Deep Value Model Predictive Control

*Farbod Farshidian[1], * David Hoeller[1,2], Marco Hutter[1]
[1]Robotic Systems Lab, ETH Zurich
[2]NVIDIA
{farbodf, dhoeller, mahutter}@ethz.ch

Abstract: In this paper, we introduce an actor-critic algorithm called Deep Value Model Predictive Control (DMPC), which combines model-based trajectory optimization with value function estimation. The DMPC actor is a Model Predictive Control (MPC) optimizer with an objective function defined in terms of a value function estimated by the critic. We show that our MPC actor is an importance sampler, which minimizes an upper bound of the cross-entropy to the state distribution of the optimal sampling policy. In our experiments with a Ballbot system, we show that our algorithm can work with sparse and binary reward signals to efficiently solve obstacle avoidance and target reaching tasks. Compared to previous work, we show that including the value function in the running cost of the trajectory optimizer speeds up the convergence. We also discuss the necessary strategies to robustify the algorithm in practice.

Keywords: Reinforcement Learning, Value Function Learning, Trajectory Optimization, Model Predictive Control

1 Introduction

Learning in environments with sparse reward/cost functions remains a challenging problem for Reinforcement Learning (RL). As the exploration strategy employed plays a vital role in such scenarios, the agent has to find and leverage small sets of informative samples maximally. Often, an agent can be provided with prior knowledge about the environment in the form of an incomplete model, such as the agent's dynamics. The sparsity of the reward and potential non-differentiability rule out the possibility of using Trajectory Optimization (TO). Furthermore, the sparsity of the cost function can be problematic even for RL methods that do not use structured and directed exploration policies, e.g., ε-greedy techniques. Thus, the goal of this work is to combine model-based and sample-based approaches in order to exploit the knowledge of the system dynamics while effectively exploring the environment.

Model Predictive Control (MPC) as a TO technique has proven to be a powerful tool in many robotic tasks [1, 2, 3]. While this model-based approach truncates the time horizon of the task, it continually shifts the shortened horizon forward and optimizes the state-input trajectory based on new state measurements. The main disadvantage of the MPC approach is its relatively high computational cost. As a consequence, the optimization time horizon is often kept short (e.g., in the order of seconds), which in turn prevents MPC from finding temporally global solutions. Furthermore, MPC heavily relies on the differentiability of the formulation and has a hard time dealing with sparse and non-continuous reward/cost signals.

On the other hand, RL solves the same problem by exploring and collecting information about the environment and making decisions based on samples. The RL agent has to learn about its environment, including its dynamics from scratch via trial and error. Nevertheless, deep reinforcement learning has displayed remarkable performance in long-horizon tasks with sparse rewards [4], even in the continuous domain [5, 6]. The main drawbacks of RL are that it still requires enormous amounts of data and suffers from the exploration-exploitation dilemma [7].

*Both authors contributed equally to this work (alphabetical ordering)

3rd Conference on Robot Learning (CoRL 2019), Osaka, Japan.

In this work, we derive an actor-critic approach where the critic is a value function learner and the actor an MPC strategy. Leveraging the generality of value functions, we propose to extend past work such as [8], resulting in an algorithm called Deep Value Model Predictive Control (DMPC). The DMPC algorithm uses an MPC policy to interact with the environment and collect informative samples to update its approximation of the value function. The running cost and the heuristic function (also known as the terminal cost) of MPC are defined in terms of the value function estimated by the critic.

We also provide an in-depth analysis of the bilateral effect of this actor-critic scheme. We show that the MPC actor is an importance sampler that minimizes an upper bound of the cross-entropy to the state trajectory distribution of the optimal sampling policy. Using the value function to define the MPC cost enables us to transform an initially stochastic task into a deterministic optimal control problem. We further empirically validate that defining a running cost in addition to the heuristic function accelerates the convergence of the value function, which makes the DMPC algorithm suitable for tasks with sparse reward function.

2 Problem Formulation

In this section, we provide more details on the control problem we solve. We consider the problem of sequential decision making in which an agent interacts with an environment to minimize the cumulative cost from the current time and onwards. We formulate this problem as a discounted Markov Decision Processes (MDP) [7] with a stochastic termination time.

The agent interacts with an environment with state $\mathbf{s}(t) \in \mathcal{S}$ and performs actions $\mathbf{u}_\pi(t) \in \mathcal{A}$ according to a time and state-dependent control policy $\boldsymbol{\pi}(t, \mathbf{s})$. For the sake of brevity, the dependency of $\mathbf{s}$ and $\mathbf{u}$ on t is dropped when clear from the context. We consider stochastic dynamical systems where the state evolves according to

$$d\mathbf{s} = \Big(\mathbf{f}(t, \mathbf{s}) + \mathbf{g}(t, \mathbf{s})\mathbf{u}_\pi\Big)dt + \mathbf{g}(t, \mathbf{s})d\mathcal{B}(t), \quad \mathbf{s}(t_0) = \mathbf{s}_0 \tag{1}$$

$$\mathbf{u}_\pi = \pi(t, \mathbf{s}) \tag{2}$$

where $\mathcal{B}(t)$ is a Brownian motion with $\mathbb{V}ar[d\mathcal{B}(t)] = \boldsymbol{\Sigma}dt$ and $\mathbf{s}_0$ is the initial state at time t_0.

The agent selects actions according to a policy $\boldsymbol{\pi}$ to minimize the discounted expected return for a set of initial states $\mathbf{s}_0 \in \mathcal{S}_0$. The discounted expected return is defined as

$$V^\pi(t_0, \mathbf{s}_0) = \mathbb{E}_\pi\big[C^\pi(t_0, \mathbf{s}_0)\big], \tag{3}$$

$$C^\pi(t_0, \mathbf{s}_0) = \gamma^{T-t_0} q_f(\mathbf{s}(T)) + \int_{t_0}^{T} \Big(\gamma^{t-t_0} q(t, \mathbf{s})dt + \frac{1}{2}\mathbf{u}_\pi^\top \mathbf{R}\, \mathbf{u}_\pi dt + \mathbf{u}_\pi^\top \mathbf{R} d\mathcal{B}\Big), \tag{4}$$

where $T \in [0, +\infty)$ is the termination time of the problem, $\gamma \in [0, 1)$ the discount factor, $q(t, \mathbf{s}) : \mathbb{R}^+ \times \mathcal{S} \to \mathbb{R}$ the running cost, $q_f(\mathbf{s}) : \mathcal{S} \to \mathbb{R}$ the termination cost, $\mathbf{R} = \mathbf{R}^\top$ a positive definite matrix regularizing the control inputs. $C^\pi(t_0, \mathbf{s}_0)$ is a stochastic variable called path cost. $V^\pi(t_0, \mathbf{s}_0)$ is computed by averaging over path costs generated from rollouts of the stochastic process in Equation (1) given a policy $\boldsymbol{\pi}$.

3 Preliminaries

Path Integral Optimal Control The optimal policy $\boldsymbol{\pi}^*(t, \mathbf{s})$ and the value function $V^*(t, \mathbf{s})$ it induces can be computed according to

$$\boldsymbol{\pi}^*(t, \mathbf{s}) = \arg\min_\pi V^\pi(t, \mathbf{s}), \tag{5}$$

$$V^*(t, \mathbf{s}) = \min_\pi V^\pi(t, \mathbf{s}). \tag{6}$$

The optimal value function, V^*, satisfies the stochastic Hamilton-Jacobi-Bellman (HJB) equation. The derivation of the HJB equation is based on the principles of dynamic programming. This formulation is quite general, but unfortunately, computing an analytical solution is only possible for some special cases such as LQ regulators. The original work of Kappen [9, 10] studies one of these

cases in which the non-linear HJB equation can be transformed into a linear equation by enforcing a constraint on the input regularization matrix and the covariance of the noise: $\mathbf{\Sigma R} = \lambda \mathbf{I}$ where $\lambda \in \mathbb{R}^+$. As a result of this linearity, the backward computation of the HJB equation can be replaced by a forward diffusion process that can be computed by stochastic integration. Therefore, the stochastic optimal control solution can be estimated with a Monte Carlo sampling method, resulting in the path integral control formulation

$$V^*(t, \mathbf{s}) = -\lambda \log \Psi^*(t, \mathbf{s}) \tag{7}$$

$$p^*(\rho) = \frac{1}{\Psi^*(t, \mathbf{s})} p^\pi(\rho) e^{-\frac{1}{\lambda} C^\pi(t, \mathbf{s})} \tag{8}$$

$$\Psi^*(t, \mathbf{s}) = \mathbb{E}_\pi[e^{-\frac{1}{\lambda} C^\pi(t, \mathbf{s})}], \tag{9}$$

where ρ is a state trajectory in the time interval $[t, T]$, $p^\pi(\rho)$ its corresponding probability distribution under policy $\boldsymbol{\pi}$, and $\Psi^*(t, \mathbf{s})$ is called the desirability function for the optimal policy.

