A Differential Dynamic Programming-based Approach for Balancing Energy and Time Optimality in Motion Planning

Yunshen Huang, Wenbo He, and Shen Zeng

***Abstract*— Optimal motion planning that simultaneously considers energy and time efficiency is crucial for a wide range of applications from industrial manufacturing to autonomous vehicle navigation. This paper introduces a novel perspective on discrete-time systems which recognizes the time interval size as an additional aspect of the control variable that is conventionally sought to be determined. By incorporating ideas from Dynamic Differential Programming (DDP), our method, called BO-DDP (Balancing energy and time Optimality via DDP), enables an adjustable trade-off between energy and time optimality. DDP leverages quadratic approximation of dynamics and cost function, ensuring accurate information capture and a high convergence rate. To address the challenge of high computational complexity in obtaining related derivatives, we introduce a Taylor series-based numerical scheme for simultaneous forward integration and differentiation. Extensive simulation experiments on two scenarios, including autonomous car navigation and quadcopter flight, demonstrate the practicality and effectiveness of our algorithm.**

I. Introduction

Optimal motion planning plays a critical role in various fields, including robotics, autonomous systems, and aerospace engineering, to name a few. It is concerned with the task of steering a dynamical system from a source to a destination while achieving a specific optimality criterion. Over the decades, various techniques have been developed to solve optimal control problems. These include indirect methods, which transform the problem into a boundary value challenge, capitalizing, for instance, on Pontryagin's Maximum Principle [1]. Conversely, direct methods, like direct collocation [2] and shooting methods [3], reformulate the problem into nonlinear programs. Modern numerical techniques and strategies for nonlinear systems have also been explored, see [4]–[6]. On the other hand, to reach the target with minimum time, the controller tends to utilize all available resources within the given constraints, which is referred to as the bang-bang principle [7]. Time-optimal problems have been addressed through approaches such as reformulating to convex optimization [8], considering minimal time steps while satisfying dynamic and target constraints [9], and leveraging Pontryagin's Maximum Principle for analytical expressions of time-optimal trajectories [10]. However, these methods are limited to linear or second-order nonlinear systems, thereby restricting their applicability.

Differential Dynamic Programming (DDP) [11] is a well-established and widely-applied approach to addressing nonlinear trajectory optimization. It iteratively refines control policies based on dynamic programming principles. The recursive nature is commonly shared with other optimal motion planning techniques, see [12] [13]. DDP and its variants have achieved success in various applications, showcasing their effectiveness and versatility, see [14] [15].

Traditionally, motion planning has focused on optimizing a single objective, whether it is maximizing energy efficiency or minimizing time to reach a target. However, in many real-world scenarios, a sweet spot between these two optimal criteria is a crucial consideration. For example, battery-powered systems require striking a balance between minimizing energy usage and achieving the shortest possible duration. Surprisingly, to the best of our knowledge, no existing algorithm addresses the challenge of combined energy and time optimality for general nonlinear systems.

In this paper, we introduce a novel approach to address motion planning problems in discrete-time nonlinear dynamical systems, allowing for adjustable priority between energy and time efficiency. Unlike the conventional assumption of fixed and equal time intervals, we propose considering the time interval size as an additional control aspect. By incorporating the interval size alongside the input signal, the system is simultaneously actuated, enabling us to optimize both energy and time efficiency. Having done so, given an appropriate cost function, the standard DDP algorithm can be directly applied to the augmented system to address motion planning problems with combined energy and time optimality. To tackle the computational complexity arising from the quadratic approximations in DDP, we introduce a Taylor series-based numerical method. This method efficiently computes the forward integration of Ordinary Differential Equations (ODEs) and their associated linear and quadratic approximations. Importantly, the derivatives required for optimization are computed as by-products during the integration process, reducing computational overhead.

This paper is organized as follows. Section II briefly reviews DDP. Then Section III develops and discusses our method, which is followed by a numerical approach on efficient integration and differentiation in Section IV. The applicability of the advocated technique is presented in Section V. Finally, the paper is concluded in Section VI.

II. Preliminaries

This section first provides a short review of the standard DDP framework, which is widely adapted to find an optimal control sequence for motion planning. Readers are referred to [11] for a detailed description. We additionally use this

Authors are with Department of Electrical and System Engineering, Washington University, St. Louis, MO 63130, USA. Email: {yunshen.huang,wenbo.he,s.zeng}@wustl.edu. This work was supported by the NSF grant CMMI-1933976.

section to formally define the mathematical notations used throughout this paper.

A. Optimal Motion Problem Formulation

Consider a discretized-time dynamical system

$$\mathbf{x}_{k+1} = \mathbf{F}(\mathbf{x}_k, \mathbf{u}_k), \tag{1}$$

where $\mathbf{x}_k \in \mathbb{R}^n, \mathbf{u}_k \in \mathbb{R}^m$ are the state and input of the system at the time step k, respectively, and the flow $\mathbf{F}$ governs the state transition based on the current state and input. By steering the system (1) with a sequence of inputs $\mathbf{U} \coloneqq \{\mathbf{u}_0, \cdots, \mathbf{u}_{N-1}\}$, the corresponding state trajectory can be found as $\mathbf{X} \coloneqq \{\mathbf{x}_0, \cdots, \mathbf{x}_N\}$. The total cost associated with $\mathbf{X}$ and $\mathbf{U}$ can be further defined as

$$J(\mathbf{X}, \mathbf{U}) = \sum_{k=0}^{N-1} l(\mathbf{x}_k, \mathbf{u}_k) + l_f(\mathbf{x}_N), \tag{2}$$

where $l(\mathbf{x}, \mathbf{u}) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ is known as the running cost and $l_f : \mathbb{R}^n \to \mathbb{R}$ is called the final cost.

