Learning optimal controllers: a dynamical motion primitive approach ⋆

Hugo T. M. Kussaba [*] **Abdalla Swikir** [*,**,****] **Fan Wu** [*]
Anastasija Demerdjieva [*] **Gitta Kutyniok** [***]
Sami Haddadin [*,****]

[*] *Chair of Robotics and Systems Intelligence, Munich Institute of Robotics and Machine Intelligence, Technical University of Munich, Germany.*
[**] *Department of Electrical and Electronic Engineering, Omar Al-Mukhtar University, Libya.*
[***] *Mathematical Institute, Ludwig-Maximilians University of Munich, Germany.*
[****] *Centre for Tactile Internet with Human-in-the-Loop (CeTI), Germany.*

Abstract: Real-time computation of optimal control is a challenging problem and, to solve this difficulty, many frameworks proposed to use learning techniques to learn (possibly sub-optimal) controllers and enable their usage in an online fashion. Among these techniques, the optimal motion framework is a simple, yet powerful technique, that obtained success in many complex real-world applications. The main idea of this approach is to take advantage of dynamic motion primitives, a widely used tool in robotics to learn trajectories from demonstrations. While usually these demonstrations come from humans, the optimal motion framework is based on demonstrations coming from optimal solutions, such as the ones obtained by numeric solvers. As usual in many learning techniques, a drawback of this approach is that it is hard to estimate the suboptimality of learned solutions, since finding easily computable and non-trivial upper bounds to the error between an optimal solution and a learned solution is, in general, unfeasible. However, we show in this paper that it is possible to estimate this error for a broad class of problems. Furthermore, we apply this estimation technique to achieve a novel and more efficient sampling scheme to be used within the optimal motion framework, enabling the usage of this framework in some scenarios where the computational resources are limited.

Keywords: Real-time optimal control, Learning from control, Autonomous robotic systems

1. INTRODUCTION

Optimal control problems (OCPs) arise in many applications in robotics, ranging from motion planning (LaValle, 2006) to robot design (Dinev et al., 2022). Generally, analytical solutions are not known for these problems, which then must be solved by numerical methods such as direct methods (Diehl et al., 2006). These methods transcribe the optimal control problem into a non-linear programming (NLP) problem that can be usually solved by interior-point methods even for non-convex problems (Nocedal et al., 2009). Nevertheless, many problems in robotics result in an NLP with a large number of variables and constraints. Hence, finding numerical solutions for those problems requires extensive computational power which in turn makes the usage of OCP solvers intractable for high dimensional systems (such as robotics systems with many degrees of freedom), especially in real-time.

Common ways to overcome this restriction and enable the online usage of optimal controllers are based on exploiting *a priori* information of the solution (Harzer et al., 2022) and/or by providing near-optimal solutions. The latter approach, for instance, has been explored several times in model-predictive controllers (MPC), where embedded applications with low computational power or limited amount of energy prohibit solving the nonlinear optimization problem online (Pohlodek et al., 2022).

In view of this, machine-learning based techniques which exploit the structure of the problem to yield approximate optimal controllers are a natural candidate for obtaining near-optimal solutions in real-time. Indeed, the intersection between machine learning and optimization problems has shown promising results ranging from combinatorial optimization problems (Bengio et al., 2021), MPC controllers (E. et al., 2022; Karg and Lucia, 2020) and solv-

⋆ This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement no. 899987. This work was also supported by the German Research Foundation (DFG) as part of Germany's Excellence Strategy, EXC 2050/1, Project ID 390696704 – Cluster of Excellence "Centre for Tactile Internet with Human-in-the-Loop" (CeTI) of Technische Universität Dresden. The authors would also like to thank Luis F.C. Figueredo and Dennis Ossadnik for the scientific discussions.

ing Hamilton-Jacobi-Bellman partial differential equation (Bansal and Tomlin, 2021).

In particular, some supervised learning methods leverage the numerical solution of optimal control problems to generate training data for teaching a function approximator to the optimal control solution. This "Learning from Optimal Control" (LfOC) approach has been successfully demonstrated for real-time quadrotor aggressive maneuvers in Tomic et al. (2014) and throwing/reaching tasks for robotic manipulators in Haddadin et al. (2013), where dynamical motion primitives (DMPs), also referred to as dynamical movement primitives, act as the function approximator for the optimal control solution.

While the aforementioned LfOC methods are not burdensome when used online, the generation of training datasets can be very time-expensive since it requires solving multiple OCPs. Moreover, missing an important sample in the training dataset could lead to very costly solutions. Furthermore, there are no results so far on defining a metric to measure the performance of learned controllers compared to the optimal ones. Thus, it is a practical concern how to define a metric that quantifies the error between the out-of-sample solutions and the real optimal solution, and establishes a framework to choose the most representative solutions for decreasing such an error.

In this work, we develop the basis for such a framework by leveraging an important information obtained from the training trajectories: their optimal costs and the sensitivity of the optimal value function corresponding to the initial condition of these trajectories. Based on this information, we devise a way to estimate the suboptimality of out-of-sample solutions and, moreover, show how to take advantage of this estimate for efficiently selecting training data.

