an optimal control distribution defined by the control problem. This optimal distribution is intractable to sample from, so the algorithm instead tries to bring a tractable distribution (in this case, Gaussian with fixed covariance) as close as possible in the sense of KL divergence. This ends up being the same as finding the mean of the optimal control distribution. The mean is then approximated as a weighted sum of sampled control trajectories, where the weight is determined by the exponentiated costs. Although this algorithm works well in practice (including a robust variant [33] achieving state-of-the-art performance in aggressive driving [10]), it is not clear that matching the mean of the distribution should guarantee good performance, such as in the case of a multimodal optimal distribution. By contrast, our update rule in (25) results from optimizing an exponential utility.

A closely related approach is the cross-entropy method (CEM) [6], which also assumes a Gaussian sampling distribution but minimizes the KL divergence between the Gaussian distribution and a uniform distribution over low cost samples. CEM has found applicability in reinforcement learning [19, 21, 26], motion planning [17, 18], and MPC [9, 32].

These sampling-based control algorithms can be considered special cases of general derivative-free optimization algorithms, such as covariance matrix adaptation evolutionary strategies (CMA-ES) [15] and natural evolutionary strategies (NES) [29]. CMA-ES samples points from a multivariate Gaussian, evaluates their fitness, and adapts the mean and covariance of the sampling distribution accordingly. On the other hand, NES optimizes the parameters of the sampling distribution to maximize some expected fitness through steepest ascent, where the direction is provided by the natural gradient. Akimoto et al. [2] showed that CMA-ES can also be interpreted as taking a natural gradient step on the parameters of the sampling distribution. As we showed in Section III-B, natural gradient descent is a special case of DMD-MPC framework. A similar observation that connects between MPPI and mirror descent was made by Okada and Taniguchi [24], but their derivation is limited to the KL divergence and Gaussian case.

V. Experiments

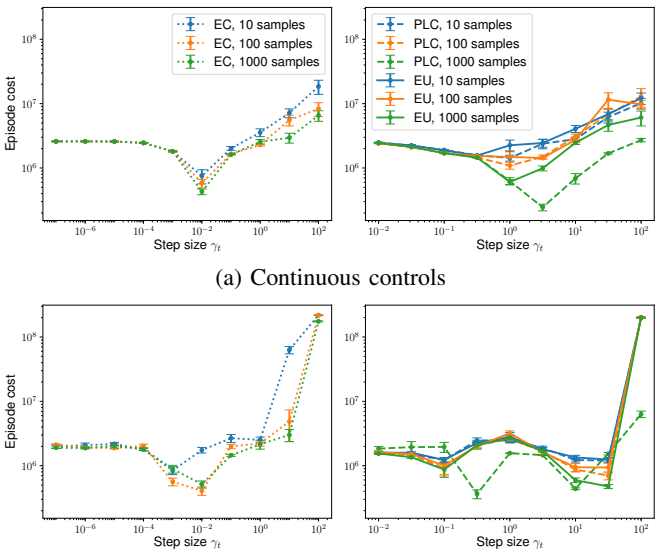
We use experiments to the validate the flexibility of DMD-MPC. We show that this framework can handle both continuous (Gaussian distribution) and discrete (categorial distribution) variations of control problems, and that MPC algorithms like MPPI and CEM can be generalized using different step sizes and control distributions to improve performance. Extra details and results are included in Sections E and F.

A. Cartpole

We first consider the classic cartpole problem where we seek to swing a pole upright and keep it balanced only using actuation on the attached cart. We consider both the continuous and discrete control variants. For the continuous case, we choose the Gaussian distribution as the control distribution and keep the covariance fixed. For the discrete case, we choose the categorical distribution and use update (20). In either case, we have access to a biased stochastic model (uses a different pole length compared to the real cart).

We consider the interaction between the choice of loss, step size, and number of samples used to estimate (9),[14] shown in Figs. 3 and 4. For this environment, we can achieve low cost when optimizing the expected cost in (10) with a proper step size (10^{-2} for both continuous and discrete problems) while being fairly robust to the number of samples. When using either of the utilities, the number of samples is more crucial in the continuous domain, with more samples allowing for larger step sizes. In the discrete domain (Fig. 3b), performance is largely unaffected by the number of samples when the step size is below 10, excluding the threshold utility with 1000 samples. In Fig. 4a, for a large range of utility parameters, we see that using step sizes above 1 (the step size set in MPPI and CEM) give significant performance gains. In Fig. 4b, there's a more complicated interaction between the utility parameter and step size, with huge changes in cost when altering the utility parameter and keeping the step size fixed.