We aim to find a control input $\mathbf{U}$ that drives the system (1) from a predefined start state $\mathbf{x}_0$ to the target $\mathbf{x}_{\text{target}}$ in N time steps, minimizing the total cost (2). One can apply dynamic programming to break down the minimization over the entire sequence of inputs to a sequence of minimizations over a single control, which is achieved backward in time. To facilitate this approach, we introduce the optimal value function at time step i to represent the optimal cost-to-go starting from a particular state $\mathbf{x}$

$$V_i(\mathbf{x}) = \min_{\mathbf{U}} \sum_{j=i}^{N-1} l(\mathbf{x}_j, \mathbf{u}_j) + l_f(\mathbf{x}_N).$$

Hereby, the associated Bellman equation is expressed as

$$V_i(\mathbf{x}) = \min_{\mathbf{u}} l(\mathbf{x}_i, \mathbf{u}_i) + V_{i+1}(\mathbf{F}(\mathbf{x}_i, \mathbf{u}_i)), \tag{3}$$

with the boundary condition $V_N(\mathbf{x}_N) = l_f(\mathbf{x}_N)$. For conciseness, in the subsequent sections, we adopt the notation V^+ for V_{i+1} and drop the time index subscript when it does not cause ambiguity.

B. Differential Dynamic Programming

The DDP algorithm tackles the aforementioned optimal control problem through consecutive backward and forward passes. In the backward pass, DDP employs quadratic Taylor expansion to approximate the value function along the nominal state-action trajectory $(\mathbf{X}, \mathbf{U})$. Subsequently, the forward pass updates the nominal trajectory by incorporating the locally optimal feedback law derived from the approximated value function. This iterative process continues until the desired level of convergence is achieved.

1) Backward Pass: Exciting (1) with a control input starting from a given state, we can examine the value of the cost-to-go function by defining an action-value function $Q : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$. Following the convention, we define perturbed Q around nominal $(\mathbf{x}, \mathbf{u})$ at an arbitrary step

$$Q(\delta\mathbf{x}, \delta\mathbf{u}) = l(\mathbf{x} + \delta\mathbf{x}, \mathbf{u} + \delta\mathbf{u}) + V^+(\mathbf{F}(\mathbf{x} + \delta\mathbf{x}, \mathbf{u} + \delta\mathbf{u})), \tag{4}$$

which is a discrete-time analogue of the Hamiltonian. We further approximate Q using the second-order Taylor expansion around $(\mathbf{0}, \mathbf{0})$, and arrive at

$$\begin{aligned} Q(\delta\mathbf{x}, \delta\mathbf{u}) = Q(\mathbf{0}, \mathbf{0}) &+ \begin{bmatrix} Q_{\mathbf{x}} & Q_{\mathbf{u}} \end{bmatrix} \begin{bmatrix} \delta\mathbf{x} \\ \delta\mathbf{u} \end{bmatrix} \\ &+ \frac{1}{2} \begin{bmatrix} \delta\mathbf{x} \\ \delta\mathbf{u} \end{bmatrix}^\top \begin{bmatrix} Q_{\mathbf{xx}} & Q_{\mathbf{ux}} \\ Q_{\mathbf{xu}} & Q_{\mathbf{uu}} \end{bmatrix} \begin{bmatrix} \delta\mathbf{x} \\ \delta\mathbf{u} \end{bmatrix}, \end{aligned} \tag{5}$$

where

$$\begin{aligned} Q(\mathbf{0}, \mathbf{0}) &= l(\mathbf{x}, \mathbf{u}) + V^+(\mathbf{x}), \\ Q_{\mathbf{x}} &= l_{\mathbf{x}} + \mathbf{F}_{\mathbf{x}}^\top V_{\mathbf{x}}^+, \\ Q_{\mathbf{u}} &= l_{\mathbf{u}} + \mathbf{F}_{\mathbf{u}}^\top V_{\mathbf{x}}^+, \\ Q_{\mathbf{xx}} &= l_{\mathbf{xx}} + \mathbf{F}_{\mathbf{x}}^\top V_{\mathbf{xx}}^+ \mathbf{F}_{\mathbf{x}} + V_{\mathbf{x}}^+ \mathbf{F}_{\mathbf{xx}}, \\ Q_{\mathbf{ux}} &= l_{\mathbf{ux}} + \mathbf{F}_{\mathbf{u}}^\top V_{\mathbf{xx}}^+ \mathbf{F}_{\mathbf{x}} + V_{\mathbf{x}}^+ \mathbf{F}_{\mathbf{ux}}, \\ Q_{\mathbf{uu}} &= l_{\mathbf{uu}} + \mathbf{F}_{\mathbf{u}}^\top V_{\mathbf{xx}}^+ \mathbf{F}_{\mathbf{u}} + V_{\mathbf{x}}^+ \mathbf{F}_{\mathbf{uu}}, \end{aligned}$$

