JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS
Vol. 41, No. 12, December 2018

Min-Max Differential Dynamic Programming: Continuous and Discrete Time Formulations

Wei Sun,* Yunpeng Pan,† Jaein Lim,‡ Evangelos A. Theodorou,§ and Panagiotis Tsiotras¶
Georgia Institute of Technology, Atlanta, Georgia 30332-0150

DOI: 10.2514/1.G003516

In this work, the first min-max Game-Theoretic Differential Dynamic Programming (GT-DDP) algorithm in continuous time is derived. A set of backward differential equations for the value function is provided, along with its first- and second-order derivatives without assuming proximity of the initial nominal controls to the optimal controls. The corresponding update laws for both the minimizing and maximizing controls are also derived. For comparison, the derivation of the GT-DDP algorithm in discrete time is presented in order to better elucidate the differences between the continuous and discrete time formulations. The effect of the game-theoretic formulation in the feed-forward and feedback parts of the optimal control policies is analyzed, and the discrete and continuous time GT-DDP algorithms are compared through numerical examples. Experimental results using a quadrotor demonstrate the superiority of GT-DDP in handling model uncertainties and external disturbances over the standard DDP algorithm.

I. Introduction

DIFFERENTIAL game-theoretic or min-max formulations are important extensions of optimal control having direct connections to robust and H_∞ nonlinear control theory [1,2], as well as risk-sensitive control [3,4]. Despite the plethora of previous works in this area, min-max algorithms for trajectory optimization have been scarce. We are familiar only with the work of [5], where a min-max algorithm was derived and was applied to a humanoid robotic control problem. The approach in [5] is based on differential dynamic programming (DDP) [6,7]. DDP uses dynamic programming ideas and attempts to find a locally optimal control policy through the iterative improvement of a nominal control and state trajectory. One attractive feature of DDP is its second-order convergence [8,9]. The performance of DDP has been compared with other local methods and was found to be, in general, superior to them [10,11]. DDP has also been applied to various realistic problems with notable success, such as robot locomotion [12], helicopter acrobatic maneuvers [13], and biological nervous systems [10]. Although the initial derivation of DDP was in continuous time, most of the work on trajectory optimization, including the min-max DDP formulation in [5], addresses either discrete time nonlinear systems or discretized versions of systems that are originally expressed in continuous time.

Many variations of the basic DDP algorithm have been derived and applied to deterministic and stochastic systems in robotics, autonomous systems, and computational neuroscience. In particular, in [14] a discrete-time DDP algorithm was derived for nonlinear stochastic systems with state and control multiplicative noise, and was applied to biomechanical models. The resulting algorithm, known as iterative Linear Quadratic Gaussian (iLQG) control, relies on a first-order expansion of the dynamics. In [15], second-order expansions of a stochastic dynamical system with state and control multiplicative noise were considered. The resulting algorithm, known as Stochastic Differential Dynamic Programming (SDDP), is a generalization of iLQG. The DDP algorithm has been applied in a receding horizon manner to account for complex dynamics and alleviates the curse of dimensionality [16,17]. In [18], random sampling techniques are proposed to improve the scalability of DDP. An infinite-horizon version of the discrete time DDP is derived in [19,20], where DDP is used for deterministic nonlinear systems with control limits and is subsequently applied to control of a humanoid robot in simulation. DDP with control and state constraints is discussed in [21]. Finally, DDP is combined with machine learning methods for systems with learned dynamics [22–24].

Although there has been a lot of work on several versions of DDP algorithms and their application to engineering systems, most of the work in the literature thus far has been on discrete time formulations and for cases where there are no disturbances. Following up on this existing amount of work, in this paper we present the derivation of a min-max DDP (or Game-Theoretic Differential Dynamic Programming [GT-DDP]) algorithm in continuous time that solves a two-player zero-sum differential game. The basic outline of the proposed GT-DDP algorithm was first reported in [25]. In the current paper all the missing details and assumptions used in [25] for the derivation of GT-DDP have been added and elaborated upon. Furthermore, and in order to show the benefits of working directly in the continuous-time domain, we compare the developed min-max DDP algorithm with a similar, more standard, discrete time formulation. In particular, we provide a set of backward differential equations for the continuous-time case and difference equations for the discrete-time case, respectively. From these, we subsequently derive the optimal policies for the two players/controllers. We compare the continuous and discrete time formulations in terms of their convergence and numerical efficiency and, in particular, we investigate the effect of the min-max formulation in the feed-forward and feedback parts of the optimal control policies.

With respect to the initial treatment of DDP in [7], our analysis and derivation of the GT-DDP follows [26] and avoids a restrictive assumption of the initial derivation of DDP. Specifically, the fundamental assumption in the derivation of continuous-time DDP in [7] is that the nominal control $\bar{\boldsymbol{u}}$ is close to the optimal control $\boldsymbol{u}^*$. This assumption was also discussed in [27]. This assumption allows the expansion of the terms in the Hamilton–Jacobi–Bellman (HJB) partial differential equation (PDE) around $\boldsymbol{u}^*$ instead of $\bar{\boldsymbol{u}}$ and results in the cancelation of terms that depend on $\mathcal{H}_u = 0$, where $\mathcal{H}_u$ stands for the partial derivative of the Hamiltonian with respect to the control input. GT-DDP, on the other hand, does not rely on this assumption, and therefore the quadratic expansions of the terms in the HJB PDE are computed around the nominal controls $\bar{\boldsymbol{u}}$, $\bar{\boldsymbol{v}}$ and not around the

Received 16 December 2017; revision received 24 May 2018; accepted for publication 5 August 2018; published online 31 October 2018.

*Ph.D. Candidate, School of Aerospace Engineering; currently Postdoctoral Fellow, Department of Aeronautics and Astronautics, The University of Washington, Seattle, WA; wsun42@gatech.edu.