2. PRELIMINARIES

2.1 Optimal control problems

In this paper, we consider systems of the form

$$\dot{x}(t) = f(x(t)) + g(x(t))u(t) \tag{1}$$

where $x \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}^m$ is the control input, and $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ are smooth functions describing the system dynamics.

Motion planning problems linked to (1) are frequently cast as an optimal control problem of the form

$$\underset{x\in\mathcal{X},u\in\mathcal{U}}{\text{minimize}} \int_0^{t_f} L(x(t),u(t))\,dt \tag{2a}$$

subject to

$$\dot{x}(t) = f(x(t)) + g(x(t))u(x(t))),\ t \in [0,t_f], \tag{2b}$$
$$x(0) = x_0,\ x(t_f) = x_f,$$

where $\mathcal{X}$ is the set of differentiable functions $x : [0,t_f] \to \mathbb{R}^n$, $\mathcal{U}$ is the set of continuous functions $u : [0,t_f] \to \mathbb{R}^m$ with $u(0) = 0$, and $L : \mathcal{X}\times\mathcal{U} \to \mathbb{R}$ is the running cost, given by

$$L(x(\cdot),u(\cdot)) = Q(x) + u^{\mathsf{T}}Ru,$$

with $Q : \mathbb{R}^n \to \mathbb{R}$ either identically to zero, or a continuous and positive definite function, and $R \in \mathbb{R}^{m\times m}$ is a positive-definite symmetric matrix. Moreover, $x_0 \in \mathbb{R}^n$ is a given initial state, $x_f \in \mathbb{R}^n$ and $t_f > 0$ are respectively the terminal state and terminal time.

We assume that (2) has a unique solution for each $x_f \in \mathbb{P}$, where $\mathbb{P} \subseteq \mathbb{R}^n$ is a compact set, and denote the optimal state trajectory and optimal control respectively by x^* and u^*. Many problems in motion planning require solving (or at least finding an approximate solution) of (2) for multiple instances. In particular, given a compact set $\mathbb{P}$ with the desired points to be reached, we would need to solve one instance of (2) for each fixed $x_f \in \mathbb{P}$.

In many real-world applications, it is often impossible to find a closed-form solution for (2), and numerical methods that transform the optimal control problem into a nonlinear programming problem are widely used (Diehl et al., 2006). Those methods, known as direct methods, are based on the time discretization of the state and control variables. Thus, the obtained solution from the NLP is generally inferior to the solution of the original OCP and a finer discretization may be needed to improve the solution. The price of making the discretization finer is increasing the cost of optimization by requiring more iterations to be performed within the larger search space (Sahlodin and Barton, 2017). Consequently, one can not use direct methods in real-time in general. Indeed, even if $\mathbb{P}$ is a finite set, it can be very time-consuming to produce optimal solutions that cover the entire set $\mathbb{P}$.

2.2 Review of DMP-based optimal motion framework

Since in general it is not possible to obtain a closed form solution to (2) and the complexity of numerical methods limits their usage in real-time, many works in the literature propose to leverage learning techniques to obtain near-optimal solutions in real time. In this context, Haddadin et al. (2013) proposed a learning framework for optimal control based on dynamical motion primitives (DMPs). Before reviewing this framework, we will briefly review the theory of DMPs and the features that make them an attractive tool within the context of a learning optimal control framework. For more in-depth details of DMPs, the reader is referred to Saveriano et al. (2021).

Dynamic movement primitives Following the notation of Weitschat et al. (2013), a DMP is a dynamical system defined by the differential equation

$$-\tau^2\ddot{x}(t) + \kappa(x_f - x(t)) - D\tau\dot{x}(t) = F(s(t)) \tag{3}$$

where $\tau > 0$, $\kappa > 0$ and $D > 0$ are tuning parameters, and $F : \mathbb{R} \to \mathbb{R}^n$ and $s : \mathbb{R} \to \mathbb{R}$ are functions defined as

$$F_i(s(t)) = \frac{\sum_{j=1}^N \omega_{i,j}\, e^{-h_{i,j}(s(t)-c_{i,j})^2}}{\sum_{j=1}^N e^{-h_{i,j}(s(t)-c_{i,j})^2}} s(t), \tag{4}$$

$$s(t) = e^{-(\alpha/\tau)t}, \tag{5}$$

with $\alpha > 0$. The forcing function F is a sum of N Gaussian basis functions with center points $c_{i,j}$ and widths $h_{i,j}$ given by

$$c_{i,j} = e^{\left(-\alpha\frac{j-1}{N-1}\right)},$$

$$h_{i,j} = \begin{cases} (c_{i,j+1} - c_{i,j})^{-2}, & j = 1,\dots,N-1, \\ h_{i,N-1} & j = N, \end{cases}$$

and the weights $\omega_{i,j}$ are learned from data by minimizing the error given by

$$\sum_k \| -\tau^2 \ddot{x}(t_k) + \kappa(x_f - x(t_k)) - D\tau\dot{x}(t_k) - F(s(t_k))\|, \quad (6)$$

where $x(t_k)$, $\dot{x}(t_k)$ and $\ddot{x}(t_k)$ are obtained by sampling a given trajectory with starting point at $x(0) = x_0$ and ending point in $x(\tau) = x_f$. By the next theorem, solutions of (3) will always converge to the latter point.