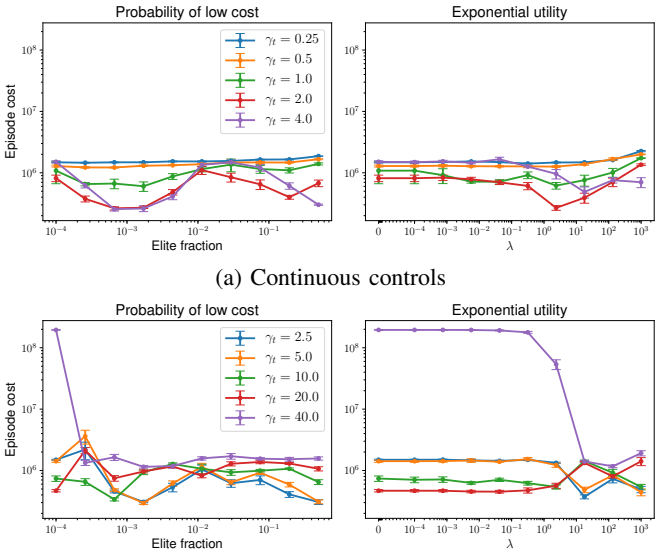
[14]For our experiments, we vary the number of samples from $\boldsymbol{\pi_\theta}$ and fix the number of samples from $\hat{\boldsymbol{f}}$ to ten. Furthermore, we use common random numbers when sampling from $\hat{\boldsymbol{f}}$ to reduce estimation variance.



(a) Continuous controls

(b) Discrete controls

Fig. 3: Varying step size and number of samples (same legends for (a) and (b)). EC = expected cost (10). PLC = probability of low cost (14) with elite fraction $= 10^{-3}$. EU = exponential utility (16) with $\lambda = 1$.



(a) Continuous controls

(b) Discrete controls

Fig. 4: Varying loss parameter and step size (1000 samples).

B. AutoRally

1) Platform Description: We use the autonomous AutoRally platform [13] to run a high-speed driving task on a dirt track, with the goal of the task to achieve as low a lap time as possible. The robot (Fig. 5) is a 1:5 scale RC chassis capable of driving over 20 m/s (45 mph) and has a desktop-class Intel Core i7 CPU and Nvidia GTX 1050 Ti GPU. For real-world experiments, we estimate the car's pose using a

Fig. 5: Rally car driving during an experiment.

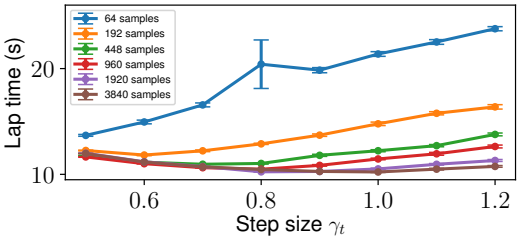


Fig. 6: Simulated AutoRally performance with different step sizes and number of samples. Though many samples coupled with large steps yield the smallest lap times, the performance gains are small past 1920 samples. With fewer samples, a lower step size helps recover some lost performance.

particle filter from [10] which relies on a monocular camera, IMU, and GPS. In both simulated and real-world experiments, the dynamics model is a neural network which has been fitted to data collected from human demonstrations. We note that the dynamics model is deterministic, so we don't need to estimate any expectations with respect to the dynamics.

2) Simulated Experiments: We first use the Gazebo simulator (Fig. 9 in Section E-B) to perform a sweep of algorithm parameters, particularly the step size and number of samples, to evaluate how changing these parameters can affect the performance of DMD-MPC. For all of the experiments, the control distribution is a Gaussian with fixed covariance, and we use update (25) (i.e., the loss is the exponential utility (16)) with $\lambda = 6.67$. The resulting lap times are shown in Fig. 6.[15] We see that although using more samples does result in smaller lap times, there are diminishing returns past 1920 samples per gradient. Indeed, with a proper step size, even as few as 192 samples can yield lap times within a couple seconds of 3840 samples and a step size of 1. We also observe that the curves converge as the step size decreases further, implying that only a certain number of samples are needed for a given step size. This is a particularly important advantage of DMD-MPC over methods like MPPI: by changing the step size, DMD-MPC can perform much more effectively with fewer samples, making it a good choice for embedded systems which can't produce many samples due to computational constraints.