†Ph.D. Candidate, School of Aerospace Engineering; currently Senior Scientist, JD.com American Technologies Corporation, Mountain View, CA; ypan37@gatech.edu.

‡Graduate Student, School of Aerospace Engineering; jaeinlim126@gatech.edu.

§Professor, School of Aerospace Engineering; evangelos.theodorou@ae.gatech.edu.

¶Dean's Professor, School of Aerospace Engineering; tsiotras@gatech.edu. Fellow AIAA.

optimal controls $\boldsymbol{u}^*, \boldsymbol{v}^*$. In this case, the neglected terms $\mathcal{H}_u$ and $\mathcal{H}_v$ in the standard DDP problem formulation may not necessarily be equal to zero. More details on this issue are given in Sec. II.E.

The rest of the paper is organized as follows. In Sec. II the game-theoretic problem is formulated and the backward differential equations in continuous time are derived. In Sec. III the derivation of the game-theoretic DDP in discrete time is presented, and a comparison between the two formulations is given in Sec. IV. Simulation and experimental results are shown in Sec. V. Finally, Sec. VI provides a summary of the results and suggests some possible future research directions.

II. Game Theoretic Differential Dynamic Programming in Continuous Time

A. Notation

The notation used throughout this paper is quite standard, but is repeated here for the sake of completeness. Specifically, $\mathbb{R}$ denotes the set of real numbers, $\mathbb{R}_+$ denotes the set of all positive real numbers, $\mathbb{R}^n$ denotes the set of $n \times 1$ real column vectors, and $\mathbb{R}^{n\times m}$ denotes the set of $n \times m$ real matrices. For single-variable functions, $\mathcal{C}^1(\mathbb{R})$ denotes the set of functions having domain $\mathbb{R}$ that are continuously differentiable and $\mathcal{C}^2(\mathbb{R})$ denotes the set of functions whose second-order derivative exists and is continuous with domain $\mathbb{R}$. Given a multivariable function of two arguments $g: \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$, we write $g \in \mathcal{C}^{k,\ell}(\mathbb{R}^n \times \mathbb{R}^m)$ if g has continuous derivatives up to order k with respect to the first argument and continuous derivatives up to order ℓ with respect to the second argument. We also write $(\cdot)^T$ for the transpose operator and $(\cdot)^{-1}$ for the inverse operator of a matrix.

Given a function $\psi: \mathbb{R}^n \times [0, t_f] \to \mathbb{R}$, we let $\psi_x(\boldsymbol{x}, t) \coloneqq [(\partial\psi(\boldsymbol{x}, t)/\partial \boldsymbol{x}_1), \ldots, (\partial\psi(\boldsymbol{x}, t)/\partial \boldsymbol{x}_n)]^T$ denote the partial derivative of $\boldsymbol{\psi}$ with respect to $\boldsymbol{x}$, and

$$\psi_{xx}(\boldsymbol{x}, t) \coloneqq \begin{bmatrix} \frac{\partial^2\psi(\boldsymbol{x},t)}{\partial^2 \boldsymbol{x}_1}, & \cdots, & \frac{\partial^2\psi(\boldsymbol{x},t)}{\partial \boldsymbol{x}_1 \boldsymbol{x}_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2\psi(\boldsymbol{x},t)}{\partial \boldsymbol{x}_1 \partial \boldsymbol{x}_n}, & \cdots, & \frac{\partial^2\psi(\boldsymbol{x},t)}{\partial^2 \boldsymbol{x}_n} \end{bmatrix}$$

to denote the Hessian of ψ with respect to $\boldsymbol{x}$. Given the vector-valued function $\omega: \mathbb{R}^n \times \mathbb{R}_+ \to \mathbb{R}^n$ defined by $\omega(\boldsymbol{x}, t) = [\omega_1(\boldsymbol{x}, t), \ldots, \omega_n(\boldsymbol{x}, t)]^T$, the notation

$$\omega_x(\boldsymbol{x}, t) \coloneqq \begin{bmatrix} \frac{\partial\omega_1(\boldsymbol{x},t)}{\partial \boldsymbol{x}_1}, & \cdots, & \frac{\partial\omega_1(\boldsymbol{x},t)}{\partial \boldsymbol{x}_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial\omega_n(\boldsymbol{x},t)}{\partial \boldsymbol{x}_1}, & \cdots, & \frac{\partial\omega_n(\boldsymbol{x},t)}{\partial \boldsymbol{x}_n} \end{bmatrix}$$

denotes the Jacobian matrix of ω with respect to $\boldsymbol{x}$.

B. Problem Formulation

We consider a min-max differential game problem where the dynamics of the system can be described by

$$\frac{\mathrm{d}\boldsymbol{x}(t)}{\mathrm{d}t} = F(\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{v}(t), t), \qquad \boldsymbol{x}(t_0) = \boldsymbol{x}_0 \tag{1}$$

where $\boldsymbol{x}(t) \in \mathbb{R}^n$ is the state of the dynamic system at $t \in [t_0, t_f]$, and $\boldsymbol{u}(t) \in U_1 \subset \mathbb{R}^{m_u}$ and $\boldsymbol{v}(t) \in U_2 \subset \mathbb{R}^{m_v}$ denote the conflicting controls, where U_1 and U_2 are open and convex constraint sets of the controls $\boldsymbol{u}$ and $\boldsymbol{v}$, respectively. We seek nonanticipative feedback strategies for both players; that is, we wish to find maps $\gamma_u: [t_0, t_f] \times \mathbb{R}^n \to U_1$ and $\gamma_v: [t_0, t_f] \times \mathbb{R}^n \to U_2$ such that one of the players minimizes the cost function