Theorem 1. (Ijspeert et al., 2013, Sec. 2.1.3) Suppose that $\kappa = D^2/4$. Then the parameter x_f in (3) is a global asymptotic stable equilibrium point.

After learning the weights $\omega_{i,j}$ from a single trajectory $\tilde{x} : \mathbb{R} \to \mathbb{R}^n$ it is possible to use (3) to generate new trajectories that are qualitatively similar to $\tilde{x}$, but with different ending points. In exact terms, to reproduce a new trajectory with ending point in x'_f, one integrates the differential equation

$$-\tau^2 \ddot{x}(t) + \kappa(x'_f - x(t)) - D\tau\dot{x}(t) = F(s(t)), \; x(0) = x_0. \quad (7)$$

It is clear that the only difference between (3) and (7) are the parameters x_f and x'_f. Also note that the same forcing term F learned in (3) is used without changes and there is no need to re-learn it. The convergence of the new trajectory to x'_f is guaranteed by Theorem 1.

For distinguishing between these trajectories more easily, we will introduce the following notation: the trajectory obtained by solving (3) with initial condition $x(0) = x_0$ will be denoted as $\Phi_D(\mathrm{x}_f|\mathrm{x}_f)$, while the trajectory obtained by solving (7) as $\Phi_D(\mathrm{x}'_f|\mathrm{x}_f)$.

Upper bounds to the distance between $\Phi_D(\mathrm{x}_f|\mathrm{x}_f)(t)$ and $\Phi_D(\mathrm{x}'_f|\mathrm{x}_f)(t)$ at each time t can be quantitatively expressed as a function of the distance between x_f and x'_f. This follows from the fact that the differential equation in (3) has a continuous dependence on the parameter x_f (Perko, 2001). These bounds, however, depend on the computation of (local) Lipschitz constants, whereas in this work we show that is possible to obtain an easy computable expression for the distance between $\Phi_D(\mathrm{x}_f|\mathrm{x}_f)(t)$ and $\Phi_D(\mathrm{x}'_f|\mathrm{x}_f)(t)$ in terms of the DMP parameters and the distance between x_f and x'_f:

Proposition 2. The following equality is valid for all $t \in [0, t_f]$:

$$\|\Phi_D(x_f \mid x_f)(t) - \Phi_D(x'_f \mid x_f)(t)\| = \left| e^{-\frac{Dt}{2\tau}} \left(-\frac{Dt}{2\tau} - 1 \right) + 1 \right| \|x_f - x'_f\|.$$

Proof 1. See Appendix.

Learning controllers with optimal motion primitives The DMP-based framework relies on the assumption that optimal trajectories that have close terminal end-points have qualitatively similar shapes. While this assumption seems very restrictive, it has been verified in practice for reaching/tracking motions with minimal energy in elastic robots in Weitschat et al. (2013) and time-optimal maneuvers for quadrotors in Tomic et al. (2014).

Under this assumption, the optimal trajectories obtained by numerically solving (2) will be used to create DMPs with different x_f's at each point of a finite grid $\hat{\mathbb{P}}$ contained in $\mathbb{P}$. If a terminal state is outside $\hat{\mathbb{P}}$, it should be inside a hypercube with vertices in $\hat{\mathbb{P}}$, and a trajectory can be computed almost instantaneously by "blending" the weights of the DMPs in the vertices. For instance, Tomic et al. (2014) blends the weights by bilinear interpolation while Haddadin et al. (2013) uses a weighted interpolation based on the costs of the trajectories. The new DMP is then used to instantaneously generate a new trajectory to the unsampled terminal state.

Assuming that there exists a well-defined function

$$u := u(x, \dot{x}, \ddot{x}, \ldots, x^{(m)}), \quad (8)$$

where $x^{(m)}$ is the m-th derivative of x with respect to time, the control input u associated with a trajectory x can be directly recovered from the DMP trajectories. In particular, if g in (1) is invertible, then u can be recovered with

$$u(t) = g^{-1}(x(t))(\dot{x}(t) - f(x(t)))$$

While in general the new solutions will not be optimal, they will be close to it and the stability of the terminal state is guaranteed by Theorem 1.

Finally, it is interesting to highlight that many extensions of the DMP framework can be inherited by this optimal motion framework. For instance, Tomic et al. (2014) used the possibility of combining multiple DMPs to join a sequence of maneuvers for a quadrotor.

3. ESTIMATION OF OPTIMAL VALUE AND SAMPLING ALGORITHM

The DMP-based approach for learning optimal control requires storing KN weights in memory, where N is the number of basis functions used in (3) and K is the number of sample trajectories obtained by solving (2) for different $x_f \in \mathbb{P}$. While Weitschat et al. (2013) proposed a technique for finding a minimal number N of basis functions, the product KN can still be large depending on the requirements of the application. Moreover, some complex trajectories may require a high number K to obtain good generalization properties, such as time-optimal perching maneuvers in quadrotors (Tomic et al., 2014).