3) Real-World Experiments: In the real-world setting (Fig. 7), the control distribution is a Gaussian with fixed covariance, and we use update (25) with $\lambda = 8$. We used the following experimental configurations: each of 1920 and 64 samples, and each of step sizes 1 (corresponding to MPPI), 0.8, and 0.6. Overall (Table I), there's a mild degradation in performance when decreasing the step size at 1920 samples, due to the car taking a longer path on the track (Fig. 12a vs. Fig. 12c in Section F-B). Using just 64 samples surprisingly only increases the lap times by 2 seconds and seems unaffected by the step size. This could be because, despite the noisiness of the DMD-MPC update, the setpoint controller in the car's steering servo acts as a filter, smoothing out the control signal and allowing the car to drive on a consistent path (Fig. 13 in Section F-B).

[15]The large error bar for 64 samples and step size of 0.8 is due to one particular lap where the car stalled at a turn for about 60 seconds.

Fig. 7: Real-world AutoRally task.

TABLE I: Avg. lap times (in sec.) for real-world experiments.

Step size γ_t	1920 samples	64 samples
1	31.76 ± 0.55	33.74 ± 0.78
0.8	31.81 ± 0.21	33.84 ± 0.80
0.6	32.83 ± 0.31	33.64 ± 0.74

VI. Conclusion

We presented a connection between model predictive control and online learning. From this connection, we proposed an algorithm based on dynamic mirror descent that can work for a wide variety of settings and cost functions. We also discussed the choice of loss function within this online learning framework and the sort of preference each loss function imposes. From this general algorithm and assortment of loss functions, we show several well known algorithms are special cases and presented a general update for members of the exponential family.

We empirically validated our algorithm on continuous and discrete simulated problems and on a real-world aggressive driving task. In the process, we also studied the parameter choices within the framework, finding, for example, that in our framework a smaller number of rollout samples can be compensated for by varying other parameters like the step size.

We hope that the online learning and stochastic optimization viewpoints of MPC presented in this paper opens up new possibilities for using tools from these domains, such as alternative efficient sampling techniques [5] and accelerated optimization methods [22, 24], to derive new MPC algorithms that perform well in practice.

Acknowledgements

This material is based upon work supported by NSF NRI award 1637758, NSF CAREER award 1750483, an NSF Graduate Research Fellowship under award No. 2015207631, and a National Defense Science & Engineering Graduate Fellowship. We thank Aravind Battaje and Hemanth Sarabu for assisting in AutoRally experiments.

References

[1] Pieter Abbeel, Adam Coates, and Andrew Y. Ng. Autonomous helicopter aerobatics through apprenticeship learning. *The International Journal of Robotics Research*, 29(13):1608–1639, 2010.

[2] Youhei Akimoto, Yuichi Nagata, Isao Ono, and Shigenobu Kobayashi. Theoretical foundation for CMA-ES from information geometry perspective. *Algorithmica*, 64(4):698–716, 2012.

[3] Arindam Banerjee, Srujana Merugu, Inderjit S. Dhillon, and Joydeep Ghosh. Clustering with Bregman divergences. *Journal of machine learning research*, 6(Oct): 1705–1749, 2005.

[4] Amir Beck and Marc Teboulle. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

[5] Richard Bellman and John Casti. Differential quadrature and long-term integration. *Journal of Mathematical Analysis and Applications*, 34(2):235–238, 1971.

[6] Zdravko I. Botev, Dirk P. Kroese, Reuven Y. Rubinstein, and Pierre L'Ecuyer. The cross-entropy method for optimization. In *Handbook of statistics*, volume 31, pages 35–59. Elsevier, 2013.

[7] Eduardo F. Camacho and Carlos Bordons Alba. *Model predictive control*. Springer Science & Business Media, 2013.

[8] Herman Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, 23(4):493–507, 1952.

[9] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, pages 4754–4765, 2018.

[10] Paul Drews, Grady Williams, Brian Goldfain, Evangelos A. Theodorou, and James M. Rehg. Vision-Based High-Speed Driving With a Deep Dynamic Observer. *IEEE Robotics and Automation Letters*, 4(2):1564–1571, 2019.