$$J(\gamma_u, \gamma_v) = \phi(\boldsymbol{x}(t_f), t_f) + \int_{t_0}^{t_f} \mathcal{L}(\boldsymbol{x}(t), \gamma_u(t, \boldsymbol{x}(t)), \gamma_v(t, \boldsymbol{x}(t)), t)\,\mathrm{d}t \tag{2}$$

while the other player maximizes Eq. (2). In Eq. (2) $\gamma_u(t, \boldsymbol{x}(t)) = \boldsymbol{u}(t)$ and $\gamma_v(t, \boldsymbol{x}(t)) = \boldsymbol{v}(t)$ are the feedback strategies of the players, t_f is a fixed terminal time of the game, $\mathcal{L}: \mathbb{R}^n \times \mathbb{R}^{m_u} \times \mathbb{R}^{m_v} \times [t_0, t_f] \to \mathbb{R}_+$ denotes the running cost, and $\phi: \mathbb{R}^n \times [t_0, t_f] \to \mathbb{R}_+$ denotes the terminal cost.

Standard regularity assumptions on the functions F, $\mathcal{L}$, and ϕ will be assumed throughout the paper to ensure existence and uniqueness of solutions of the differential Eq. (1). Accordingly, it will be assumed that $\gamma_u(t, \boldsymbol{x})$ and $\gamma_v(t, \boldsymbol{x})$ are piecewise continuous functions in t and Lipschitz continuous functions in $\boldsymbol{x}$.

We assume that the player manipulating $\boldsymbol{u}$ attempts to minimize the cost function, whereas the player manipulating $\boldsymbol{v}$ aims to maximize the cost function. The function describing the minimax value of the cost function at t_0 and $\boldsymbol{x}_0$ is then given by

$$V(\boldsymbol{x}_0, t_0) = \min_{\gamma_u} \max_{\gamma_v} \left\{ \phi(\boldsymbol{x}(t_f), t_f) + \int_{t_0}^{t_f} \mathcal{L}(\boldsymbol{x}(t), \gamma_u(t, \boldsymbol{x}(t)), \gamma_v(t, \boldsymbol{x}(t)), t)\,\mathrm{d}t \right\} \tag{3}$$

which is known as the value function.

For the derivation of the continuous time GT-DDP we will need additional conditions on the smoothness of the dynamics and the cost function, as well as some additional assumptions on the differential game (1) and (3). In particular, we will assume that

(A1) The dynamics $F \in \mathcal{C}^{1,1,1,1}(\mathbb{R}^n \times \mathbb{R}^{m_u} \times \mathbb{R}^{m_v} \times \mathbb{R}_+)$.

(A2) The running cost $\mathcal{L} \in \mathcal{C}^{2,2,2,1}(\mathbb{R}^n \times \mathbb{R}^{m_u} \times \mathbb{R}^{m_v} \times \mathbb{R}_+)$.

(A3) For any fixed t_f, the terminal cost $\phi \in \mathcal{C}^2(\mathbb{R}^n)$.

(A4) The value $V \in \mathcal{C}^{2,1}(\mathbb{R}^n \times \mathbb{R}_+)$.

(A5) Both the minimizer and maximizer have perfect knowledge of the dynamics of the system given by Eq. (1), the constraint sets U_1 and U_2, the cost function, and the current state $\boldsymbol{x}$.

(A6) Only (nonanticipative) state feedback strategies for both players are considered.

(A7) The Isaacs condition [28]

$$\min_{u\in U_1} \max_{v\in U_2} \{\mathcal{L}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}, t) + V_x(\boldsymbol{x}, t)^T F(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}, t)\} = \max_{v\in U_2} \min_{u\in U_1} \{\mathcal{L}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}, t) + V_x(\boldsymbol{x}, t)^T F(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}, t)\} \tag{4}$$

holds for all $\boldsymbol{x}$, $\boldsymbol{u}$, $\boldsymbol{v}$ and $t \geq 0$.

Assumptions (A1) through (A4) ensure that all derivatives in the subsequent derivations exist, whereas (A5) is a standard assumption for games with complete information [28]. Assumption (A6) comes from the fact that as long as a saddle point in feedback strategies exists, it is not necessary to consider other classes of strategies (see Theorem 6.9 in [29]). The Isaacs's condition in assumption (A7) ensures that the min and max operators can be interchanged without affecting the outcome and thus ensures the existence of the value of the game. Note that the Isaacs's condition holds when the controls are separable in both dynamics and cost [28], which is the case for many problems in practice.

The derivation of the min-max DDP is similar to the standard DDP case, in the sense that it consists of two main steps. In the first step the optimal policy update is derived. This policy update is a function of the zero, first, and second order of terms of the value function expansion along the nominal trajectory. The second step of the derivation results in the equations of the backward propagation of the value function along the nominal trajectory. Next, we start our analysis with the derivation of the optimal policy update.

C. Optimal Policy Update

In continuous time, it is known that the value function satisfies the Hamilton–Jacobi–Bellman–Isaacs (HJBI) PDE. Specifically, we have for the value of the game in (3)

$$-\frac{\partial V(\boldsymbol{x}, t)}{\partial t} = \min_{u} \max_{v} \{\mathcal{L}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}, t) + V_x(\boldsymbol{x}, t)^T F(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}, t)\} \tag{5}$$

with boundary condition

$$V(\boldsymbol{x}(t_f), t_f) = \phi(\boldsymbol{x}(t_f), t_f) \tag{6}$$

Given an initial/nominal trajectory of the state and control histories $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}, \bar{\boldsymbol{v}})$, and letting $\delta\boldsymbol{x} = \boldsymbol{x} - \bar{\boldsymbol{x}}, \delta\boldsymbol{u} = \boldsymbol{u} - \bar{\boldsymbol{u}}, \delta\boldsymbol{v} = \boldsymbol{v} - \bar{\boldsymbol{v}}$, one obtains