Equations (7) and (8) give an explicit expression for the optimal value function and the optimal distribution of state trajectories in terms of the expectation of the cumulative cost over trajectories. The quality of this estimation depends on how well we can estimate the desirability function. In general, for problems with sparse cost functions, using the path integral approach is challenging and requires the use of more advanced approaches to extract samples.

Cross Entropy Efficient estimation of the solution to the path integral control problem critically depends on the quality of the collected samples. As shown by [11], the best sampling strategy for estimating the optimal solution is, ironically, the optimal policy itself*. Based on this observation, open-loop approaches such as [12, 13] use the latest estimate of the optimal policy to generate trajectory samples. However, they do not provide a systematic approach to control the variance of the sampling policy, which could lead to an inefficient sampling method, in particular, if the underlying system is unstable.

Using a policy that also controls the variance of the sampled state trajectories is required to have a state-dependent feedback [10]. However, finding such a feedback policy is equivalent to estimating the optimal sampling policy over the entire state space for each time. The challenge of this approach lies in the design and the update scheme of such a policy.

A common approach to tackle this issue is the cross-entropy method, which is an adaptive importance sampling scheme to estimate the probability of rare events. In this approach, the optimal sampler is estimated by a sequence of more tractable distributions, which are iteratively improved based on the collected samples. For that matter, the cross-entropy between the state probability distributions of the optimal (p^*) and the current sampler (p) is minimized, where the cross-entropy is defined as

$$\mathrm{H}(p^*, p) = \mathrm{H}(p^*) + \mathrm{KL}(p^* \| p).$$

It follows that minimizing the cross-entropy with respect to p, is equivalent to minimizing the KL divergence between p^* and p. This method has been proven to be useful for path-integral based problems. For example, Kappen and Ruiz [10] have employed a parameterized distribution to estimate the optimal sampler. Similarly, we here use a cross-entropy approach to estimate the optimal sampler. However, instead of a parameterized distribution family, we use an MPC strategy to estimate the optimal sampler implicitly.

Model Predictive Control The MPC strategy replaces the infinite-horizon optimization problem by a sequence of finite-horizon optimal control problems with prediction horizon H, which are numerically more tractable. At each control step, MPC solves the following optimal control problem from the current state and time. Then, only the first segment of the optimized sequence is used until the controller receives a new state and repeats the procedure.

$$\begin{aligned} \underset{\pi}{\text{minimize}} \quad & \gamma^H \phi(\mathbf{x}(t+H)) + \int_t^{t+H} \Big(\gamma^{\tau - t} l(\tau, \mathbf{x}) + \frac{1}{2} \mathbf{u}_\pi(\tau)^\top \mathbf{R}\, \mathbf{u}_\pi(\tau) \Big) d\tau \\ \text{subject to} \quad & d\mathbf{x}(\tau) = \Big(\mathbf{f}(\tau, \mathbf{x}) + \mathbf{g}(\tau, \mathbf{x}) \mathbf{u}_\pi(\tau) \Big) d\tau, \quad \mathbf{x}(t) = \mathbf{s} \\ & \mathbf{u}_\pi(\tau) = \pi(\tau, \mathbf{x}(\tau)). \end{aligned} \tag{10}$$

*Thijssen and Kappen [11] show that the variance of the desirability function estimation approaches to zero when the optimal policy is used as the sampling distribution ($\pi = \pi^*$)

Algorithm 1 Deep Value MPC

given An initial estimate $\hat{V}$ of V^*
repeat
  Sample an initial state $\mathbf{x}$
  $t := 0$
  while $t < T$ **do**
    Compute $\mathbf{u}_{\hat{\pi}(t)}$ according to Eq. (18)
    Execute $\mathbf{u}_{\hat{\pi}(t)}$, obtain a cost $c(t)$
    Add the transition tuple to a buffer
    $t = t + \delta t$
  end while
  Using the buffer, update $\hat{V}$ (Eq. (12))

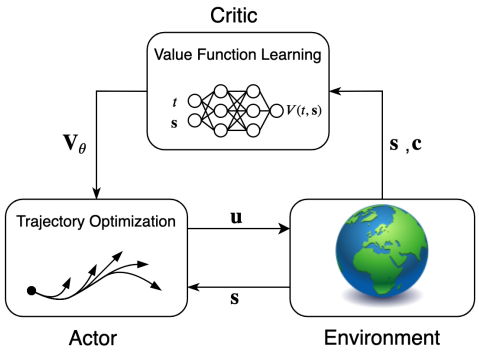


Figure 1: The DMPC pipeline. The actor is a trajectory optimizer that depends on the value learned by the critic.

Here, $l(\tau, \mathbf{x}) : \mathbb{R}^+ \times \mathcal{S} \to \mathbb{R}$ is the running cost, $\phi(\mathbf{x}) : \mathcal{S} \to \mathbb{R}$ the heuristic function which accounts for the truncated-time accumulated cost. We denote the state by $\mathbf{x}$ to clearly distinguish between the actor's computation and the state of the MDP (indicated by $\mathbf{s}$). Note that the system dynamics are deterministic, and therefore the optimal control problem is formulated deterministically.

4 Deep Value Model Predictive Control

Our DMPC algorithm is an actor-critic approach where a value function is used to provide a measure of how well an MPC actor performs. We assume that a model of the robot dynamics is available to an agent, and this internal nominal model is accurate. In the following, we briefly discuss the structure of the critic and the actor.

DMPC Critic The goal of the critic is to asses the performance of the actor by means of the value function. When the actor interacts with the environment (i.e. rolls out the policy), it collects transition tuples [7]. Using these samples, the critic is able to compute the empirical value along each trajectory and thus refine its estimate of the actual value function. The value function is represented by a function approximator parametrized by $\boldsymbol{\theta}$. Computing the one step Bellman target for a sample taken at time t, i.e.,

$$y(t) = c(t) + \gamma \hat{V}(t + \delta t, \mathbf{s}(t + \delta t)|\boldsymbol{\theta})), \tag{11}$$

where δt is the timestep and $c(t)$ a cost reflecting the performance on a given task, the critic can refine the value function estimate $\hat{V}$ by solving

$$\underset{\boldsymbol{\theta}}{\text{minimize}}\, \mathbb{E}\left[\left(y(t) - \hat{V}(t, \mathbf{s}(t)|\boldsymbol{\theta})\right)^2\right]. \tag{12}$$

Similar to [5], while the target y depends on $\boldsymbol{\theta}$, we neglect this dependency during the optimization.

DMPC Actor The DMPC actor is a trajectory optimizer as defined in equation (10), where the heuristic function and the running cost are defined as

$$\phi(\mathbf{x}(t + H)) = \hat{V}(t + H, \mathbf{x}(t + H)), \tag{13}$$

$$l(\tau, \mathbf{x}) = -\partial_t \hat{V}(\tau, \mathbf{x}) - \mathbf{f}(\tau, \mathbf{x})^\top \partial_{\mathbf{x}} \hat{V}(\tau, \mathbf{x}) + \frac{1}{2} \partial_{\mathbf{x}} \hat{V}(\tau, \mathbf{x})^\top \boldsymbol{\Xi}(\tau, \mathbf{x})\, \partial_{\mathbf{x}} \hat{V}(\tau, \mathbf{x}), \tag{14}$$

with $\boldsymbol{\Xi}(\tau, \mathbf{x}) = \mathbf{g}(\tau, \mathbf{x}) \mathbf{R}^{-1}\, \mathbf{g}(\tau, \mathbf{x})^\top$.