To reduce K, one may consider to avoid sampling trajectories whenever the cost of the out-of-sample DMP is good enough for a given application. For that, one needs to compute (or at least estimate) how suboptimal a learned trajectory is. To achieve this cost-aware sampling strategy, we propose to leverage the sensitivity information in the learned trajectories, i.e. the trajectories that are obtained by solving (2).

In the next subsection we show how this can be done and how it can be used to estimate the value function in a neighborhood of the ending points of the learned trajectories. We also show how to apply this approximation to efficiently create a sampling grid for learning optimal control when there are restrictions on the number of samples.

3.1 First-order approximation to the value function

In order to estimate the suboptimality of an out-of-sample trajectory $\Phi_D(\mathrm{x}'_f|\mathrm{x}_f)$, we will need to estimate the cost of the true optimal trajectory. To this end, we will get the cost of optimal trajectories by using the notion of the value function, whose definition we precisely state next.

Following p. 159 of Liberzon (2011), the value function $V : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ for (2) is defined as

$$V(x,t) = \inf_{u \in \mathcal{U}} \left\{ \int_t^{t_f} L(x(s), u(s))\ ds \right\}. \tag{9}$$

In other words, assuming the existence of an optimal control, $V(x,t)$ is the cost of the optimal trajectory that starts from the initial state x at the time t.

Assuming that V is differentiable on $\mathbb{P}$, it is possible to use Bellman's optimality principle to show that V is the solution of a partial differential equation (known as the Hamilton-Jacobi-Bellman equation) and that it is possible to recover the solution of (2) from this V. This is precisely stated for systems of the form (1) as follows. Suppose that (1) is Lipschitz continuous and stabilizable on $\Omega \subseteq \mathbb{R}^n$, and that $f(0) = 0$. Assume that there exists a differentiable function $V : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}$ satisfying the partial differential equation given by [1]

$$\frac{\partial V}{\partial t}(x,t) = -\frac{\partial V}{\partial x}(x,t) f(x) + \frac{1}{4} \left\| R^{-\frac{1}{2}} \frac{\partial V}{\partial x}^{\mathsf{T}}(x,t) g(x) \right\|^2. \tag{10}$$

Then the optimal control solving (2) is given by

$$u^*(x,t) = -\frac{1}{2} R^{-1} g^{\mathsf{T}}(x) \frac{\partial V}{\partial x}(x,t). \tag{11}$$

Given an optimal trajectory u^* obtained by numerically solving (2) and assuming (10) holds, it is possible to recover $\frac{\partial V}{\partial x}$ along this optimal trajectory:

$$\frac{\partial V}{\partial x}(x^*(t),t) = -2 g^{-\mathsf{T}}(x^*(t)) R u^*(x^*(t),t). \tag{12}$$

In particular, $\frac{\partial V}{\partial x}(x^*(0),0)$ can be interpreted as the sensitivity of the optimal cost V with respect to changes in the initial condition $x^*(0)$—for more details, the reader is referred to (Kamien and Schwartz, 2012). In the DMP-based framework, however, the initial state is fixed while the terminal state $x_f \in \mathbb{P}$ is changed. To enable the use of (12) in our framework, we will use Lemma 3 below.

Lemma 3. The optimal cost of the OCP

$$\underset{z(\cdot), v(\cdot)}{\text{minimize}} \int_0^{t_f} L(z(t), v(t))\, dt \tag{13a}$$

subject to

$$\begin{aligned} &\dot{z}(t) = -f(z(t)) - g(z(t)) v(t),\ t \in [0, t_f], \\ &z(0) = x_f,\ z(t_f) = x_0 \end{aligned} \tag{13b}$$

is equal to the optimal cost of (2), and the optimal functions are related by

$$z^*(t) = x^*(t_f - t),\ v^*(t) = u^*(t_f - t). \tag{14}$$

Proof 2. Define the functions z and v as

$$z(t) = x(t_f - t),\ v(t) = u(t_f - t).$$

By definition, $z(0) = x_f$ and $z(t_f) = x_0$ if and only if $x(0) = x_0$ and $x(t_f) = x_f$.

Moreover, since $\dot{z}(t) = -\dot{x}(t_f - t)$, we have by (1) that

$$\begin{aligned} \dot{z}(t) &= -f(x(t_f - t)) - g(x(t_f - t)) u(t_f - t) \\ &= -f(z(t)) - g(z(t)) v(t). \end{aligned}$$

Finally, performing integration by substitution we can show that the cost of the trajectories is the same:

$$\begin{aligned} \int_0^{t_f} L(x(t), u(t))\, dt &= \int_{t_f}^0 -L(x(t_f - t), u(t_f - t))\, dt \\ &= \int_0^{t_f} L(z(t), v(t))\, dt. \end{aligned}$$

Thus, solutions of (2b) can be mapped to solutions of (13b) with same cost, and by similar arguments, the converse also holds. In particular, the optimal cost of the two OCP problems is the same. □

Lemma 3 allows us to convert a (parametric) optimal control problem in which the initial condition is fixed and the terminal condition is varying in $\mathbb{P}$ to a problem in which the initial condition changes in $\mathbb{P}$ while the terminal condition is fixed. How this helps us to achieve first-order estimates for the optimal value function is explained next.