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = F(\bar{\boldsymbol{x}} + \delta\boldsymbol{x}, \bar{\boldsymbol{u}} + \delta\boldsymbol{u}, \bar{\boldsymbol{v}} + \delta\boldsymbol{v}, t) \tag{7}$$

which leads to the linearized dynamics around $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}, \bar{\boldsymbol{v}})$ as follows:

$$\frac{\mathrm{d}\delta\boldsymbol{x}}{\mathrm{d}t} = \bar{F}_x\delta\boldsymbol{x} + \bar{F}_u\delta\boldsymbol{u} + \bar{F}_v\delta\boldsymbol{v} \tag{8}$$

where $\bar{F}_x$, $\bar{F}_u$, and $\bar{F}_v$ stand for $F_x(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}, \bar{\boldsymbol{v}}, t)$, $F_u(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}, \bar{\boldsymbol{v}}, t)$, and $F_v(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}, \bar{\boldsymbol{v}}, t)$, respectively. Henceforth, the arguments for the functions V, F, and so on, will be omitted for brevity unless specified otherwise, and an overbar will denote that the corresponding quantity is evaluated along the nominal trajectory $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}, \bar{\boldsymbol{v}})$.

The main idea is to take expansions of the terms in both sides of Eq. (5) around the nominal state and control trajectories $(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}, \bar{\boldsymbol{v}})$ to derive the update laws for the minimizing control, maximizing control, and backward differential equations for the zeroth-, first-, and second-order approximation terms of the value function. After some mathematical manipulations, we obtain the expansion equation as follows. Please refer to Appendix A for a detailed derivation of this equation.

$$\begin{aligned}
&-\frac{\mathrm{d}\bar{V}}{\mathrm{d}t} - \delta\boldsymbol{x}^T\frac{\mathrm{d}\bar{V}_x}{\mathrm{d}t} - \frac{1}{2}\delta\boldsymbol{x}^T\frac{\mathrm{d}\bar{V}_{xx}}{\mathrm{d}t}\delta\boldsymbol{x} = \min_{\delta u}\max_{\delta v}\Bigg\{\bar{\mathcal{L}} + \bar{\mathcal{L}}_x^T\delta\boldsymbol{x} + \bar{\mathcal{L}}_u^T\delta\boldsymbol{u} + \bar{\mathcal{L}}_v^T\delta\boldsymbol{v} \\
&+ \frac{1}{2}\begin{bmatrix}\delta\boldsymbol{x}\\ \delta\boldsymbol{u}\\ \delta\boldsymbol{v}\end{bmatrix}^T\begin{bmatrix}\bar{\mathcal{L}}_{xx} & \bar{\mathcal{L}}_{xu} & \bar{\mathcal{L}}_{xv}\\ \bar{\mathcal{L}}_{ux} & \bar{\mathcal{L}}_{uu} & \bar{\mathcal{L}}_{uv}\\ \bar{\mathcal{L}}_{vx} & \bar{\mathcal{L}}_{vu} & \bar{\mathcal{L}}_{vv}\end{bmatrix}\begin{bmatrix}\delta\boldsymbol{x}\\ \delta\boldsymbol{u}\\ \delta\boldsymbol{v}\end{bmatrix} + \bar{V}_x^T\bar{F}_x\delta\boldsymbol{x} + \bar{V}_x^T\bar{F}_u\delta\boldsymbol{u} \\
&+ \bar{V}_x^T\bar{F}_v\delta\boldsymbol{v} + \delta\boldsymbol{x}^T\bar{V}_{xx}\bar{F}_x\delta\boldsymbol{x} + \delta\boldsymbol{x}^T\bar{V}_{xx}\bar{F}_u\delta\boldsymbol{u} + \delta\boldsymbol{x}^T\bar{V}_{xx}\bar{F}_v\delta\boldsymbol{v}\Bigg\} \\
&= \min_{\delta u}\max_{\delta v}\Bigg\{\bar{\mathcal{L}} + \delta\boldsymbol{x}^T\bar{Q}_x + \delta\boldsymbol{u}^T\bar{Q}_u + \delta\boldsymbol{v}^T\bar{Q}_v + \frac{1}{2}\delta\boldsymbol{x}^T\bar{Q}_{xx}\delta\boldsymbol{x} \\
&+ \frac{1}{2}\delta\boldsymbol{u}^T\bar{Q}_{uu}\delta\boldsymbol{u} + \frac{1}{2}\delta\boldsymbol{v}^T\bar{Q}_{vv}\delta\boldsymbol{v} + \delta\boldsymbol{u}^T\bar{Q}_{ux}\delta\boldsymbol{x} + \delta\boldsymbol{v}^T\bar{Q}_{vx}\delta\boldsymbol{x} + \delta\boldsymbol{u}^T\bar{Q}_{uv}\delta\boldsymbol{v}\Bigg\}
\end{aligned} \tag{9}$$

where

$$\bar{Q}_x = \bar{F}_x^T\bar{V}_x + \bar{\mathcal{L}}_x \tag{10a}$$

$$\bar{Q}_u = \bar{F}_u^T\bar{V}_x + \bar{\mathcal{L}}_u \tag{10b}$$

$$\bar{Q}_v = \bar{F}_v^T\bar{V}_x + \bar{\mathcal{L}}_v \tag{10c}$$

$$\bar{Q}_{xx} = \bar{\mathcal{L}}_{xx} + \bar{V}_{xx}\bar{F}_x + \bar{F}_x^T\bar{V}_{xx} \tag{10d}$$