This MPC formulation replaces the original T-horizon problem by a sequence of finite-horizon optimal control sub-problems with prediction horizon H ($H < T$). The dynamic programming principle ensures that if the sub-problems are formulated with the exact value function as a heuristic function, then the MPC method solves the problem globally. However, in practice, the actor only has an approximation of the optimal value function. While the approximation error degrades the

performance of greedy policies, the MPC benefits from the look-ahead mechanism. It can be shown that if the horizon of the actor is long enough, the effect of the value function error is mitigated [8]. Therefore, MPC strategies using an approximate value function for the heuristic function, in general, outperform methods that are based on the instantaneous minimization of the value function estimate.

Another significant advantage of using such an approach is that we can compute a temporally global optimal sequence of actions using the temporally local solution of the MPC. This further allows us to tune the MPC time horizon based on the available computational resource and use the value function to account for the truncated accumulated cost.

Actor-Critic Interaction The procedure for the DMPC algorithm directly follows: From an estimate $\hat{V}$ of V^* given by the critic, the MPC actor computes the policy $\boldsymbol{\pi}$ by solving Equation (10). The critic improves the estimate $\hat{V}$ based on the collected samples using Equation (12). This results in an off-policy actor-critic method since instead of assessing the value of the current policy, the actor directly estimates the optimal value function. The main steps of DMPC are outlined in Algorithm 1.

The MPC policy is an importance sampler for the value function learner. To motivate this, consider a problem with a sparse cost. Since the MPC actor predicts the state evolution, it can foresee less costly areas of the state space and propagate this information back to the current time. As a result, it can coordinate the action sequence to steer the agent towards future rewarding regions. In the next section, we provide more formal insights and show that MPC minimizes an upper bound of the cross-entropy between the state trajectory distribution of the optimal policy and the current policy.

By repeating the actor-critic interactions, the estimation of the value function gets closer to the optimal one, which, in turn, improves the MPC performance by enhancing the estimate of the truncated cumulative cost.

5 DMPC Properties

In this section, we analyze the properties of the DMPC algorithm. We start with a setup where the MPC actor uses the learned value function only for the heuristic function. There, we study the effect of this actor-critic setup on the convergence of the value function learning. Next, we propose a setting where a running cost is defined based on the value function. We discuss that such a cost extends the application of the actor-critic method to problems with a sparse and non-differentiable cost. Moreover, it allows us to use a deterministic MPC solver instead of a stochastic one.

5.1 Role of the Heuristic Function

As a first step, we focus on the case where we only use the learned value function as the heuristic function of the MPC actor. This setup is similar to the approach proposed by [8]. We here build upon their result and provide a more in-depth analysis of the impact of the MPC actor on the acceleration of the value function convergence.

As discussed, the convergence of the value function critically depends on the sampling distribution and, the optimal sampler for the stochastic optimal control problem defined in (1)-(4) is the optimal control policy π^* [11]. However, we cannot compute the optimal distribution during the learning process, and instead, we wish to find the near-optimal control policy π such that p^π is close to p^*. As discussed in the introduction of the cross-entropy method, we wish to minimize

$$\mathrm{KL}(p^* \| p^\pi) = -\mathbb{E}_{p^*}\left[\log\left(\frac{p^\pi}{p^*}\right)\right]. \tag{15}$$

Theorem 1. *Assuming that from a state $\mathbf{s}(t)$ the policy remains in the vicinity of the optimal policy up to time T, i.e., $\int_t^T \frac{1}{2}\|\mathbf{u}_*(\tau) - \mathbf{u}_\pi(\tau)\|^2_{\mathbf{R}} d\tau \leq \mathcal{E}$, an upper bound for the forward KL divergence between the state trajectory distributions of the optimal policy and policy π is given by*

$$\mathrm{KL}(p^* \| p^\pi) \leq \mathrm{KL}(p^\pi \| p^*) + \Big(\mathcal{E}\ \mathrm{KL}(p^\pi \| p^*)\, \mathbb{V}ar\Big[\frac{e^{-\frac{1}{\lambda}C^\pi(t,\mathbf{s})}}{\Psi(t,\mathbf{s})}\Big]\Big)^{\frac{1}{2}}. \tag{16}$$

Proof. The proof is provided in Appendix A.2. □

This shows that $\mathrm{KL}(p^*\|p^\pi)$ has an upper bound defined by the reverse KL divergence $\mathrm{KL}(p^\pi\|p^*)$ and the variance of the normalized desirability function using the policy π for sampling. Next, we show that the MPC actor minimizes this reverse KL divergence and that its performance is bounded by the approximation error of the value function. Moreover, we show that the quality of the estimation improves as the MPC horizon H increases.

Theorem 2. *Suppose that $\hat{V}$ is an approximation of the optimal value function with infinity norm $\mathcal{L} := \max_{t,s} |\hat{V}(t,\mathbf{s}) - V^*(t,\mathbf{s})|$. The policy $\boldsymbol{\pi}$ is the solution to the problem (10) with terminal cost $\phi(\mathbf{x}_f) = \hat{V}(t_f, \mathbf{x}_f)$. Then for all MPDs, the reverse KL divergence $\mathrm{KL}(p^\pi\|p^*)$ can be bounded as*

$$\mathrm{KL}(p^\pi\|p^*) \leq \frac{2\mathcal{L}\gamma^H}{\lambda(1-\gamma^H)} \tag{17}$$

Proof. The proof is provided in Appendix A.3. □

Note that if $H_1 < H_2$ then $\frac{\gamma^{H_2}}{(1-\gamma^{H_2})} < \frac{\gamma^{H_1}}{(1-\gamma^{H_1})}$. Therefore, as the horizon increases the MPC is less susceptible to the value function approximation error.

5.2 Role of the Running Cost

In this section, we motivate the choice of the DMPC running cost. However, we do not provide any formal proof. The goal is to see how we can extend the idea of using the value function in the heuristic function to the running cost. We show that, at least for the case where we have the optimal value function, this is indeed possible. Thus, for the following analysis, we assume that the exact optimal value function is provided. However, later in the result section, we will empirically show that for the scenarios with sparse costs, the running cost based on the approximate value function also accelerates the convergence of the critic.

Proposition 1. *The control policy $\pi(t,\mathbf{s})$ that minimizes the reverse KL divergence $\mathrm{KL}(p^\pi\|p^*)$ of the problem defined in Equations (1)-(4) is also the solution to the deterministic problem*

$$\boldsymbol{\pi}(t,\mathbf{s}) = \arg\min_{\pi} \left\{ \phi_d(\mathbf{x}(t+H)) + \int_t^{t+H} \Big(l_d(\tau,\mathbf{x}) + \frac{1}{2}\mathbf{u}_\pi^\top(\tau)\mathbf{R}\,\mathbf{u}_\pi(\tau) \Big) d\tau \right\} \tag{18}$$

where $l_d(\tau,\mathbf{x})$, the running cost, and $\phi_d(\mathbf{x}(t+H))$, the heuristic function are defined as

$$l_d(\tau,\mathbf{x}) = -\partial_t V^*(\tau,\mathbf{x}) - \mathbf{f}(\tau,\mathbf{x})^\top \partial_{\mathbf{x}} V^*(\tau,\mathbf{x}) + \frac{1}{2}\partial_{\mathbf{x}} V^*(\tau,\mathbf{x})^\top \boldsymbol{\Xi}(\tau,\mathbf{x})\, \partial_{\mathbf{x}} V^*(\tau,\mathbf{x}), \tag{19}$$

$$\phi_d(\mathbf{x}(t+H)) = V^*(t+H, \mathbf{x}(t+H)), \tag{20}$$

with $V^*(\tau,\mathbf{x})$ is the optimal value function of the stochastic problem defined in (6) and the state trajectory evolves based on the following deterministic dynamics