First, we solve [2] the optimal control problem (13), obtaining a backward-time trajectory z^* and its associated control input v^*. In particular, we obtain the value of v^* which starts at time 0 in x_f, which will be used in the final step.

Second, we define x^* using (14) and encode the forward-time trajectory solution in the DMP so we can generate a trajectory $\Phi_D(x'_f \mid x_f)$ to a new goal $x'_f \in \mathbb{P}$.

Finally, to estimate the suboptimality of the new out-of-sample trajectory, we will need to consider the value function associated with the backward optimal control problem (13), which is defined as

$$\tilde{V}(x,t) := \inf_{u \in \mathcal{U}} \left\{ \int_t^{t_f} L(x(t_f - s), u(t_f - s))\ ds \right\},$$

and note that $\tilde{V}(x'_f, 0)$ is, by Lemma 3, the cost of the optimal trajectory with same endpoints as $\Phi_D(x'_f \mid x_f)$.

To compute $\tilde{V}(x'_f, 0)$, note that the differentiability of V (with respect to x) implies (see Corollary 1.24 of Güler (2010))

$$\tilde{V}(x'_f, 0) = \tilde{V}(x_f, 0) + \frac{\partial \tilde{V}}{\partial x}(x_f, 0)(x'_f - x_f) + o(\|x'_f - x_f\|), \tag{15}$$

where $o(\|x'_f - x_f\|)$ is an (unknown) function that converges to 0 as $\|x'_f - x_f\|$ tends to 0.

Moreover, the term $\tilde{V}(x_f, 0)$ of the right-hand side of (15) is already computed by the first step, and the term $\frac{\partial \tilde{V}}{\partial x}(x_f, 0)$ can be computed applying (12) for the backward-time system, that is [3]

$$\frac{\partial \tilde{V}}{\partial x}(x_f, 0) = 2 g^{-\mathsf{T}}(x_f) R v^*(x_f, 0), \tag{16}$$

where $v^*(x_f, 0)$ was also computed in the first step. Figure 1 summarizes the above steps.

[1] Similar to Isidori and Astolfi (1992) and van der Schaft (1992), we have assumed the existence of differentiable solutions to (10). We refer the interested readers to (Cheng et al., 2007, Remark 1) for a discussion about the differentiability of the function $V(x,t)$.

[2] Please note that the we have assumed that the optimal solution is obtained numerically. However, and without losing generality, we only need to assume that an optimal solution is given to us.

[3] Note that here we use g instead of -g because we are applying (12) to the backward-time system.

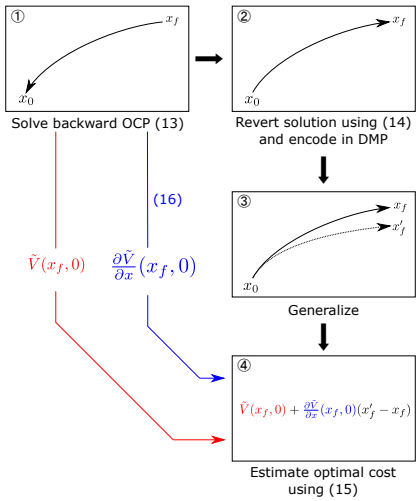

Fig. 1. Diagram illustrating the steps for computing an estimate for the cost of an optimal trajectory that corresponds to an out-of sample trajectory with same endpoints.

In conclusion, (15) enables us to estimate (for sufficiently small values of $\|x'_f - x_f\|$) the suboptimality of $\Phi_D(\mathrm{x}'_f|\mathrm{x}_f)$ by computing

$$\int_0^{t_f} L(\Phi_D(x'_f \mid x_f)(s), \hat{u}_D(s))\, ds - \left[\tilde{V}(x_f, 0) + \frac{\partial \tilde{V}}{\partial x}(x_f, 0)(x'_f - x_f)\right], \tag{17}$$

where $\hat{u}_D$ is the input associated to $\Phi_D(x'_f \mid x_f)$ and is computed by (8).

3.2 Sampling algorithm description

In this section, we explain how the approximation obtained in (17) can be used to design a sampling algorithm for applications in which memory storage restrictions possibly prohibit the usage of a fine grid for $\mathbb{P}$.

The details of the sampling scheme are presented in the pseudo-code of Algorithm 1. First, a terminal point $x_f \in \mathbb{P}$ is chosen and a direction $v \in \mathbb{R}^n$ is given. Then, the backward optimal control with $z(0) = x_f$ is solved and encoded in $\Phi_D(x_f \mid x_f)$.

After that, we select another point in $\mathbb{P}$ by adding a user-defined $\Delta x \in \mathbb{R}^n$ to the starting point. If this new point x'_f is outside $\mathbb{P}$, it means that we do not need to continue any further in this direction and the algorithm stops.