$$\bar{Q}_{uu} = \bar{\mathcal{L}}_{uu} \tag{10e}$$

$$\bar{Q}_{vv} = \bar{\mathcal{L}}_{vv} \tag{10f}$$

$$\bar{Q}_{ux} = \bar{F}_u^T\bar{V}_{xx} + \bar{\mathcal{L}}_{ux} \tag{10g}$$

$$\bar{Q}_{vx} = \bar{F}_v^T\bar{V}_{xx} + \bar{\mathcal{L}}_{vx} \tag{10h}$$

$$\bar{Q}_{uv} = \bar{\mathcal{L}}_{uv} \tag{10i}$$

To find the optimal control $\delta\boldsymbol{u}^*$ and $\delta\boldsymbol{v}^*$, we compute the gradients of the expression inside the brackets in Eq. (9) with respect to $\delta\boldsymbol{u}$ and $\delta\boldsymbol{v}$, respectively, and make them equal to zero to obtain:

$$\delta\boldsymbol{u}^* = -\bar{Q}_{uu}^{-1}(\bar{Q}_{ux}\delta\boldsymbol{x} + \bar{Q}_{uv}\delta\boldsymbol{v}^* + \bar{Q}_u) \tag{11a}$$

$$\delta\boldsymbol{v}^* = -\bar{Q}_{vv}^{-1}(\bar{Q}_{vx}\delta\boldsymbol{x} + \bar{Q}_{vu}\delta\boldsymbol{u}^* + \bar{Q}_v) \tag{11b}$$

where we have used the fact that $\bar{Q}_{vu} = \bar{Q}_{uv}^T$. Solving Eq. (11) yields the final expressions for $\delta\boldsymbol{u}^*$ and $\delta\boldsymbol{v}^*$ as follows:

$$\delta\boldsymbol{u}^* = \boldsymbol{l}_u + \boldsymbol{K}_u\delta\boldsymbol{x} \quad \text{and} \quad \delta\boldsymbol{v}^* = \boldsymbol{l}_v + \boldsymbol{K}_v\delta\boldsymbol{x} \tag{12}$$

with the feed-forward gains $\boldsymbol{l}_v$, $\boldsymbol{l}_u$ and feedback gains $\boldsymbol{K}_v$, $\boldsymbol{K}_u$ defined as:

$$\boldsymbol{l}_u = -(\bar{Q}_{uu} - \bar{Q}_{uv}\bar{Q}_{vv}^{-1}\bar{Q}_{vu})^{-1}(\bar{Q}_u - \bar{Q}_{uv}\bar{Q}_{vv}^{-1}\bar{Q}_v) \tag{13}$$

$$\boldsymbol{l}_v = -(\bar{Q}_{vv} - \bar{Q}_{vu}\bar{Q}_{uu}^{-1}\bar{Q}_{uv})^{-1}(\bar{Q}_v - \bar{Q}_{vu}\bar{Q}_{uu}^{-1}\bar{Q}_u) \tag{14}$$

$$\boldsymbol{K}_u = -(\bar{Q}_{uu} - \bar{Q}_{uv}\bar{Q}_{vv}^{-1}\bar{Q}_{vu})^{-1}(\bar{Q}_{ux} - \bar{Q}_{uv}\bar{Q}_{vv}^{-1}\bar{Q}_{vx}) \tag{15}$$

$$\boldsymbol{K}_v = -(\bar{Q}_{vv} - \bar{Q}_{vu}\bar{Q}_{uu}^{-1}\bar{Q}_{uv})^{-1}(\bar{Q}_{vx} - \bar{Q}_{vu}\bar{Q}_{uu}^{-1}\bar{Q}_{ux}) \tag{16}$$

D. Backward Propagation of the Value Function

The next step is to substitute the optimal control (11a) and disturbance (maximizing control) (11b) to the HJBI Eq. (5) in order to find the update law of the value function and its first- and second-order partial derivatives. We present the result in the following proposition.

Proposition II.1: In the continuous time formulation, the value function and its first- and second-order partial derivatives with respect to $\boldsymbol{x}$ can be determined by the backward ordinary differential equations as follows:

$$-\frac{\mathrm{d}\bar{V}}{\mathrm{d}t} = \bar{\mathcal{L}} + \boldsymbol{l}_u^T\bar{Q}_u + \boldsymbol{l}_v^T\bar{Q}_v + \frac{1}{2}\boldsymbol{l}_u\bar{Q}_{uu}\boldsymbol{l}_u + \boldsymbol{l}_u^T\bar{Q}_{uv}\boldsymbol{l}_v + \frac{1}{2}\boldsymbol{l}_v^T\bar{Q}_{vv}\boldsymbol{l}_v \tag{17a}$$

$$\begin{aligned}
-\frac{\mathrm{d}\bar{V}_x}{\mathrm{d}t} &= \bar{Q}_x + \boldsymbol{K}_u^T\bar{Q}_u + \boldsymbol{K}_v^T\bar{Q}_v + \bar{Q}_{ux}^T\boldsymbol{l}_u + \bar{Q}_{vx}^T\boldsymbol{l}_v + \boldsymbol{K}_u^T\bar{Q}_{uu}\boldsymbol{l}_u \\
&+ \boldsymbol{K}_u^T\bar{Q}_{uv}\boldsymbol{l}_v + \boldsymbol{K}_v^T\bar{Q}_{vu}\boldsymbol{l}_u + \boldsymbol{K}_v^T\bar{Q}_{vv}\boldsymbol{l}_v
\end{aligned} \tag{17b}$$

$$\begin{aligned}
-\frac{\mathrm{d}\bar{V}_{xx}}{\mathrm{d}t} &= \boldsymbol{K}_u^T\bar{Q}_{ux} + \bar{Q}_{ux}^T\boldsymbol{K}_u + \boldsymbol{K}_v^T\bar{Q}_{vx} + \bar{Q}_{vx}^T\boldsymbol{K}_v + \boldsymbol{K}_v^T\bar{Q}_{vu}\boldsymbol{K}_u \\
&+ \boldsymbol{K}_u^T\bar{Q}_{uv}\boldsymbol{K}_v + \boldsymbol{K}_u^T\bar{Q}_{uu}\boldsymbol{K}_u + \boldsymbol{K}_v^T\bar{Q}_{vv}\boldsymbol{K}_v + \bar{Q}_{xx}
\end{aligned} \tag{17c}$$

under terminal conditions

$$\bar{V}(t_f) = \phi(\bar{\boldsymbol{x}}(t_f), t_f) \tag{18a}$$

$$\bar{V}_x(t_f) = \phi_x(\bar{\boldsymbol{x}}(t_f), t_f) \tag{18b}$$

$$\bar{V}_{xx}(t_f) = \phi_{xx}(\bar{\boldsymbol{x}}(t_f), t_f) \tag{18c}$$

Proof: See Appendix B.