which can be found by applying the chain rule on (4). Having done so, the Bellman equation (3) can be given by optimizing its quadratic approximation (5) function over control deviation $\delta\mathbf{u}$. The solution is expressed as a locally-feedback law as

$$\delta\mathbf{u} = \min_{\delta\mathbf{u}} Q(\delta\mathbf{x}, \delta\mathbf{u}) = \mathbf{K}\delta\mathbf{x} + \mathbf{k}, \tag{6}$$

where $\mathbf{K} = -Q_{\mathbf{uu}}^{-1} Q_{\mathbf{ux}}$ and $\mathbf{k} = -Q_{\mathbf{uu}}^{-1} Q_{\mathbf{u}}$ representing the feedback gain matrix and feed-forward term, respectively. By substituting the feedback law into the Q function (5), the value function (3) can be evaluated in a quadratic form. With simplification, we arrive at

$$V(\mathbf{x} + \delta\mathbf{x}) = \frac{1}{2}\delta\mathbf{x}^\top V_{\mathbf{xx}} \delta\mathbf{x} + V_{\mathbf{x}} \delta\mathbf{x} + \Delta V,$$

where

$$\begin{aligned} V_{\mathbf{xx}} &= Q_{\mathbf{xx}} + \mathbf{K}^\top Q_{\mathbf{uu}} \mathbf{K} + Q_{\mathbf{ux}}^\top \mathbf{K} + \mathbf{K}^\top Q_{\mathbf{uu}}, \\ V_{\mathbf{x}} &= Q_{\mathbf{x}} + \mathbf{K}^\top Q_{\mathbf{uu}} \mathbf{k} + Q_{\mathbf{ux}}^\top \mathbf{k} + \mathbf{K}^\top Q_{\mathbf{u}}. \end{aligned} \tag{7}$$

are the Hessian matrix and gradient evaluated at the nominal state $\mathbf{x}$, respectively. Note that the drifting term ΔV is irrelevant to the algorithm and thus ignored. Given the system (1) and a nominal state-action pair $(\mathbf{X}, \mathbf{U})$, the backward pass is processed by initializing the value function with the terminal cost and its derivatives as $V_N = l_f(\mathbf{x}_N)$, then recursively solves (6) and (7) all the way back to the beginning of the trajectory.

2) Forward Pass: The forward pass simply updates $(\mathbf{X}, \mathbf{U})$ in the new iteration by propagating the input derived from the feedback law (6). By starting from the given initial state, i.e. $\mathbf{x}_0^{\text{new}} = \mathbf{x}_0$, the forward pass is expressed as

$$\begin{aligned} \mathbf{u}_k^{\text{new}} &= \mathbf{u}_k + \mathbf{K}_k(\mathbf{x}_k^{\text{new}} - \mathbf{x}_k) + \mathbf{k}_k \\ \mathbf{x}_{k+1}^{\text{new}} &= \mathbf{F}(\mathbf{x}_k^{\text{new}}, \mathbf{u}_k^{\text{new}}). \end{aligned}$$

III. Balancing Energy and Time Optimality via DDP

In this section, we present an unconventional perspective on the analysis of discrete-time flow, enabling the utilization of the standard DDP method for solving motion planning problems in nonlinear dynamical systems with balanced energy and time optimality.

A. Proposed Approach

Discrete-time systems are practically advantageous for physical platform implementation. Conventionally, when dealing with such systems, the entire time horizon is evenly divided into a fixed number of time steps for the sake of computational convenience and notation simplicity. As a result, the explicit representation of the time interval is often omitted, as observed in the commonly used expression (1).

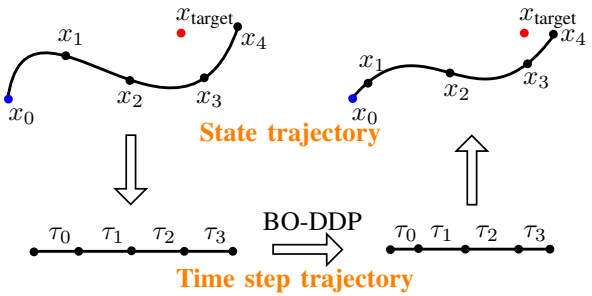


Fig. 1: Schematic diagram of the intuition behind our approach that varies the time interval size to achieve optimality