Otherwise, we use the procedure outlined in the last subsection to estimate the suboptimality of the generalization $\Phi_D(\mathrm{x}'_f|\mathrm{x}_f)$ using (17). If the difference between the cost of $\Phi_D(\mathrm{x}'_f|\mathrm{x}_f)$ and the estimated cost of the optimal trajectory ending at x'_f is lower than a user-defined threshold, then it means that the generalization $\Phi_D(\mathrm{x}'_f|\mathrm{x}_f)$ is good enough for the application and there is no need to solve a new optimal control problem for encoding it in a new DMP.

It is important to remark that since (15) is only a first-order approximation, the estimate can become useless if x'_f and x_f are distant. Because of this, the algorithm should re-sample in the case the number of steps given without re-sampling surpass t_{steps}, a user-defined limit. The entire procedure repeats until a number t_{samples} of samples, defined by the user to reflect the limitations of his or her application, are reached.

By repeating the sampling algorithm for each direction orthogonal to v the algorithm can be used to create a n-dimensional grid $\hat{\mathbb{P}}$, whose points are made by the Cartesian product of the samples generated by each direction orthogonal to v.

Finally, a near optimal trajectory can be generated online for each point in $\mathbb{P}$ by using a bilinear or a cost-weighted interpolation in $\hat{\mathbb{P}}$ (Tomic et al., 2014; Haddadin et al., 2013). In either case, it is possible to use (17) again to estimate the suboptimality of each point x'_f in the entire region $\mathbb{P}$ by using (17) with the nearest points x_f of x'_f in $\hat{\mathbb{P}}$.

Remark 4. Though the algorithm presented here was explained within the framework of learning optimal control with DMPs, it could be easily generalized for other frameworks that use an alternative primitive for encoding optimal solutions, for instance, Gaussian processes (Clever et al., 2017).

Remark 5. Note that (12) assumes the invertibility of g. Nevertheless, the sampling scheme presented here can also be applied in cases where it is possible to partially recover some of the derivatives of V by (11), even when g is not invertible. Such is the case of mechanical systems, which can be represented by (1) and satisfy

$$\frac{\partial V}{\partial \dot{q}} = -2M(q)R\tau^*,$$

where q is the vector of generalized coordinates, M is the mass matrix and τ is the optimal input torque.

4. NUMERICAL SIMULATION

In this numerical simulation, we illustrate the application of the proposed method described in Algorithm 1 for the optimal control problem (2) with $L(x(\cdot), u(\cdot)) = u^\mathsf{T} u$ and considering a non-linear system in the form (1), given by

$$\begin{bmatrix}\dot{x}_1(t)\\ \dot{x}_2(t)\end{bmatrix} = \begin{bmatrix}-x_1^2(t)\\ -2x_2(t)\end{bmatrix} + \begin{bmatrix}1 & x_1(t)\\ 0 & 1\end{bmatrix}\begin{bmatrix}u_1(t)\\ u_2(t)\end{bmatrix}. \tag{18}$$

Moreover, the boundary conditions are $x_0 = (5, 5)$ for the initial state $x(0)$, and the terminal state $x(8)$ will be equal to x_f, which will be a varying parameter within

$$\mathbb{P} = \{(x_1, 5) : x_1 \ \in [1, 9]\}.$$

Remark 6. We only considered variations in the x_1 direction since similar results would be obtained by considering variations in the x_2 direction.

Algorithm 1 was executed by taking x_0 as the first sampling point and with direction $v = (1, 0)$. The threshold for the cost difference ($J_{\text{threshold}}$) was chosen as 10, and the maximum number of allowed samples (t_{samples}) in this direction was chosen as 15. The size of the step (Δ_x) used was 0.2, and the maximum number of steps (t_{steps}) allowed without taking any sample was 5.

Algorithm 1 Sampling method for a specific direction v

Input: v: initial direction to start sampling

procedure SAMPLE(v)
  $J_{\text{threshold}} \leftarrow$ threshold of allowed deviation from estimated optimal cost
  $t_{\text{samples}} \leftarrow$ maximum number of allowed samples in $\hat{\mathbb{P}}$
  $\Delta x \leftarrow$ size of step in direction v
  $t_{\text{steps}} \leftarrow$ maximum number of steps
  $x_f \leftarrow$ a given point in $\mathbb{P}$
  $\hat{\mathbb{P}} \leftarrow \emptyset$
  while $|\hat{\mathbb{P}}| \leq t_{\text{samples}}$ **do**
    $x_f \leftarrow x'_f$
    $(z^*, v^*) \leftarrow$ solution of (13) with $z(0) = x_f$
    $x^*(t) \leftarrow z^*(t_f - t)$
    Encode optimal solution x^* in $\Phi_D(\mathrm{x}_f|\mathrm{x}_f)$
    Include x_f in $\hat{\mathbb{P}}$ and store weights $\omega_{i,j}$ of $\Phi_D(\mathrm{x}_f|\mathrm{x}_f)$ in memory
    $V_x \leftarrow$ gradient of $\tilde{V}$ with respect to x computed by (16)
    $J_s \leftarrow$ cost of optimal solution v^*
    $n_{\text{step}} \leftarrow 1$
    do
      $x'_f \leftarrow x'_f + \Delta x \cdot v$
      $\hat{V} \leftarrow J_s + V_x \cdot x'_f$
      $J_{\text{DMP}} \leftarrow$ cost of $\Phi_D(x'_f \mid x_f)$
      $n_{\text{step}} \leftarrow n_{\text{step}} + 1$
    while $J_{\text{DMP}} - \hat{V} < J_{\text{threshold}}$ and $n_{\text{step}} < t_{\text{steps}}$ and $x'_f \in \mathbb{P}$
  end while
end procedure