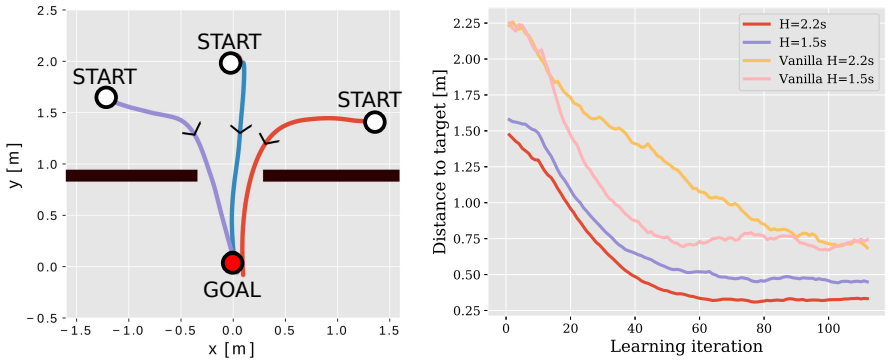
$$\dot{\mathbf{x}} = \mathbf{f}(\tau,\mathbf{x}) + \mathbf{g}(\tau,\mathbf{x})\mathbf{u}_\pi, \quad \mathbf{x}(t) = \mathbf{s} \tag{21}$$

Proof. The proof is provided in Appendix A.4. □

Proposition 1 has an important implication; the primary stochastic optimization problem is transformed into a deterministic one. Therefore, a deterministic MPC solver such as the one defined in Equation (10) can be used. This ultimately means that in order to find the optimal sampling policy, we only need to solve a deterministic MPC problem. This further allows us to employ more sophisticated tools from deterministic optimal control, e.g., Differential Dynamic Programming (DDP) [14].

6 Results

In this section, we describe the implementation details of the DMPC pipeline. We describe how the different components are designed and highlight the techniques used to make the interaction between the actor and the critic more robust. We then present the experiment results.



(a) (b)

Figure 2: (a) Top down view of the trajectories of the Ballbot's center of mass for different starting positions. (b) Performance on the target reaching task for different time horizons and cost functions.

Critic Implementation During the rollout of the policy, the critic stores the transition tuples in a replay buffer. Regularly, it samples K mini-batches of size N to update the value function. The value function is represented by a multilayer perceptron (in our experiments, 3 layers with 12 units each and tanh activations). Following an approach similar to [15] and [16], we condition the value function on a goal and a time to reach the goal. It can be interpreted as the value of being in a particular state if a specific target has to be reached within a given amount of time. Since the derivatives of the network are computed on the actor side, particular attention has to be given to the architecture and the training procedure. The choice of a differentiable activation function, such as tanh, is necessary to guarantee the differentiability of the whole network. Moreover, we noticed that decaying the weights results in smoother behavior. Finally, the critic uses a target network [5] so that the actor only receives a Polyak averaged version of the value function.

Actor Implementation The MPC actor uses a DDP-based algorithm known as SLQ [17, 2]. The Jacobian and the Hessian of the value function network, which are necessary to compute the derivatives of the cost function, are computed using an automatic differentiation library.

6.1 Experimental Results

With the following experiments in simulation, we would like to confirm the theory and intuition derived above. More specifically, we would like to show that the algorithm is capable of handling a sparse binary reward set-up, that using the running cost (19) yields better convergence, and highlight the importance of the MPC time horizon during learning.

Our experiments are based on the Ballbot robot (see Appendix B for more details). In order to answer the questions above, we design a task where the agent has to reach a target in 3 seconds from any initial position. Additionally, we add walls to the environment so that most of the time, it cannot reach the target via the shortest path, see Figure 2a. While the target reaching cost is simply encoded as the euclidean distance from the current position to the goal, the walls are encoded by a termination of the episode with a fixed penalty. First, we analyze the system for different MPC time horizons. Then, we assess the performance when the running cost (19) is left out (i.e., we only use the value function as the heuristic function), which corresponds to the vanilla case [8]. Trajectories of the Ballbot's center of mass for different starting positions are shown in Figure 2a.

Influence of the Running Cost As shown in Figure 2b, the running cost plays a vital role in solving the task. While the actors devoid of the running cost solve the reaching task well (it is a smooth and straightforward cost after all), they often collide with the walls. When the horizon is too long for them, the optimization is prone to overlooking the wall. This is because the value is only used at the end of the trajectory. Also, the exploration provided by the value function is not sufficient enough, and the Ballbot often collides with the wall. When using the running cost, the task is solved successfully upon convergence. The impact of that term is shown in Figure 3. The information encoded in the value function is able to produce regions of low cost and guide the solution to avoid

*Supplementary video: https://youtu.be/9p4qHBUZDMA

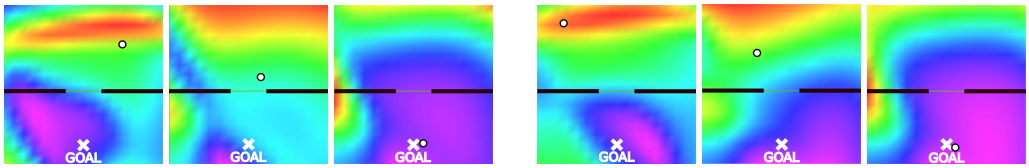


Figure 3: Evolution of the running cost (19) along two trajectories (first three and last three images). The position of the Ballbot is denoted by the white dot and the target by the white cross. The heat-map represents the magnitude of the running cost at a given time for different x and y coordinates. Red corresponds to high cost and violet to low cost regions.

the obstacles along the path. When the robot starts on the left side of the field, a region of low cost on the bottom right side is created, encouraging the robot to move forward and avoid the obstacle. The opposite happens when the robot starts on the right. Moreover, at the beginning of the trajectories, regions of high cost at the top of the field encourage the robot to move towards the goal.

Influence of the Horizon on the Convergence of the Value Function As described in the previous section, the time horizon of the MPC plays a vital role. Indeed, with a longer time horizon, the actor can look further into the future during the trajectory optimization step. It accelerates the convergence of the value function and improves exploration. As can be seen in Figure 2b, during training, the actor with the longer time horizon outperforms the one with a shorter one. The actor with a longer time horizon is able to see beyond the wall faster in order to reach the target. The actor with the shorter horizon tends to focus more on the wall and needs more time to get past it. On the other hand, increasing the time horizon results in a higher computational cost for the trajectory optimizer so that a trade-off has to be made.

7 Related Work

The use of learning in conjunction with planning has been studied extensively in the past. Prominently, in the discrete domain, learning evaluation functions has been employed with tree search methods resulting in systems capable of planning over long time horizons [4, 18]. In inverse reinforcement learning/optimal control [19, 20, 21], the planner is taught to match the behavior of an expert by inferring a cost function. These methods, however, are bounded by the performance of the expert, which is assumed to showcase optimal behavior. In our problem setting, the cost function is inferred indirectly via the value function that is learned from the environment-issued rewards/costs. The performance is only bounded by the capacity to learn the optimal value function.

The use of trajectory optimization with value function learning has been studied most recently in the Plan Online Learn Offline framework [8]. By using the value function as the terminal cost of their trajectory optimizer, they show improved performance of the policy beyond the optimizer's time horizon. Here, we further extend their idea to handle stochastic systems explicitly and formulate the optimizer's running cost such that it results in an importance sampler of the optimal value function.

When combining planning and learning, exploration plays a key role because it is difficult to cover the task-relevant state space in high-dimensional problems efficiently. To this end, methods such as path integral optimal control employ importance sample schemes [10]. In RL, methods such as Guided Policy Search [22] use a planner to direct the policy learning stage and sample more efficiently from high reward regions.

8 Conclusion and Future Work

In this paper, we presented an off-policy actor-critic algorithm called DMPC that extends previous work on the combination of trajectory optimization and global value function learning. We first show that using a value function in the heuristic function leads to a temporally global optimal solution. Next, we show that using the running cost (19) results not only in an importance sampling scheme that improves the convergence of the value function estimation but is also capable of taking system uncertainties into account. In future work, we like to extend the value function to encode more information. For example, we could condition it on a local robot-centric map that would allow it to make decisions in dynamic environments to avoid obstacles.

Acknowledgments

This work was supported by NVIDIA, the Swiss National Science Foundation through the National Centre of Competence in Research Robotics (NCCR Robotics), the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme under grant agreement No. 852044.

References

[1] K. Alexis, C. Papachristos, G. Nikolakopoulos, and A. Tzes. Model predictive quadrotor indoor position control. In *2011 19th Mediterranean Conference on Control Automation (MED)*, pages 1247–1252, June 2011. doi:10.1109/MED.2011.5983144.