By contrast, we emphasize the significance of the time interval size, as it plays a crucial role in determining the state transition. Hereby, we slightly abuse the notation and rewrite the discrete-time system (1) as follows:

$$\mathbf{x}_{k+1} = \mathbf{F}(\mathbf{x}_k, \mathbf{u}_k, \tau_k), \tag{8}$$

where $\tau_k \in \mathbb{R}^+$ is the interval size at time step k. The key points are illustrated in Figure 1. By adopting this idea, we can formulate the motion planning problem with adjustable priority between energy and time optimality as the following optimization problem

$$\begin{aligned} \underset{\mathbf{U},\mathcal{T}}{\text{minimize}} \quad & \sum_{k=0}^{N-1} w_e \mathbf{u}_k^\top \mathbf{u}_k \tau_k + w_t \tau_k + w_f \|\mathbf{x}_N - \mathbf{x}_{\text{target}}\|^2 \\ \text{subject to} \quad & \mathbf{x}_{k+1} = \mathbf{F}(\mathbf{x}_k, \mathbf{u}_k, \tau_k), \\ & \mathbf{x}_0 = \mathbf{x}_{\text{start}}, \\ & \underline{\mathbf{u}} \le \mathbf{u}_k \le \overline{\mathbf{u}}, \\ & 0 \le \tau_k^{\text{n}} + \underline{\delta\tau} \le \tau_k \le \tau_k^{\text{n}} + \overline{\delta\tau} \le \overline{\tau}, \end{aligned} \tag{9}$$

where $\mathcal{T} = \{\tau_0, \cdots, \tau_{N-1}\}$ represents the time step trajectory, w_e, w_t, w_f determine the weights associated with the energy and time consumption, and target steering progress, respectively, τ_k^{n} denotes the nominal value of τ_k from the previous iteration, and $\underline{\cdot}, \overline{\cdot}$ are boxed boundary. The last constraint regulates the allowable variation of τ for each iteration which will be discussed in detail in Section III-B. This optimization problem is well-suited for the DDP framework, because the weighted summation of consumed energy and time can be defined as the running cost, and the steering progress as the final cost.

Furthermore, we augment the control variable by incorporating it with the time interval through stacking, i.e. $\hat{\mathbf{u}}_k = \begin{bmatrix} \mathbf{u}_k & \tau_k \end{bmatrix} \in \mathbb{R}^{m+1}$. Having done so, we not only simplify the notation but, more importantly, recognize the time window size τ as an additional dimension of the control variable. By doing so, the augmented system (8) is naturally compatible with the DDP algorithm, allowing for seamless integration into the optimization framework.

Remark: This work focuses on emphasizing the perspective of treating the time interval as an additional control aspect. Therefore, for (9) we simply enforce boxed constraints on $\mathbf{U}$ and $\mathcal{T}$, which can be easily solved by, e.g. logarithmic barrier method and projected Newton method [16]. For more sophisticated constraints, alternative approaches, e.g. [17] [18], can be explored.

Warm Start: We have observed that in DDP, simultaneously finding optimal control and time interval sequences while satisfying the terminal requirement can be challenging compared to finding only the feasible control. To address this challenge, we utilize DDP to first obtain a feasible control sequence that guides the system towards the target. This is achieved by considering a modified value function

$$J_{\text{ws}}(\mathbf{X}, \mathbf{U}) = \sum_{k=0}^{N-1} w_{e1} \mathbf{u}_k^\top \mathbf{u}_k + w_{f1} \|\mathbf{x}_N - \mathbf{x}_{\text{target}}\|^2, \tag{10}$$

where w_{e1} and w_{f1} are weight coefficients. By incorporating this modified value function, we can warm-start DDP to reach the combined optimality with a feasible initialization.

B. Further Discussions

1) High Nonlinearity Regarding Time Interval: We have found that the augmented dynamics (8) exhibit higher nonlinearity with respect to τ compared to $\mathbf{u}$. Figure 2 summarizes the magnitude of the Taylor expansion coefficients of (8) up to the second order of $\mathbf{u}$ and τ. Specifically, we examine the ℓ^2-norm of $\mathbf{F}_{\mathbf{u}}$, $\mathbf{F}_\tau$, $\mathbf{F}_{\mathbf{uu}}$, and $\mathbf{F}_{\tau\tau}$, by applying 40 randomly initialized inputs sequence to a cart pendulum model [19] with fixed $\tau_k \equiv 50\,ms$ and $N = 100$. Note that the input of the cart pendulum is a scalar force acting horizontally on the system. Therefore, the ℓ^2-norm can be safely applied to the associated second derivatives.

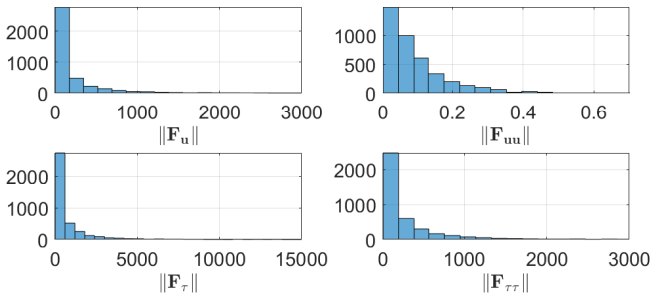


Fig. 2: Histogram of ℓ^2-norm of each derivative term. $\|\mathbf{F}_{\tau\tau}\|$ is seen to have much larger value compared to $\|\mathbf{F}_{\mathbf{uu}}\|$.

Given the higher nonlinearity observed in terms of τ in our method, it becomes crucial to provide a well-initialized $\mathbf{U}$ to enhance the performance. A warm-started DDP is particularly advantageous in this regard, as it improves subsequent iterations based on the results from the previous one.