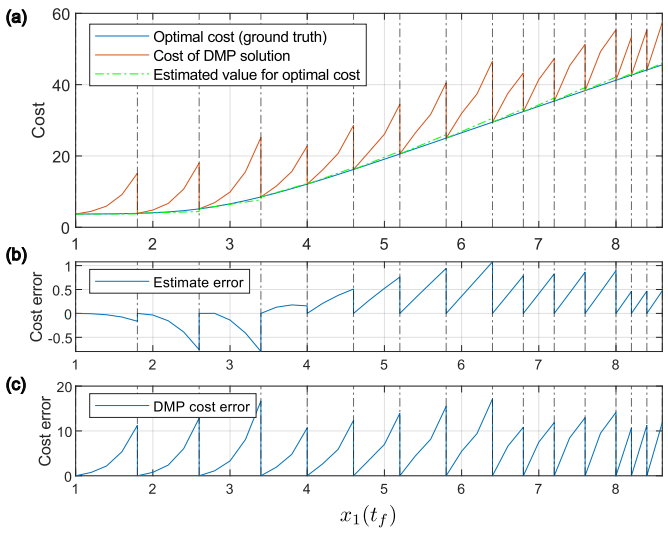

Fig. 2. (a) Application of sampling algorithm to example system (18). The dashed vertical lines indicate the terminal points wherein a new DMP should be used, while the points between the lines are out-of-sample learned trajectories. (b) Error between the estimated optimal cost and the real value of the optimal cost. (c) Difference between the cost of the DMP trajectory and the real optimal cost.

Figure 2 shows the efficacy of the approximation method for the optimal value function using (15). The maximum absolute error between the optimal cost (computed using a numerical optimal control solver only for the purpose of benchmark) and the cost estimated by (15) is given by 1.07.

To clarify the difference between the proposed sampling strategy and the naive uniform strategy, Figure 3 shows the respective grids that are obtained by each strategy. It can be seen that the resulting grid $\hat{\mathbb{P}}$ is non-uniform, reflecting the high sensitivity of the cost when x_f is within the subset $\{(x_1, 5) : x_1 \in [7, 8]\} \subset \mathbb{P}$. On the other hand, a uniform grid with length given by the minimal distance (of 0.2) between the sampling points of the uniform grid would require at least 39 samples to cover $\hat{\mathbb{P}}$, more than the double of the samples obtained using the proposed sampling strategy.

5. CONCLUSION

In this paper, we augmented a DMP-based framework for learning optimal control with a method to estimate the optimal value near the learned optimal trajectories, and used it to derive a sampling scheme that could be used for reducing storage requirements in applications where this could be an issue, such as in embedded systems.

The results in this paper could be extended in many directions, such as how to generalize this framework for dealing with optimal control problems with other varying parameters beside the terminal state. This would be interesting, for instance, for selecting the optimal configuration of a reconfigurable robot to execute a given task or in optimal braking problems, such as the one presented in (Hamad et al., 2023).

REFERENCES

Bansal, S. and Tomlin, C.J. (2021). DeepReach: a deep learning approach to high-dimensional reachability. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*.

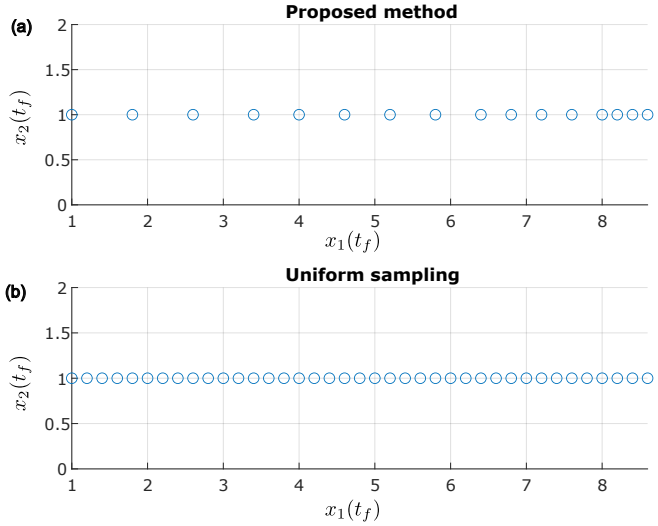

Fig. 3. (a) Grid obtained by proposed Algorithm 1. It uses only 15 samples. (b) Uniform grid using the minimal distance between sampling points of grid obtained by Algorithm 1. It uses 39 samples.

Bengio, Y., Lodi, A., and Prouvost, A. (2021). Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 290(2), 405–421.

Cheng, T., Lewis, F., and Abu-Khalaf, M. (2007). Fixed-final-time-constrained optimal control of nonlinear systems using neural network HJB approach. *IEEE Transactions on Neural Networks*, 18(6), 1725–1737.