In many engineering applications the cost function is designed such that $\mathcal{L}_{vu} = \mathcal{L}_{uv}^T = 0$. In this case, the differential equations for

the backward propagation of the value function along the nominal trajectory are simplified as follows:

$$-\frac{\mathrm{d}\bar{V}}{\mathrm{d}t} = \bar{\mathcal{L}} + \boldsymbol{l}_u^T\bar{Q}_u + \boldsymbol{l}_v^T\bar{Q}_v + \frac{1}{2}\boldsymbol{l}_u\bar{Q}_{uu}\boldsymbol{l}_u + \frac{1}{2}\boldsymbol{l}_v^T\bar{Q}_{vv}\boldsymbol{l}_v \tag{19a}$$

$$-\frac{\mathrm{d}\bar{V}_x}{\mathrm{d}t} = \bar{Q}_x + \boldsymbol{K}_u^T Q_u + \boldsymbol{K}_v^T\bar{Q}_v + \bar{Q}_{ux}^T\boldsymbol{l}_u + \bar{Q}_{vx}^T\boldsymbol{l}_v + \boldsymbol{K}_u^T\bar{Q}_{uu}\boldsymbol{l}_u + \boldsymbol{K}_v^T\bar{Q}_{vv}\boldsymbol{l}_v \tag{19b}$$

$$-\frac{\mathrm{d}\bar{V}_{xx}}{\mathrm{d}t} = \boldsymbol{K}_u^T\bar{Q}_{ux} + \bar{Q}_{ux}^T\boldsymbol{K}_u + \boldsymbol{K}_v^T\bar{Q}_{vx} + \bar{Q}_{vx}^T\boldsymbol{K}_v + \boldsymbol{K}_u^T\bar{Q}_{uu}\boldsymbol{K}_u + \boldsymbol{K}_v^T\bar{Q}_{vv}\boldsymbol{K}_v + \bar{Q}_{xx} \tag{19c}$$

Remark 1: It is worth pointing out that our approach requires that the value function be twice differentiable [see assumption (A4)]. In general, V may not be twice differentiable [30]. Indeed, many Hamilton–Jacobi equations have solutions only in a relaxed sense, namely, viscosity solutions [31], which are continuous but not necessarily differentiable everywhere. However, it is known that linear-quadratic problems, in which the system has linear dynamics and a quadratic cost function, have smooth value functions [32]. Beyond linear-quadratic problems, the same is true for nonlinear problems that have a controllable linearization at every point and their cost function can be locally approximated by a nondegenerate quadratic cost. In this case, the HJB equation reduces to a matrix Riccati equation of the type encountered in H_∞ problems [1]. The theory of solutions of such matrix Riccati equations has been investigated extensively in the literature. So, for such problems locally, the problem can be approximated by a twice differentiable value function. Each iteration in the GT-PPD algorithm performs a first-order approximation of the dynamics model and a second-order approximation of the cost function along a nominal trajectory, resulting in a linear-quadratic problem at each iteration, for which the value function is twice continuously differentiable.

E. Comparison to the Derivation of Original DDP

It is worth comparing the basic steps of GT-DDP with the steps of the minimax DDP extended from the original derivation of the continuous-time DDP in [7]. The major differences between the derivation of continuous-time DDP in [7] and our approach were mentioned briefly in Sec. I. In this subsection, we will elaborate more on the similarities and differences between the two derivations. First, we present an extension of the original DDP and explain how this can be applied to solve problems with conflicting controls.

To this end, let the Hamiltonian $\mathcal{H}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{v},V_x,t) = \mathcal{L}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{v},t) + V_{\boldsymbol{x}}^T F(\boldsymbol{x},\boldsymbol{u},\boldsymbol{v},t)$. The derivation of the original DDP starts with the HJB equation. Accordingly, the HJBI equation given in Eq. (5) is used for the case of conflicting controls. By letting $\boldsymbol{x} = \bar{\boldsymbol{x}} + \delta\boldsymbol{x}$, $\boldsymbol{u} = \bar{\boldsymbol{u}} + \delta\boldsymbol{u}$, $\boldsymbol{v} = \bar{\boldsymbol{v}} + \delta\boldsymbol{v}$, and after some mathematical manipulations, Eq. (5) is expanded up to second order to yield

$$\begin{aligned}-\frac{\partial\bar{V}}{\partial t} - \frac{\partial a}{\partial t} - \frac{\partial\bar{V}_x{}^T}{\partial t}\delta\boldsymbol{x} - \frac{1}{2}\delta\boldsymbol{x}^T\frac{\partial\bar{V}_{xx}}{\delta}\boldsymbol{x} &= \min_{\delta u}\max_{\delta v}\Big\{\mathcal{H}(\bar{\boldsymbol{x}}+\delta\boldsymbol{x},\bar{\boldsymbol{u}}+\delta\boldsymbol{u},\bar{\boldsymbol{v}}+\delta\boldsymbol{v},\bar{V}_x,t)\\ &+\left(\bar{V}_{xx}\delta\boldsymbol{x} + \frac{1}{2}\bar{V}_{\boldsymbol{xxx}}\delta\boldsymbol{x}\delta\boldsymbol{x}\right)^T F(\bar{\boldsymbol{x}}+\delta\boldsymbol{x},\bar{\boldsymbol{u}}+\delta\boldsymbol{u},\bar{\boldsymbol{v}}+\delta\boldsymbol{v},t)\Big\}\end{aligned} \tag{20}$$