[2] F. Farshidian, E. Jelavic, A. Satapathy, M. Giftthaler, and J. Buchli. Real-time motion planning of legged robots: A model predictive control approach. In *Humanoids*, pages 577–584, 2017. doi:10.1109/HUMANOIDS.2017.8246930.

[3] J. Koenemann, A. D. Prete, Y. Tassa, E. Todorov, O. Stasse, M. Bennewitz, and N. Mansard. Whole-body model-predictive control applied to the hrp-2 humanoid. In *IROS*, pages 3346–3351, 2015.

[4] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis. Mastering the game of go without human knowledge. *Nature*, 550:354–, Oct. 2017. URL http://dx.doi.org/10.1038/nature24270.

[5] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. In *4th International Conference on Learning Representations, ICLR 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1509.02971.

[6] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26), 2019. doi:10.1126/scirobotics.aau5872. URL https://robotics.sciencemag.org/content/4/26/eaau5872.

[7] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

[8] K. Lowrey, A. Rajeswaran, S. Kakade, E. Todorov, and I. Mordatch. Plan online, learn offline: Efficient learning and exploration via model-based control. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Byey7n05FQ.

[9] H. J. Kappen. An introduction to stochastic control theory, path integrals and reinforcement learning. In *AIP conference proceedings*, volume 887, pages 149–181. AIP, 2007.

[10] H. J. Kappen and H. C. Ruiz. Adaptive importance sampling for control and inference. *Journal of Statistical Physics*, 162(5):1244–1266, Mar 2016. ISSN 1572-9613. doi:10.1007/s10955-016-1446-7.

[11] S. Thijssen and H. Kappen. Path integral control and state-dependent feedback. *Physical Review E*, 91(3):032104, 2015.

[12] E. Theodorou, J. Buchli, and S. Schaal. A generalized path integral control approach to reinforcement learning. *journal of machine learning research*, 11(Nov):3137–3181, 2010.

[13] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. 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.

[14] D. Mayne. A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems. *International Journal of Control*, 3(1):85–95, 1966.

[15] V. Pong*, S. Gu*, M. Dalal, and S. Levine. Temporal difference models: Model-free deep rl for model-based control. In *6th International Conference on Learning Representations (ICLR)*, May 2018. URL https://openreview.net/forum?id=Skw0n-W0Z&noteId=Skw0n-W0Z. *equal contribution.

[16] T. Schaul, D. Horgan, K. Gregor, and D. Silver. Universal value function approximators. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 1312–1320. JMLR.org, 2015. URL http://dl.acm.org/citation.cfm?id=3045118.3045258.

[17] F. Farshidian, M. Neunert, A. W. Winkler, G. Rey, and J. Buchli. An efficient optimal planning and control framework for quadrupedal locomotion. In *ICRA*, pages 93–100. IEEE, 2017.

[18] A. Guez, T. Weber, I. Antonoglou, K. Simonyan, O. Vinyals, D. Wierstra, R. Munos, and D. Silver. Learning to search with MCTSnets, 2018. URL http://arxiv.org/pdf/1802.04697v2.

[19] S. Ross, G. J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. *Journal of Machine Learning Research - Proceedings Track*, 15, 11 2010.

[20] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 1–, New York, NY, USA, 2004. ACM. ISBN 1-58113-838-5. doi:10.1145/1015330.1015430. URL http://doi.acm.org/10.1145/1015330.1015430.

[21] K. Dvijotham and E. Todorov. Inverse optimal control with linearly-solvable mdps. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 335–342, USA, 2010. Omnipress. ISBN 978-1-60558-907-7. URL http://dl.acm.org/citation.cfm?id=3104322.3104366.

[22] S. Levine and V. Koltun. Guided policy search. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pages III–1–III–9. JMLR.org, 2013. URL http://dl.acm.org/citation.cfm?id=3042817.3042937.

[23] M. V. Minniti, F. Farshidian, R. Grandia, and M. Hutter. Whole-body mpc for a dynamically stable mobile manipulator. *ArXiv*, abs/1902.10415, 2019.

A Proof of Theorems

A.1 Girsanov Theorem

For the stochastic process defined in Equation (1), it is possible to relate the state trajectory distribution p of two policies π_1 and π_2 via the Girsanov theorem. Here, we briefly outline the steps of an informal derivation by discretizing the state trajectory with infinitesimal time dt. The distribution of $\mathbf{x}(t+dt)$ conditioned on $\mathbf{x}(t)$ is a Gaussian distribution with mean $\boldsymbol{\mu}_\pi(t) = \mathbf{x}(t) + \mathbf{f}(t,\mathbf{x}(t))dt + \mathbf{g}(t,\mathbf{x}(t))\mathbf{u}_\pi(t)dt$ and variance $\boldsymbol{\Xi}(t) = \mathbf{g}(t,\mathbf{x})\mathbf{R}^{-1}\mathbf{g}(t,\mathbf{x})^\top$. As a result, for the state trajectory distribution under policy π_1 we have

$$
\begin{aligned}
p^{\pi_1}(\rho) &= \lim_{dt\to 0} \prod_{s=0}^{T-dt} \mathcal{N}(\boldsymbol{\mu}_{\pi_1}(s), \boldsymbol{\Xi}(s)dt) \\
&\propto \lim_{dt\to 0} \prod_{s=0}^{T-dt} \exp\left(-\frac{1}{2}\left\|\mathbf{x}(s+dt) - \boldsymbol{\mu}_{\pi_1}(s)\right\|^2_{\boldsymbol{\Xi}^{-1}} dt^{-1}\right) \\
&\propto \lim_{dt\to 0} \prod_{s=0}^{T-dt} \exp\left(-\frac{1}{2}\left\|(\mathbf{x}(s+dt) - \boldsymbol{\mu}_{\pi_2}(s)) - (\boldsymbol{\mu}_{\pi_1}(s) - \boldsymbol{\mu}_{\pi_2}(s))\right\|^2_{\boldsymbol{\Xi}^{-1}} dt^{-1}\right).
\end{aligned}
$$

After further simplifications,

$$
\begin{aligned}
p^{\pi_1}(\rho) = \lim_{dt\to 0} \prod_{s=0}^{T-dt} &\mathcal{N}(\boldsymbol{\mu}_{\pi_2}(s), \boldsymbol{\Xi}(s)dt) \exp\left(-\frac{1}{2}\left\|\boldsymbol{\mu}_{\pi_1}(s) - \boldsymbol{\mu}_{\pi_2}(s)\right\|^2_{\boldsymbol{\Xi}^{-1}} dt^{-1}\right) \\
&\exp\left((\mathbf{x}(s+dt) - \boldsymbol{\mu}_{\pi_2}(s))^\top \boldsymbol{\Xi}(s)^{-1}(\boldsymbol{\mu}_{\pi_1}(s) - \boldsymbol{\mu}_{\pi_2}(s))dt^{-1}\right) \\
= p^{\pi_2}(\tau) &\exp\Big(-\frac{1}{2}\int_0^T \|\mathbf{u}_1(t) - \mathbf{u}_2(t)\|^2_{\mathbf{R}}\, dt\Big) \exp\Big(\int_0^T (\mathbf{u}_1(t) - \mathbf{u}_2(t))^\top \mathbf{R}\, d\mathcal{B}(t))\Big).
\end{aligned}
$$

Thus we have

$$
\frac{p^{\pi_1}(\rho)}{p^{\pi_2}(\tau)} = \exp\Big(-\frac{1}{2}\int_0^T \|\mathbf{u}_1(t) - \mathbf{u}_2(t)\|^2_{\mathbf{R}}\, dt\Big) \exp\Big(\int_0^T (\mathbf{u}_1(t) - \mathbf{u}_2(t))^\top \mathbf{R}\, d\mathcal{B}(t))\Big) \tag{22}
$$