2) Updating Time Interval: Being aware of the high sensitivity of τ, we should limit its variation for each iteration such that it is small enough to ensure the valid approximation in (5). Therefore, we enforce an element-wise trust region on τ as shown in the last constraint of (9). In addition, to avoid numerical issues during the state propagation, we examine the entire $\mathcal{T}$ after each forward pass, and then rule out τ_k that is too close to 0, where we consider the following rule

$$\tau_k = 0, \qquad \text{if } \tau_k \leq \underline{\tau}. \tag{11}$$

Our proposed BO-DDP is summarized in Algorithm 1.

Algorithm 1 Optimal Motion Planning via BO-DDP

Require: $x_{\text{start}}, x_{\text{target}}$, initialized $U_0, \mathcal{T}_0$, weight coefficients, constrained boundary

1) Start from $\mathbf{U}_0$, apply DDP to obtain a warm-starting control $\mathbf{U}_{\text{w}}$ by minimizing (10).
2) Start from $\mathbf{U}_{\text{w}}, \mathcal{T}_0$, apply DDP on optimization (9).
3) During Step 2, examine and update $\mathcal{T}$ by applying rule (11) after each forward pass.

IV. Efficient Integration and Differentiation

The quadratic terms in DDP are crucial for accurately depicting dynamics and cost function. However, obtaining second-order approximations is computationally expensive. Consequently, optimization techniques like iLQR [12] often omit the Hessians of $\mathbf{F}$. However, as discussed in Section III-B, the rich information conveyed by $\mathbf{F}_{\tau\tau}$ highlights the necessity of Hessians in achieving time-optimal results. To address this dilemma, we introduce a fundamental yet novel approach for the numerical solution of the ODE by examining the Taylor series of its solution. This process allows us to obtain Jacobians and Hessians as by-products.

A. Numerical Integrator

We start with a continuous-time autonomous system

$$\dot{\xi}(t) = F(\xi(t)), \quad \xi \in \mathbb{R}^n. \tag{12}$$

Then we expand ξ around a particular time point t

$$\xi(t+h) = \xi(t) + h\dot{\xi}(t) + \frac{h^2}{2}\ddot{\xi}(t) + \frac{h^3}{3!}\dddot{\xi}(t) + \dots$$

to some desired order $p \in \mathbb{N}$, for which a larger value gives a more accurate approximation of $\xi(t+h)$. Now, we examine each derivative by starting from the first order, i.e. $\dot{\xi} = F(\xi(t))$. As for the second order, we have

$$\ddot{\xi}(t) = \left.\frac{d}{dt}\dot{\xi}\right|_t = J(F)|_{\xi(t)} \left.\frac{d}{dt}\xi\right|_t = J(F)|_{\xi(t)}\, F|_{\xi(t)}\,.$$

If we continue to apply the chain rule to obtain the third and any higher-order derivatives, the computational complexity will be sharply increased, as these procedures involve the composition of different multilinear forms.

To confront this dilemma, adopting a slightly relaxed notation, let's define a vector field

$$\Psi_1(\xi) = \dot{\xi} = F(\xi).$$

It can be immediately followed by

$$\Psi_2(\xi) = \ddot{\xi} = \frac{d}{dt}\dot{\xi} = \frac{d}{dt}\Psi_1(\xi) = J(\Psi_1)F(\xi),$$

which can be easily recognized as another vector field just like Ψ_1. Moreover, we can keep on repeating this definition indefinitely to any Ψ_p, and will find they are all vector fields. By carefully inspecting their patterns, we conclude the following functional recursion

$$\Psi_{k+1} = J(\Psi_k)F, \qquad \Psi_0 = \text{Id}, \tag{13}$$

where $\text{Id} : \xi \mapsto \xi$ denotes the identity operator. In doing so, instead of working with composited functions, any higher-order derivative can be easily obtained via sequentially computing the Jacobian of a vector field by following (13). Hereby, the numerical integration for ODE (12) to any desired order p is achieved

$$\xi(t+h) = \sum_{q=0}^{\infty} \frac{h^q}{q!}\Psi_q(\xi(t)) \approx \sum_{q=0}^{p} \frac{h^q}{q!}\Psi_q(\xi(t)). \tag{14}$$

B. Calculation of Derivatives

Another great feature of the proposed integrator is that the procedure for computing the Taylor expansion components also directly carries out the derivatives of the flow as a by-product. To show this, we first define a flow ϕ of (12) over the time step h and state ξ, i.e. $\phi : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$.

1) First Derivative: Given (14), we can express the approximation

$$\phi(h, \xi) \approx \Psi_0(\xi) + h\Psi_1(\xi) + \dots + \frac{h^p}{p!}\Psi_p(\xi),$$

whose first derivative over h can be immediately carried out

$$\frac{\partial}{\partial h}\phi \approx \sum_{q=1}^{p} \frac{h^{q-1}}{(q-1)!}\Psi_q(\xi).$$

While the derivative over ξ can be expressed as

$$\frac{\partial}{\partial \xi}\phi \approx J(\Psi_0) + hJ(\Psi_1) + \dots + \frac{h^p}{p!}J(\Psi_p).$$

By recognizing the terms

$$J(\Psi_0) = I, \;\; J(\Psi_1) = J(F), \;\; J(\Psi_2) = J(J(F)F), \;\; \dots,$$

we can see that these components have already been computed as intermediate products during the integration (14).