where all the terms whose arguments are unspecified are evaluated at $(\bar{\boldsymbol{x}},\bar{\boldsymbol{u}},\bar{\boldsymbol{v}})$, and a is the difference between the cost obtained by evaluating it along $(\bar{\boldsymbol{x}},\boldsymbol{u}^\star,\boldsymbol{v}^\star)$ and the nominal cost $\bar{V}$ obtained at $(\bar{\boldsymbol{x}},\bar{\boldsymbol{u}},\bar{\boldsymbol{v}})$, where $\boldsymbol{u}^\star$ and $\boldsymbol{v}^\star$ are given by Eqs. (21) and (22). The term $\bar{V}_{xxx}\delta\boldsymbol{x}\delta\boldsymbol{x}$ denotes the vector $[\delta\boldsymbol{x}^T\bar{V}_{xxx}^{(1)}\delta\boldsymbol{x},\ \ldots,\ \delta\boldsymbol{x}^T\bar{V}_{xxx}^{(n)}\delta\boldsymbol{x}]^T$.

Instead of finding the optimal $\delta\boldsymbol{u}$ and $\delta\boldsymbol{v}$ with respect to $\bar{\boldsymbol{u}}$ and $\bar{\boldsymbol{v}}$, in the original DDP approach the intermediate optimal controls $\boldsymbol{u}^\star$ and $\boldsymbol{v}^\star$ are obtained (analytically or numerically) by solving the problems

$$\boldsymbol{u}^* = \arg\min_u \mathcal{H}(\bar{\boldsymbol{x}},\boldsymbol{u},\boldsymbol{v}^\star,\bar{V}_x,t) \tag{21}$$

$$\boldsymbol{v}^* = \arg\max_v \mathcal{H}(\bar{\boldsymbol{x}},\boldsymbol{u}^\star,\boldsymbol{v},\bar{V}_x,t) \tag{22}$$

Note that $\boldsymbol{u}^\star$ and $\boldsymbol{v}^\star$ are only intermediate optimal controls as they are evaluated along $\bar{\boldsymbol{x}}$ and they would eventually converge to the true optimal controls when $\bar{\boldsymbol{x}}$ converges to $\boldsymbol{x}^\star$. The corresponding backward propagation equations with respect to the value function and its first- and second-order partial derivatives are then found by equating the left-hand side of Eq. (20) with the second-order expansion of

$$\begin{aligned}\min_{\delta u}\max_{\delta v}\Big\{&\mathcal{H}(\bar{\boldsymbol{x}}+\delta\boldsymbol{x},\boldsymbol{u}^\star+\delta\boldsymbol{u},\boldsymbol{v}^\star+\delta\boldsymbol{v},\bar{V}_x,t)\\ &+\left(\bar{V}_{xx}\delta\boldsymbol{x} + \frac{1}{2}\bar{V}_{xxx}\delta\boldsymbol{x}\delta\boldsymbol{x}\right)^T F(\bar{\boldsymbol{x}}+\delta\boldsymbol{x},\boldsymbol{u}^\star+\delta\boldsymbol{u},\boldsymbol{v}^\star+\delta\boldsymbol{v},t)\Big\}\end{aligned} \tag{23}$$

Since $\boldsymbol{u}^\star$ minimizes the Hamiltonian while $\boldsymbol{v}^\star$ maximizes the Hamiltonian, the necessary conditions $\mathcal{H}_u(\bar{\boldsymbol{x}},\boldsymbol{u}^\star,\boldsymbol{v}^\star,\bar{V}_x,t) = 0$ and $\mathcal{H}_v(\bar{\boldsymbol{x}},\boldsymbol{u}^\star,\boldsymbol{v}^\star,\bar{V}_x,t) = 0$ are satisfied. After expansion of Eq. (23) and collection of similar terms, the backward differential equations that describe the propagation of the value function are presented as follows:

$$-\frac{\mathrm{d}a}{\mathrm{d}t} = \mathcal{H} - \mathcal{H}(\bar{\boldsymbol{x}},\bar{\boldsymbol{u}},\bar{\boldsymbol{v}},\bar{V}_x,t) \tag{24a}$$

$$-\frac{\mathrm{d}\bar{V}_x}{\mathrm{d}t} = \mathcal{H}_x + \beta^T\mathcal{H}_u^{=0} + \eta^T\mathcal{H}_v^{=0} + \bar{V}_{xx}(F - F(\bar{\boldsymbol{x}},\bar{\boldsymbol{u}},\bar{\boldsymbol{v}},t)) \tag{24b}$$

$$\begin{aligned}-\frac{\mathrm{d}\bar{V}_{xx}}{\mathrm{d}t} &= \mathcal{H}_{xx} + \mathcal{H}_{xu}\beta + \beta^T\mathcal{H}_{ux} + \mathcal{H}_{xv}\eta + \eta^T\mathcal{H}_{vx} + \beta^T\mathcal{H}_{uu}\beta\\ &+ \eta^T\mathcal{H}_{vv}\eta + \beta^T\mathcal{H}_{uv}\eta + \eta^T\mathcal{H}_{vu}\beta + F_x^T\bar{V}_{xx} + \bar{V}_{xx}F_x + \beta^T F_u^T\bar{V}_{xx}\\ &+ \bar{V}_{xx}F_u\beta + \eta^T F_v^T\bar{V}_{xx} + \bar{V}_{xx}F_v\eta + \frac{1}{2}(F - F(\bar{\boldsymbol{x}},\bar{\boldsymbol{u}},\bar{\boldsymbol{v}},t))^T\bar{V}_{xxx}\\ &+ \frac{1}{2}\bar{V}_{xxx}(F - F(\bar{\boldsymbol{x}},\bar{\boldsymbol{u}},\bar{\boldsymbol{v}},t))\end{aligned} \tag{24c}$$