Using the Girsanov theorem, it can be readily shown that the KL divergence between p^{π_1} and p^{π_2} takes the following form

$$
\mathrm{KL}(p^{\pi_2}\|p^{\pi_1}) = -\,\mathbb{E}_{p^{\pi_2}}\left[\log\left(\frac{p^{\pi_1}(\rho)}{p^{\pi_2}(\rho)}\right)\right] = \mathbb{E}_{p^{\pi_2}}\left[\int_0^T \frac{1}{2}\|\mathbf{u}_1(t) - \mathbf{u}_2(t)\|^2_{\mathbf{R}}\, dt\right] \tag{23}
$$

A.2 Proof of Theorem 1

Proof. Based on the Girsanov theorem and Equation (23), the KL-divergence between the optimal and the current state trajectory distribution starting from an initial time t and an initial state $\mathbf{x}(t) = \mathbf{s}$, can be written as

$$
\mathrm{KL}(p^*\|p^\pi) = \mathbb{E}_{p^*}\left[\int_t^T \frac{1}{2}\|\mathbf{u}^*(\tau) - \mathbf{u}_\pi(\tau)\|^2_{\mathbf{R}}\, d\tau\right] = \mathbb{E}_{p^\pi}\left[\frac{p^*}{p^\pi}\int_t^T \frac{1}{2}\|\mathbf{u}^*(\tau) - \mathbf{u}_\pi(\tau)\|^2_{\mathbf{R}} d\tau\right]
$$

Equation (8) defines the relationship in between the state trajectory probability distribution of the optimal policy, π^*, and the sampling policy, π as

$$
\begin{aligned}
\mathrm{KL}(p^*\|p^\pi) &= \mathbb{E}_{p^\pi}\left[\frac{e^{-\frac{1}{\lambda}C^\pi(t,\mathbf{s})}}{\Psi^*(t,\mathbf{s})}\int_t^T \frac{1}{2}\|\mathbf{u}^*(\tau) - \mathbf{u}_\pi(\tau)\|^2_{\mathbf{R}} d\tau\right] \\
&= \mathrm{KL}(p^\pi\|p^*) + \mathbb{E}_{p^\pi}\left[\frac{e^{-\frac{1}{\lambda}C^\pi(t,\mathbf{s})} - \Psi^*(t,\mathbf{s})}{\Psi^*(t,\mathbf{s})}\int_t^T \frac{1}{2}\|\mathbf{u}^*(\tau) - \mathbf{u}_\pi(\tau)\|^2_{\mathbf{R}} d\tau\right] \\
&= \mathrm{KL}(p^\pi\|p^*) + \Delta
\end{aligned}
$$

where we have replaced the second term by Δ. The second line naturally follows by using Equation (23) for $\mathrm{KL}(p^\pi \| p^*)$. Now we further examine the term Δ.

$$\begin{aligned}
\Delta^2 \leq & \mathbb{E}_{p^\pi}\left[\Big(\frac{e^{-\frac{1}{\lambda}C^\pi(t,\mathbf{s})} - \Psi^*(t,\mathbf{s})}{\Psi^*(t,\mathbf{s})}\Big)^2\right] \mathbb{E}_{p^\pi}\left[\Big(\int_t^T \frac{1}{2}\|\mathbf{u}^*(\tau) - \mathbf{u}_\pi(\tau)\|^2_{\mathbf{R}} d\tau\Big)^2\right] \\
= & \mathbb{V}ar\Big[\frac{e^{-\frac{1}{\lambda}C^\pi(t,\mathbf{s})}}{\Psi^*(t,\mathbf{s})}\Big] \mathbb{E}_{p^\pi}\left[\Big(\int_t^T \frac{1}{2}\|\mathbf{u}^*(\tau) - \mathbf{u}_\pi(\tau)\|^2_{\mathbf{R}} d\tau\Big)^2\right] \\
\leq & \mathbb{V}ar\Big[\frac{e^{-\frac{1}{\lambda}C^\pi(t,\mathbf{s})}}{\Psi^*(t,\mathbf{s})}\Big] \left(\mathcal{E}\, \mathbb{E}_{p^\pi}\left[\int_t^T \frac{1}{2}\|\mathbf{u}^*(\tau) - \mathbf{u}_\pi(\tau)\|^2_{\mathbf{R}} d\tau\right]\right)
\end{aligned}$$

In the first line, we used the Cauchy-Schwarz inequality. Then, in the second line we have used the definition of the desirability function in Equation (9). Finally, in the third line, we used the Bhatia-Davis inequality which provides an upper bound on the variance of a bounded probability distribution. For the above we have

$$0 \leq \int_t^T \frac{1}{2}\|\mathbf{u}^*(\tau) - \mathbf{u}^\pi(\tau)\|^2_{\mathbf{R}} d\tau \leq \mathcal{E}$$

Thus, based on the Bhatia-Davis inequality we can write

$$\begin{aligned}
\mathbb{E}_{p^\pi}\left[\Big(\int_t^T \frac{1}{2}\|\mathbf{u}^*(\tau) - \mathbf{u}_\pi(\tau)\|^2_{\mathbf{R}} d\tau\Big)^2\right] - \Big(\mathrm{KL}(p^\pi\|p^*)\Big)^2 \leq & \Big(\mathcal{E} - \mathrm{KL}(p^\pi\|p^*)\Big)\, \mathrm{KL}(p^\pi\|p^*) \\
\mathbb{E}_{p^\pi}\left[\Big(\int_t^T \frac{1}{2}\|\mathbf{u}^*(\tau) - \mathbf{u}_\pi(\tau)\|^2_{\mathbf{R}} d\tau\Big)^2\right] \leq & \mathcal{E}\, \mathrm{KL}(p^\pi\|p^*)
\end{aligned}$$

Thus, we will have the following upper bound on Δ

$$\Delta \leq |\Delta| \leq \Big(\mathcal{E}\ \mathrm{KL}(p^\pi\|p^*)\, \mathbb{V}ar\Big[\frac{e^{-\frac{1}{\lambda}C^\pi(t,\mathbf{s})}}{\Psi^*(t,\mathbf{s})}\Big]\Big)^{\frac{1}{2}}$$

□

A.3 Proof of Theorem 2

Proof. By using the relationship between the state trajectory probability distribution of the optimal policy, π^*, and the sampling policy, π, defined in Equation (8), we get

$$\begin{aligned}
\mathrm{KL}(p^\pi\|p^*) = & -\mathbb{E}_{p^\pi}\left[\log\left(\frac{p^*}{p^\pi}\right)\right] \\
= & -\mathbb{E}_{p^\pi}\left[\log\left(\frac{e^{-\frac{1}{\lambda}C^\pi(t,\mathbf{s})}}{\Psi^*(t,\mathbf{s})}\right)\right] \\
= & \frac{1}{\lambda}\mathbb{E}_{p^\pi}\left[C^\pi(t,\mathbf{s}) + \lambda \log \Psi^*(t,\mathbf{s})\right] \\
= & \frac{1}{\lambda}\mathbb{E}_{p^\pi}\left[C(t,\mathbf{s})\right] - \frac{1}{\lambda}\mathbb{E}_{p^*}\left[C(t,\mathbf{s})\right] \qquad (24)
\end{aligned}$$

where we used Equation (7) and then $\lambda \log \Psi^*(t,\mathbf{s}) = -V^*(t,\mathbf{s}) = -\mathbb{E}_{p^*}\left[C(t,\mathbf{s})\right]$.

The rest of this proof follows similar to the result presented in [8]. The difference is that our formulation is continuous-time while the formulation in [8] is discrete-time. For the sake of brevity, we will use $l(\tau,\mathbf{x},\mathbf{u}) := \gamma^{\tau-t} q(\tau,\mathbf{x}) + \frac{1}{2}\mathbf{u}^\top \mathbf{R}\, \mathbf{u}$ during this proof.

$$\begin{aligned}
\mathbb{E}_{p^*}\left[C(t,\mathbf{s})\right] &= \mathbb{E}_{p^*}\left[\gamma^{T-t} q_f(\mathbf{x}(T)) + \int_t^T l(\tau,\mathbf{x},\mathbf{u}) d\tau\right] \\
&= \mathbb{E}_{p^*}\left[\gamma^H V^*(t_f,\mathbf{x}_f) + \int_t^{t_f} l(\tau,\mathbf{x},\mathbf{u}) d\tau\right] \qquad (25)
\end{aligned}$$

where $t_f = t + H$ and $\mathbf{x}_f := \mathbf{x}(t + H)$. We here truncate the horizon of optimization to the time horizon of MPC. To compensate for the truncated cost, we use the optimal value function as the termination cost.