2) Second Derivative: Based on the results found above, computing the second derivatives is straightforward

$$\frac{\partial^2}{\partial h^2}\phi \approx \sum_{q=2}^{p} \frac{h^{q-2}}{(q-2)!}\Psi_q(\xi),$$
$$\frac{\partial^2}{\partial \xi^2}\phi \approx \sum_{q=0}^{p} \frac{h^{q}}{q!}J(J(\Psi_q)),$$
$$\frac{\partial^2}{\partial h \partial \xi}\phi \approx \sum_{q=1}^{p} \frac{h^{q-1}}{(q-1)!}J(\Psi_q).$$

Note that the term $J(J(\Psi_q))$ is viewed as a tensor involving additional yet mild computations. On the other hand, the remaining components are by-products of (14) as well.

Remark: Despite our study being on autonomous system (12), our approach can be readily extended to controlled systems. To see this, one can simply stack the state with a zero-order-hold control signal, and then truncate the dynamics related to the control, as the input remains unchanged throughout the entire time step. The subsequent analysis and computation remain identical to those presented above.

C. Comparison of Computational Time

To demonstrate the efficiency of the proposed numerical approach, we compare its computation time against the finite difference approach and, a widely-used Python library for differentiation, JAX [20]. The task is to compute state trajectory, as well as the first and second derivatives at each time step by propagating a control sequence on a full-scale nonlinear quadcopter system, whose model will be shown in Section V-B. For the other two differential approaches, the fourth-order Runge-Kutta method is applied for forward integration. The test is run on a desktop with a 3.8 GHz Intel Core i7 processor and 64 GB of RAM. Table I summarizes the results of 10 independent simulations with fixed $\tau_k \equiv 50\,ms$, $N = 200$, and approximation order $p = 4$. Our method demonstrates a significant advantage over the other two methods in terms of computational time. This enhanced efficiency greatly improves the practicality of BO-DDP for handling complex systems.

TABLE I: Computational Time of Three Methods

	Our Method	RK4 + JAX	RK4 + Finite Difference
Mean [sec]	2.81	45.26	126.89
Var	6.72×10^{-3}	0.58	3.60

V. Numerical Simulations

We demonstrate the effectiveness of the proposed BO-DDP by applying it to two different platforms: a car model that operates on a planer space, and a more complex and highly nonlinear model of the quadcopter in 3-D space. For each example, we will vary the weighting pair (w_e, w_t) for different optimality criteria, i.e. one has more budget on control effort than that on time consumption or vice versa.

A. Car in Planer Space

The corresponding vehicle dynamics are described by the following system of nonlinear differential equations

$$\frac{d}{dt}\begin{pmatrix} x \\ y \\ \phi \\ v \end{pmatrix} = \begin{pmatrix} v\sin\phi \\ v\cos\phi \\ u^\phi v \\ u^v \end{pmatrix},$$

where ϕ represents the facing angle, v stands for the forward moving velocity, the control variables $\begin{bmatrix} u^\phi & u^v \end{bmatrix}$ are to change steering angle and forward velocity, respectively.

For this test, we command a motionless car drive from the origin by facing north to stop at the spot $(2, 3)$ by facing east, i.e. the initial and target states are set to be $\mathbf{x}_{\text{start}} = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}$ and $\mathbf{x}_{\text{target}} = \begin{bmatrix} 2 & 3 & \frac{\pi}{2} & 0 \end{bmatrix}$. With this simple system, we directly apply a cold-started BO-DDP. More specifically, the optimization problem (9) is considered, where we set the terminal cost weight $w_f = 5000$, controls are bounded by $\|u^\phi\| \leq \frac{\pi}{2}$ and $\|u^v\| \leq 2$, maximum interval length is set to be $\overline{\tau} = 100\,ms$, and largest allowable variation of τ between each iteration is limited as $-\underline{\delta\tau} = \overline{\delta\tau} = 5\,ms$. To start with, we set $\mathbf{U}_0 = \mathbf{0}$, and form $\mathcal{T}_0$ by evenly distributing 5 second into 100 time intervals, i.e. $\tau_k = 50\,ms, N = 100$.

With the other settings unchanged, we gradually raise the value of $\frac{w_t}{w_e}$ in order to investigate the capability of BO-DDP in addressing problems with increasingly limited time budgets. The results of five experiments are summarized in Table II, revealing a clear trend of increasing energy usage and decreasing time consumption as the weights ratio rises. In situations where a car needs to be parked as quickly as possible, a large w_t dominating w_e should be picked.