Clever, D., Harant, M., Mombaur, K., Naveau, M., Stasse, O., and Endres, D. (2017). COCoMoPL: A novel approach for humanoid walking generation combining optimal control, movement primitives and learning and its transfer to the real robot HRP-2. *IEEE Robotics and Automation Letters*, 2(2), 977–984.

Diehl, M., Bock, H., Diedam, H., and Wieber, P.B. (2006). Fast direct multiple shooting algorithms for optimal robot control. In *Lecture Notes in Control and Information Sciences*, 65–93. Springer Berlin Heidelberg.

Dinev, T., Mastalli, C., Ivan, V., Tonneau, S., and Vijayakumar, S. (2022). Differentiable optimal control via differential dynamic programming.

E., W., Han, J., and Long, J. (2022). Empowering optimal control with machine learning: a perspective from model predictive control.

Güler, O. (2010). *Foundations of optimization*, volume 258. Springer Science & Business Media.

Haddadin, S., Weitschat, R., Huber, F., Özparpucu, M.C., Mansfeld, N., and Albu-Schäffer, A. (2013). Optimal control for viscoelastic robots and its generalization in real-time. In M. Inaba and P. Corke (eds.), *Robotics Research - The 16th International Symposium ISRR, 16-19 December 2013, Singapore*, volume 114 of *Springer Tracts in Advanced Robotics*, 131–148. Springer.

Hamad, M., Gutierrez-Moreno, J., Kussaba, H.T.M., Mansfeld, N., Abdolshah, S., Swikir, A., and Haddadin, S. (2023). Fast braking maneuvers for real-time robot control with stopping trajectory prediction. In *22nd World Congress of the International Federation of Automatic Control*.

Harzer, J., Schutter, J.D., and Diehl, M. (2022). Efficient numerical optimal control for highly oscillatory systems. *IEEE Control Systems Letters*, 6, 2719–2724.

Ijspeert, A.J., Nakanishi, J., Hoffmann, H., Pastor, P., and Schaal, S. (2013). Dynamical movement primitives: learning attractor models for motor behaviors. *Neural Computation*, 25(2), 328–373.

Isidori, A. and Astolfi, A. (1992). Disturbance attenuation and H_∞-control via measurement feedback in nonlinear systems. *IEEE Transactions on Automatic Control*, 37(9), 1283–1293.

Kamien, M.I. and Schwartz, N.L. (2012). *Dynamic optimization: the calculus of variations and optimal control in economics and management*. Dover, Mineola, New York.

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

LaValle, S.M. (2006). *Planning algorithms*. Cambridge University Press.

Liberzon, D. (2011). *Calculus of variations and optimal control theory: a concise introduction*. Princeton university press.

Nocedal, J., Wächter, A., and Waltz, R.A. (2009). Adaptive barrier update strategies for nonlinear interior methods. *SIAM Journal on Optimization*, 19(4), 1674–1693.

Perko, L. (2001). *Differential Equations and Dynamical Systems*. Springer New York.

Pohlodek, J., Morabito, B., Schlauch, C., Zometa, P., and Findeisen, R. (2022). Flexible development and evaluation of machine-learning-supported optimal control and estimation methods via HILO-MPC.

Sahlodin, A. and Barton, P. (2017). Efficient control discretization based on turnpike theory for dynamic optimization. *Processes*, 5(4), 85.

Saveriano, M., Abu-Dakka, F.J., Kramberger, A., and Peternel, L. (2021). Dynamic movement primitives in robotics: A tutorial survey. *arXiv preprint arXiv:2102.03861*.

Tomic, T., Maier, M., and Haddadin, S. (2014). Learning quadrotor maneuvers from optimal control and generalizing in real-time. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*.

van der Schaft, A. (1992). L_2-gain analysis of nonlinear systems and nonlinear state-feedback H_∞ control. *IEEE Transactions on Automatic Control*, 37(6), 770–784.

Weitschat, R., Haddadin, S., Huber, F., and Albu-Schauffer, A. (2013). Dynamic optimality in real-time: a learning framework for near-optimal robot motions. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*.

APPENDIX

Proof of Proposition 2

Let $x(t) := \Phi_D(x_f \mid x_f)(t)$ and $x'(t) := \Phi_D(x'_f \mid x_f)(t)$. Then the following equations are valid:

$$-\tau^2\ddot{x}(t) + \kappa(x_f - x(t)) - D\tau\dot{x}(t) = F(s(t)),$$
$$-\tau^2\ddot{x}'(t) + \kappa(x'_f - x'(t)) - D\tau\dot{x}'(t) = F(s(t)).$$

Defining $\hat{x}(t) = x(t) - x'(t)$ and subtracting the second equation from the first one yields

$$-\tau^2\ddot{\hat{x}}(t) - \kappa\hat{x}(t) - D\tau\dot{\hat{x}}(t) = -\kappa(x_f - x'_f),$$

a second-order linear equation with a constant forcing term which solution (when $\kappa = D^2/4$) is given by

$$\hat{x}(t) = \left(e^{-\frac{Dt}{2\tau}}\left(-\frac{Dt}{2\tau} - 1\right) + 1\right)(x_f - x'_f).$$