When the actor uses an MPC strategy, it only has access to the approximated value function.

$$\mathbb{E}_{p^\pi}\left[C(t,\mathbf{s})\right] = \mathbb{E}_{p^\pi}\left[\gamma^H \hat{V}(t_f,\mathbf{x}_f) + \int_t^{t_f} l(\tau,\mathbf{x},\mathbf{u})d\tau\right] \tag{26}$$

We can then write the KL divergence as

$$\mathrm{KL}(p^\pi\|p^*) = \frac{1}{\lambda}\mathbb{E}_{p^\pi}\left[\gamma^H \hat{V}(t_f,\mathbf{x}_f) + \int_t^{t_f} l(\tau,\mathbf{x},\mathbf{u})d\tau\right] - \frac{1}{\lambda}\mathbb{E}_{p^*}\left[\gamma^H V^*(t_f,\mathbf{x}_f) + \int_t^{t_f} l(\tau,\mathbf{x},\mathbf{u})d\tau\right]$$

Adding and subtracting $\frac{1}{\lambda}\mathbb{E}_{p^\pi}\left[\gamma^H V^*(t_f,\mathbf{x}_f) + \int_t^{t_f} l(\tau,\mathbf{x})d\tau\right]$

$$\begin{aligned}\mathrm{KL}(p^\pi\|p^*) =& \frac{\gamma^H}{\lambda}\mathbb{E}_{p^\pi}\left[\hat{V}(t_f,\mathbf{x}_f) - V^*(t_f,\mathbf{x}_f)\right] + \frac{1}{\lambda}\mathbb{E}_{p^\pi}\left[\gamma^H V^*(t_f,\mathbf{x}_f) + \int_t^{t_f} l(\tau,\mathbf{x},\mathbf{u})d\tau\right] \\ &- \frac{1}{\lambda}\mathbb{E}_{p^*}\left[\gamma^H V^*(t_f,\mathbf{x}_f) + \int_t^{t_f} l(\tau,\mathbf{x},\mathbf{u})d\tau\right]\end{aligned} \tag{27}$$

Using our assumption $\max_{t,s}|\hat{V}(t,\mathbf{s}) - V^*(t,\mathbf{s})| = \mathcal{L}$,

$$\begin{aligned}\mathbb{E}_{p^\pi}\left[\gamma^H V^*(t_f,\mathbf{x}_f) + \int_t^{t_f} l(\tau,\mathbf{x},\mathbf{u})d\tau\right] &\le \mathbb{E}_{p^\pi}\left[\gamma^H \hat{V}(t_f,\mathbf{x}_f) + \int_t^{t_f} l(\tau,\mathbf{x},\mathbf{u})d\tau\right] + \gamma^H\mathcal{L} \\ \mathbb{E}_{p^*}\left[\gamma^H V^*(t_f,\mathbf{x}_f) + \int_t^{t_f} l(\tau,\mathbf{x},\mathbf{u})d\tau\right] &\ge \mathbb{E}_{p^*}\left[\gamma^H \hat{V}(t_f,\mathbf{x}_f) + \int_t^{t_f} l(\tau,\mathbf{x},\mathbf{u})d\tau\right] - \gamma^H\mathcal{L}\end{aligned}$$

using these bounds in Equation (27), we get

$$\begin{aligned}\mathrm{KL}(p^\pi\|p^*) \le& \frac{\gamma^H}{\lambda}\mathbb{E}_{p^\pi}\left[\hat{V}(t_f,\mathbf{x}_f) - V^*(t_f,\mathbf{x}_f)\right] + \frac{1}{\lambda}\mathbb{E}_{p^\pi}\left[\gamma^H \hat{V}(t_f,\mathbf{x}_f) + \int_t^{t_f} l(\tau,\mathbf{x},\mathbf{u})d\tau\right] \\ &- \frac{1}{\lambda}\mathbb{E}_{p^*}\left[\gamma^H \hat{V}(t_f,\mathbf{x}_f) + \int_t^{t_f} l(\tau,\mathbf{x},\mathbf{u})d\tau\right] + 2\frac{\gamma^H\mathcal{L}}{\lambda}\end{aligned} \tag{28}$$

Since p_π is generated to minimize $\mathbb{E}_{p^\pi}\left[\gamma^H \hat{V}(t_f,\mathbf{x}_f) + \int_t^{t_f} l(\tau,\mathbf{x},\mathbf{u})d\tau\right]$, therefore

$$\mathbb{E}_{p^\pi}\left[\gamma^H \hat{V}(t_f,\mathbf{x}_f) + \int_t^{t_f} l(\tau,\mathbf{x},\mathbf{u})d\tau\right] \le \mathbb{E}_{p^*}\left[\gamma^H \hat{V}(t_f,\mathbf{x}_f) + \int_t^{t_f} l(\tau,\mathbf{x},\mathbf{u})d\tau\right]$$

We then have

$$\begin{aligned}\mathrm{KL}(p^\pi\|p^*) \le& \frac{\gamma^H}{\lambda}\mathbb{E}_{p^\pi}\left[\hat{V}(t_f,\mathbf{x}_f) - V^*(t_f,\mathbf{x}_f)\right] + 2\frac{\gamma^H\mathcal{L}}{\lambda} \\ \le& \gamma^H\left[\mathrm{KL}(p_f^\pi\|p_f^*)\right] + 2\frac{\gamma^H\mathcal{L}}{\lambda}\end{aligned} \tag{29}$$

Recursively applying the KL bound to $\mathrm{KL}(p_f^\pi\|p_f^*)$, we get

$$\begin{aligned}\mathrm{KL}(p^\pi\|p^*) \le& \frac{2\mathcal{L}\gamma^H}{\lambda}\left(1 + \gamma^H + \gamma^{2H} + \cdots\right) \\ \le& \frac{2\mathcal{L}\gamma^H}{\lambda(1-\gamma^H)}\end{aligned}$$

□

A.4 Proof of Proposition 1

First, we note that HJB equation for the problem defined in Equations (1)-(4) with $\mathbf{\Sigma R} = \lambda \mathbf{I}$ has the form

$$-\partial_t V^*(t,\mathbf{x}) = l(t,\mathbf{x}) + \partial_{\mathbf{x}} V^*(t,\mathbf{x})^\top \mathbf{f}(t,\mathbf{x}) - \frac{1}{2}\partial_{\mathbf{x}} V^*(t,\mathbf{x})^\top \mathbf{\Xi}(t,\mathbf{x})\, \partial_{\mathbf{x}} V^*(t,\mathbf{x}) + \frac{\lambda}{2}\mathtt{Tr}[\partial^2_{\mathbf{x}} V^*(t,\mathbf{x})\, \mathbf{\Xi}(t,\mathbf{x})], \tag{30}$$

with $V^*(T,\mathbf{x}) = \phi(\mathbf{x}(T))$ and $\mathbf{\Xi}(t,\mathbf{x}) = \mathbf{g}(t,\mathbf{x})\mathbf{R}^{-1}\,\mathbf{g}(t,\mathbf{x})^\top = \lambda\,\mathbf{g}(t,\mathbf{x})\mathbf{\Sigma}\,\mathbf{g}(t,\mathbf{x})^\top$. For sake of brevity, we define $l(t,\mathbf{x}) := \gamma^{t-t_0} q(t,\mathbf{x})$ and $\phi(\mathbf{x}) := \gamma^{T-t_0} q_f(\mathbf{x})$.