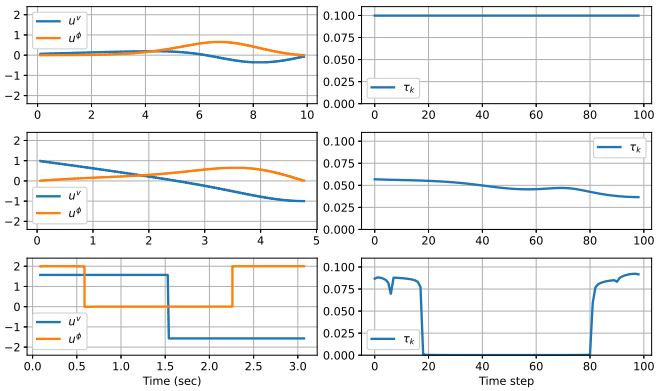
TABLE II: Used Energy and Time with Different Weights

w_t/w_e	0	0.33	1	3	∞
Energy Used	1.45	1.80	2.56	4.50	13.13
Time Used (sec)	9.88	6.04	4.78	3.68	3.07

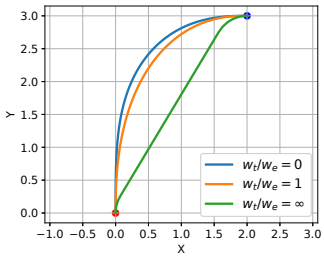
Furthermore, the results of the three most representative experiments are depicted in Figure 3a. When operating under a stringent energy budget, $\mathbf{u}_k$ strives to minimize its value at each step of k, while simultaneously maximizing τ_k. Conversely, when prioritizing time efficiency, $\mathbf{u}_k$ optimally utilizes all available resources to steer the system towards the target, resembling the characteristics of a bang-bang control strategy. Additionally, Figure 3b illustrates the corresponding trajectories. With increasingly limited timeframes, the car tends to take a more direct path toward the target instead of following a rounder curve.

B. Quadcopter

A more challenging model is considered here to showcase the effectiveness of the proposed method on motion planning problems. The dynamic model employed for the quadcopter system comprises a set of nonlinear ordinary differential equations (ODEs) with 12 states and 4 control



(a) From top to bottom, each row represents the resulting action-horizon pair with increasing $\frac{w_t}{w_e}$. The left column shows $\mathbf{U}$, while distributions of $\mathcal{T}$ over the horizon are shown on the right.



(b) Driven by different $\mathbf{U}$, the corresponding trajectories of the car start from the red point to the target (blue dot).

Fig. 3: Results of three experiments with $\frac{w_t}{w_e} \in \{0,\ 1,\ \infty\}$ by deploying the BO-DDP on the 2D car model.

inputs representing the rotational speed of the four motors in rotations per minute (rpm). It can be expressed as:

$$\frac{d}{dt}x = f_d(x) + \sum_{i=1}^{4} f_i(x)u_i^2,$$

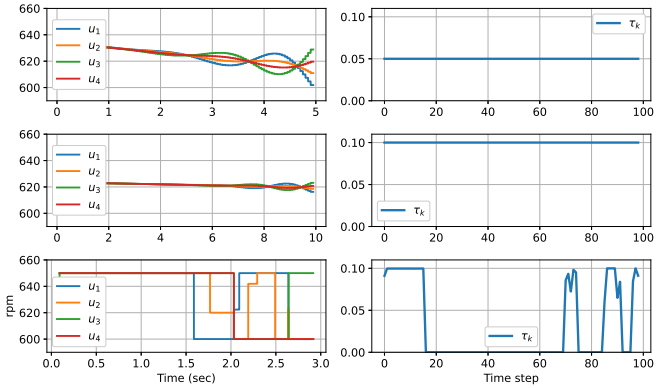
where f_d and f_i are nonlinear differentiable functions. For a detailed dynamics structure, readers are referred to [21].

For this particular test, our objective is to navigate a standing-still quadcopter from the origin to a hovering position at coordinates $(1, 3, 2)$. Given the intricacy of the system, we employed the warm-starting strategy outlined in (10) to initially synthesize a feasible $\mathbf{U}_0$. The hyperparameters used in this test were identical to those in Section V-A, with the exception of the following adjustments: we enforced a restriction on the rpm values for each motor, limiting them to the range $600 \leq u \leq 650$, and set $-\underline{\delta\tau} = \overline{\delta\tau} = 2ms$. To exclude the impact of gravity, we customize the energy consumed at step k as $(\mathbf{u}_k - \mathbf{u}_k^{\text{hov}})^\top(\mathbf{u}_k - \mathbf{u}_k^{\text{hov}})\tau_k$, where $\mathbf{u}_k^{\text{hov}}$ equals the rpm enabling the quadcopter to hover.

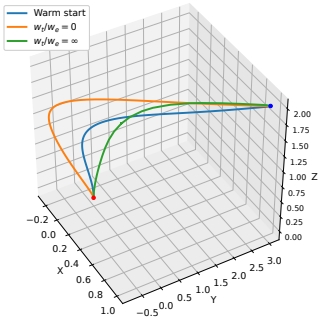
Once again, we explore the role of $\frac{w_t}{w_e}$ on the consumption of energy and time, which is summarized in Table III. As expected, the results reveal that a narrower time budget leads to quicker attainment of the target, at the expense of increased energy consumption. We additionally show the resulting sequences of $\mathbf{U}, \mathcal{T}, \mathbf{X}$ in Figure 4, where one can observe that the quadcopter picks a more aggressive control strategy to reach the target quicker.