[11] Tyrone E. Duncan. Linear-exponential-quadratic Gaussian control. *IEEE Transactions on Automatic Control*, 58(11):2910–2911, 2013.

[12] Peter W. Glynn. Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, 33(10): 75–84, 1990.

[13] Brian Goldfain, Paul Drews, Changxi You, Matthew Barulic, Orlin Velev, Panagiotis Tsiotras, and James M. Rehg. AutoRally: An Open Platform for Aggressive Autonomous Driving. *IEEE Control Systems Magazine*, 39(1):26–55, 2019.

[14] Eric Hall and Rebecca Willett. Dynamical models and tracking regret in online convex programming. In *International Conference on Machine Learning*, pages 579–587, 2013.

[15] Nikolaus Hansen, Sibylle D. Müller, and Petros Koumoutsakos. Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES). *Evolutionary Computation*, 11(1):1–18, 2003.

[16] Elad Hazan. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016.

[17] Bjarne E. Helvik and Otto Wittner. Using the Cross-Entropy Method to Guide/Govern Mobile Agent's Path Finding in Networks. *International Workshop on Mobile Agents for Telecommunication Applications*, 2164, 2002.

[18] Marin Kobilarov. Cross-entropy randomized motion planning. In *Robotics: Science and Systems*, volume 7, pages 153–160, 2012.

[19] Shie Mannor, Reuven Y. Rubinstein, and Yohai Gat. The cross entropy method for fast policy search. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 512–519, 2003.

[20] David Q. Mayne. Model predictive control: Recent developments and future promise. *Automatica*, 50(12): 2967–2986, 2014.

[21] Ishai Menache, Shie Mannor, and Nahum Shimkin. Basis Function Adaptation in Temporal Difference Reinforcement Learning. *Annals of Operations Research*, 134: 215–238, 2005.

[22] Megumi Miyashita, Shiro Yano, and Toshiyuki Kondo. Mirror descent search and its acceleration. *Robotics and Autonomous Systems*, 106:107–116, 2018.

[23] Frank Nielsen and Vincent Garcia. Statistical exponential families: A digest with flash cards. *arXiv preprint arXiv:0911.4863*, 2009.

[24] Masashi Okada and Tadahiro Taniguchi. Acceleration of Gradient-based Path Integral Method for Efficient Optimal and Inverse Optimal Control. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3013–3020. IEEE, 2018.

[25] Magnus Rattray, David Saad, and Shun-ichi Amari. Natural gradient descent for on-line learning. *Physical review letters*, 81(24):5461, 1998.

[26] István Szita and András Lörincz. Learning Tetris using the noisy cross-entropy method. *Neural computation*, 18 (12):2936–2941, 2006.

[27] Yuval Tassa, Nicolas Mansard, and Emo Todorov. Control-limited differential dynamic programming. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1168–1175. IEEE, 2014.

[28] LJ van den Broek, WAJJ Wiegerinck, and HJ Kappen. Risk sensitive path integral control. In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence (UAI 2010)*, pages 1–8. AUAI Press, 2010.

[29] Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *The Journal of Machine Learning Research*, 15(1):949–980, 2014.

[30] Grady Williams, Paul Drews, Brian Goldfain, James M. Rehg, and Evangelos A. Theodorou. Aggressive driving with model predictive path integral control. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1433–1440. IEEE, 2016.

[31] Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M. Rehg, Byron Boots, and Evangelos A. Theodorou. Information Theoretic MPC for Model-Based Reinforcement Learning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1714–1721. IEEE, 2017.

[32] Grady Williams, Paul Drews, Brian Goldfain, James M. Rehg, and Evangelos A. Theodorou. Information-Theoretic Model Predictive Control: Theory and Applications to Autonomous Driving. *IEEE Transactions on Robotics*, 34(6):1603–1622, 2018.

[33] Grady Williams, Brian Goldfain, Paul Drews, Kamil Saigol, James M. Rehg, and Evangelos A. Theodorou. Robust sampling based model predictive control with sparse objective information. In *Robotics Science and Systems*, 2018.

[34] Lijun Zhang, Shiyin Lu, and Zhi-Hua Zhou. Adaptive Online Learning in Dynamic Environments. In *Advances in Neural Information Processing Systems*, pages 1323–1333, 2018.