The optimal control policy can be derived as

$$\pi^*(t,\mathbf{x}) = -\mathbf{R}^{-1}\mathbf{g}(t,\mathbf{x})^\top \partial_{\mathbf{x}} V^*(t,\mathbf{x}). \tag{31}$$

Lemma 1. *The optimal control policy of the stochastic problem in equations (1)-(4) is the optimal solution to a deterministic problem with following cost functional*

$$C^\pi_d(t_0,\mathbf{s}_0) = \gamma^{T-t_0} q_f(\mathbf{s}(T)) + \int_{t_0}^{T} \Big(\gamma^{t-t_0} q(t,\mathbf{s}) + \frac{\lambda}{2}\,\mathrm{Tr}[\mathbf{\Xi}(t,\mathbf{s})\partial^2_s V(t,\mathbf{s})] + \frac{1}{2}\mathbf{u}^\top \mathbf{R}\,\mathbf{u}\Big) dt \tag{32}$$

Where the state evolution is based on the mean of the system dynamics defined in (1), i.e.,

$$\dot{\mathbf{s}} = \mathbf{f}(t,\mathbf{s}) + \mathbf{g}(t,\mathbf{s})\mathbf{u}, \quad \mathbf{s}(t_0) = \mathbf{s}_0 \tag{33}$$

$$\mathbf{u} = \pi(t,\mathbf{s}). \tag{34}$$

Proof. The proof easily follows using the definition of the HJB equation for the stochastic and deterministic optimal control problems. □

We finally provide the proof of Proposition 1.

Proof. Start from the Girsanov theorem and Equation (23), we get

$$\begin{aligned}
\mathrm{KL}(p^\pi \| p^*) =& \mathbb{E}_{p^\pi}\left[\int_t^T \frac{1}{2}\|\mathbf{u}_\pi(\tau) - \mathbf{u}^*(\tau)\|^2_{\mathbf{R}} d\tau\right] \\
=& \mathbb{E}_{p^\pi}\Bigg[\int_t^T \Big(\frac{1}{2}\|\mathbf{u}_\pi(\tau)\|^2_{\mathbf{R}} + \partial_{\mathbf{x}} V^*(s,\mathbf{x})^\top \mathbf{g}(\tau,\mathbf{x})\mathbf{u}_\pi(\tau)) \\
&\qquad + \frac{1}{2}\partial_{\mathbf{x}} V^*(\tau,\mathbf{x})^\top \mathbf{\Xi}(\tau,\mathbf{x})\, \partial_{\mathbf{x}} V^*(\tau,\mathbf{x}))\Big) d\tau\Bigg] \qquad (35) \\
=& \mathbb{E}_{p^\pi}\Bigg[V^*(T,\mathbf{x}(T)) + \int_t^T \Big(\frac{1}{2}\|\mathbf{u}_\pi(\tau)\|^2_{\mathbf{R}} - \partial_t V^*(\tau,\mathbf{x}) \\
&\quad - \partial_{\mathbf{x}} V^*(\tau,\mathbf{x})^\top \mathbf{f}(\tau,\mathbf{x}) - \frac{1}{2}\,\mathrm{Tr}\left[\mathbf{\Xi}(\tau,\mathbf{x}))\partial^2_{\mathbf{x}} V^*(\tau,\mathbf{x}))\right] \\
&\quad + \frac{1}{2}\partial_{\mathbf{x}} V^*(\tau,\mathbf{x})^\top \mathbf{\Xi}(\tau,\mathbf{x})\, \partial_{\mathbf{x}} V^*(\tau,\mathbf{x})\Big) d\tau\Bigg] - V^*(t,\mathbf{x}) \qquad (36) \\
=& \mathbb{E}_{p^\pi}\Bigg[V^*(T,\mathbf{x}(T)) + \int_t^T \Big(L_e(\tau,\mathbf{x}) + \frac{1}{2}\|\mathbf{u}_\pi(\tau)\|^2_{\mathbf{R}} \\
&\quad - \frac{1}{2}\,\mathrm{Tr}\left[\mathbf{\Xi}(\tau,\mathbf{x})\partial^2_{\mathbf{x}} V^*(\tau,\mathbf{x}))\right]\Big) d\tau\Bigg] - V^*(t,\mathbf{x}) \qquad (37)
\end{aligned}$$

In Equation (35) we have replaced $\mathbf{u}^*$ using Equation (31); in Equation (36) we have used Itô's Lemma for the process $\mathcal{V}^*(t) = V^*(t,\mathbf{x})$

$$d\mathcal{V}^*(t) = \partial_t V^*(t,\mathbf{x})dt + \partial_x V^*(t,\mathbf{x})^\top \big(\mathbf{f}(t,\mathbf{x}) + \mathbf{g}(t,\mathbf{x})\mathbf{u}_\pi(t,\mathbf{x})\big)dt + \frac{\lambda}{2}\,\mathrm{Tr}\left[\mathbf{\Xi}(t,\mathbf{x})\partial^2_{\mathbf{x}} V^*(t,\mathbf{x})\right]dt + \partial_{\mathbf{x}} V^*(t,\mathbf{x})^\top \mathbf{g}(t,\mathbf{x})d\mathcal{B}(t). \tag{38}$$

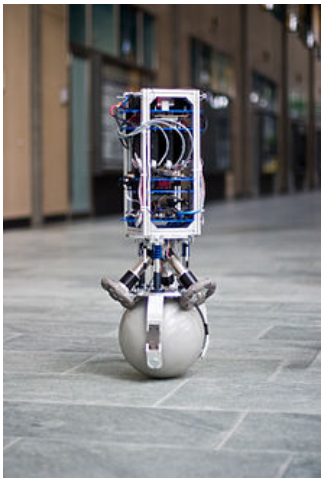


Figure 4: Image of the Ballbot used in the experiments. Ballbot is a torque-controlled, omnidirectional robot which balances on a ball through three omni-wheels.

After reordering the terms in Equation (38), taking the time integral over the interval $[t, T]$, and taking the expectation with respect to p^{π}, we get

$$\mathbb{E}_{p^{\pi}}\left[\int_t^T \partial_{\mathbf{x}} V^*(\tau,\mathbf{x})^\top \mathbf{g}(\tau,\mathbf{x})\mathbf{u}_\pi(\tau)d\tau\right] = \mathbb{E}_{p^{\pi}}\Bigg[V^*(T,\mathbf{x}) - \int_t^T \Big(\partial_t V^*(\tau,\mathbf{x}) + \partial_{\mathbf{x}} V^*(\tau,\mathbf{x})^\top \mathbf{f}(t,\mathbf{x}) + \frac{\lambda}{2}\operatorname{Tr}\left[\boldsymbol{\Xi}(\tau,\mathbf{x})\partial_{\mathbf{x}}^2 V^*(\tau,\mathbf{x})\right]\Big)d\tau\Bigg] - V^*(t,\mathbf{x}) \tag{39}$$

where the term involving $d\mathcal{B}(t)$ cancels out.
As a result, the policy $\boldsymbol{\pi}$ which minimizes the reverse KL-divergence can be derived as

$$\pi^*(t,\mathbf{s}) = \arg\min_{\pi} \mathbb{E}_{p^{\pi}}\Bigg\{V^*(T,\mathbf{x}(T)) + \int_t^T \Big(l_d(\tau,\mathbf{x}) + \frac{1}{2}\mathbf{u}_\pi^\top \mathbf{R}\,\mathbf{u}_\pi - \frac{\lambda}{2}\operatorname{Tr}\left[\boldsymbol{\Xi}(\tau,\mathbf{x})\partial_{\mathbf{x}}^2 V^*(\tau,\mathbf{x})\right]\Big)d\tau\Bigg\} \tag{40}$$

This expectation is based on the probability distribution generated by the stochastic system in (1). According to Lemma 1, we can transform this optimization problem to an equivalent deterministic problem in Equation (18). □

B Experimental Details

Platform The Ballbot robot depicted in Figure 4 is a 3D inverted pendulum capable of balancing on a ball with the help of three actuators. The mathematical formulation of system dynamics can be found in [23]. Since it balances on a single ball, it has dynamic stability, is omnidirectional, and is capable of carrying out agile movements. Due to its inherent instability and highly nonlinear dynamics, this robot can also be used as a testing platform to validate general control algorithms, which may be applied to other types of mobile platforms.

Experiments An advantage of using an MPC strategy is that a nominal controller can be used to stabilize the system. In these experiments, we use an additional cost term in the trajectory optimizer that stabilizes the system in an upright position. As a result, the system will stand from the beginning resulting in much faster convergence to the desired behaviour. This is not a limitation of the pipeline, since the value function could also be trained to encode the upright stabilization.

In Figure 2b, we tuned the hyper-parameters to achieve the best performance for each scenario. The performance at each learning iteration is computed by taking the average performance over 8 trajectories sampled from different starting positions.