TABLE III: Used Energy and Time with Different Weights

w_t/w_e	0	9	199	999	∞
Energy Used	106	147	402	1381	8310
Time Used (sec)	9.90	9.70	6.60	4.52	2.92



(a) The resulting action-horizon pair from the warm start is shown on the first row. The following two rows demonstrate the ones by setting $\frac{w_t}{w_e} = 0, \infty$, respectively. The left column shows $\mathbf{U}$s, while distributions of $\mathcal{T}$s over the horizon are shown on the right.



(b) Driven by different $\mathbf{U}$, the corresponding trajectories of the quadcopter start from the red point to the target (blue dot).

Fig. 4: Results of the warm start and two other experiments with $\frac{w_t}{w_e} \in \{0,\ \infty\}$ by deploying the BO-DDP on the 3D quadcopter model.

VI. Conclusion

In this study, we introduced a novel perspective on discrete-time systems, wherein the time interval size is considered as an additional dimension of the control variable. By incorporating Differential Dynamic Programming (DDP) and a numerical method for integration and differentiation, we developed the BO-DDP method for Balancing energy and time Optimality in motion planning. Various experiments showcased the practicality and effectiveness of our approach.

However, we also identified two areas for future improvement. Firstly, in dealing with complex systems, we currently provide a warm-starting strategy only for the control sequence. A warm-started time horizon would further enhance the overall performance of our method. Secondly, the resulting optimal time horizon tends to be unevenly distributed over the time steps, which may pose challenges for certain digital platforms operating at fixed frequencies.

REFERENCES

[1] I. M. Ross, *A Primer on Pontryagin's Principle in Optimal Control*, 2nd ed. Collegiate Publishers, 3 2015.

[2] C. R. Hargraves and S. W. Paris, "Direct trajectory optimization using nonlinear programming and collocation," *Journal of guidance, control, and dynamics*, vol. 10, no. 4, pp. 338–342, 1987.

[3] M. R. Osborne, "On shooting methods for boundary value problems," *Journal of mathematical analysis and applications*, vol. 27, no. 2, pp. 417–433, 1969.

[4] Y. Li, T. Yang, and S. Tong, "Adaptive neural networks finite-time optimal control for a class of nonlinear systems," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 11, pp. 4451–4460, 2019.

[5] W. He, Y. Huang, J. Wang, and S. Zeng, "Homotopy method for optimal motion planning with homotopy class constraints," *IEEE Control Systems Letters*, vol. 7, pp. 1045–1050, 2022.

[6] M. Giftthaler, M. Neunert, M. StÃ¤uble, J. Buchli, and M. Diehl, "A family of iterative gauss-newton shooting methods for nonlinear optimal control," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1–9.

[7] J. P. LaSalle *et al.*, "The time optimal control problem," *Contributions to the theory of nonlinear oscillations*, vol. 5, pp. 1–24, 2016.

[8] D. Verscheure, B. Demeulenaere, J. Swevers, J. De Schutter, and M. Diehl, "Time-optimal path tracking for robots: A convex optimization approach," *IEEE Transactions on Automatic Control*, vol. 54, no. 10, pp. 2318–2327, 2009.

[9] S. Al Homsi, A. Sherikov, D. Dimitrov, and P.-B. Wieber, "A hierarchical approach to minimum-time control of industrial robots," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 2368–2374.

[10] A. Taitler, I. Ioslovich, E. Karpas, and P.-O. Gutman, "Minimum time optimal control of second order system with quadratic drag and state constraints," in *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 523–528.

[11] D. Mayne, "A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems," *International Journal of Control*, vol. 3, no. 1, pp. 85–95, 1966.

[12] W. Li and E. Todorov, "Iterative linear quadratic regulator design for nonlinear biological movement systems," in *Proc. 1st International Conference on Informatics in Control, Automation and Robotics*, 2004, pp. 222–229.

[13] W. He, Y. Huang, and S. Zeng, "Motion planning with homotopy class constraints via the auxiliary energy reduction technique," in *2022 American Control Conference (ACC)*. IEEE, 2022, pp. 4933–4938.

[14] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 4906–4913.

[15] H. Li and P. M. Wensing, "Hybrid systems differential dynamic programming for whole-body motion planning of legged robots," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5448–5455, 2020.

[16] Y. Tassa, N. Mansard, and E. Todorov, "Control-limited differential dynamic programming," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 1168–1175.

[17] Z. Xie, C. K. Liu, and K. Hauser, "Differential dynamic programming with nonlinear constraints," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 695–702.

[18] H. Almubarak, K. Stachowicz, N. Sadegh, and E. A. Theodorou, "Safety embedded differential dynamic programming using discrete barrier states," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2755–2762, 2022.

[19] F. Mazenc and S. Bowong, "Tracking trajectories of the cart-pendulum system," *Automatica*, vol. 39, no. 4, pp. 677–684, 2003.

[20] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "JAX: composable transformations of Python+NumPy programs," 2018. [Online]. Available: http://github.com/google/jax

[21] F. Sabatino, "Quadrotor control: modeling, nonlinear control design, and simulation," (Masters Degree Project, KTH Royal Institute of Technology), 2015.