where

$$\begin{aligned}\beta &= -(\mathcal{H}_{uu} - \mathcal{H}_{uv}\mathcal{H}_{vv}^{-1}\mathcal{H}_{vu})^{-1}(\mathcal{H}_{ux} + F_u^T\bar{V}_{xx} - \mathcal{H}_{uv}\mathcal{H}_{vv}^{-1}\\ &\quad\times(\mathcal{H}_{vx} + F_v^T\bar{V}_{xx})),\\ \eta &= -(\mathcal{H}_{vv} - \mathcal{H}_{vu}\mathcal{H}_{uu}^{-1}\mathcal{H}_{uv})^{-1}(\mathcal{H}_{vx} + F_v^T\bar{V}_{xx} - \mathcal{H}_{vu}\mathcal{H}_{uu}^{-1}\\ &\quad\times(\mathcal{H}_{ux} + F_u^T\bar{V}_{xx}))\end{aligned}$$

where all quantities are evaluated at $(\bar{\boldsymbol{x}},\boldsymbol{u}^\star,\boldsymbol{v}^\star)$ unless otherwise stated. The corresponding update laws are $\boldsymbol{u} = \boldsymbol{u}^\star + \delta\boldsymbol{u} = \boldsymbol{u}^\star + \beta\delta\boldsymbol{x}$ and $\boldsymbol{v} = \boldsymbol{v}^\star + \delta\boldsymbol{v} = \boldsymbol{v}^\star + \eta\delta\boldsymbol{x}$.

It is worth mentioning here that the step to obtain $\boldsymbol{u}^\star$ and $\boldsymbol{v}^\star$ from Eqs. (21) and (22) may be computationally expensive if a numerical approach is adopted, because one needs to perform this optimization at each instant during the propagation of the value function. In our method, $\delta\boldsymbol{u}^*$ and $\delta\boldsymbol{v}^*$ can be computed analytically following Eq. (11). Note that in Eq. (24) the terms $\beta^T\mathcal{H}_u$ and $\eta^T\mathcal{H}_v$ become zero, since $\mathcal{H}_u = 0$ and $\mathcal{H}_v = 0$ when evaluated with the optimal controls $\boldsymbol{u}^\star$ and $\boldsymbol{v}^\star$ from Eqs. (21) and (22). In contrast, we expand the terms in the HJBI equation around the nominal controls $\bar{\boldsymbol{u}}$, $\bar{\boldsymbol{v}}$ instead of $\boldsymbol{u}^\star$, $\boldsymbol{v}^\star$.

Another observation is that the term $\bar{V}_{xxx}$ shows up in Eq. (24c), which was not present in the backward differential equations (17).

This term is neglected in the original derivation of DDP in order to integrate Eq. (24). However, the terms $(1/2)(F(\bar{\boldsymbol{x}}, \boldsymbol{u}^\star, \boldsymbol{v}^\star, t) - F(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}, \bar{\boldsymbol{v}}, t))^T \bar{V}_{xxx}$ and $(1/2)\bar{V}_{xxx}(F(\bar{\boldsymbol{x}}, \boldsymbol{u}^\star, \boldsymbol{v}^\star, t) - F(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}, \bar{\boldsymbol{v}}, t))$ can be omitted only when $\bar{\boldsymbol{u}}$ is relatively close to $\boldsymbol{u}^\star$, and $\bar{\boldsymbol{v}}$ is relatively close to $\boldsymbol{v}^\star$.

III. Game-Theoretic Differential Dynamic Programming in Discrete Time

An alternative to the previous continuous-time formulation is to apply GT-DDP in a discrete setting. In this section, we derive the discrete-time game theoretic DDP algorithm and compare it with the discrete min-max DDP derived in [5]. In the discrete-time approach, the problem is first discretized along the given time interval $[t_0, t_f]$. Let the time discretization be $0 = t_0 < t_1 < \ldots < t_M = t_f$, and let $\boldsymbol{x}_k = \boldsymbol{x}(t_k)$, $\boldsymbol{u}_k = \boldsymbol{u}(t_k)$ and $\boldsymbol{v}_k = \boldsymbol{v}(t_k)$. Starting from $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + F(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{v}_k, t_k)\delta t$, where $\delta t = t_{k+1} - t_k$, the linearized dynamics model around $\bar{\boldsymbol{x}}_k$, $\bar{\boldsymbol{u}}_k$, $\bar{\boldsymbol{v}}_k$ in discrete time can be written as

$$\delta \boldsymbol{x}_{k+1} = \bar{\boldsymbol{A}}(t_k)\delta \boldsymbol{x}_k + \bar{\boldsymbol{B}}_u(t_k)\delta \boldsymbol{u}_k + \bar{\boldsymbol{B}}_v(t_k)\delta \boldsymbol{v}_k \tag{25}$$

where $\bar{\boldsymbol{A}}(t_k) = \boldsymbol{I} + F_x(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k, \bar{\boldsymbol{v}}_k, t_k)\delta t$, $\bar{\boldsymbol{B}}_u(t_k) = F_u(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k, \bar{\boldsymbol{v}}_k, t_k)\delta t$ and $\bar{\boldsymbol{B}}_v(t_k) = F_v(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k, \bar{\boldsymbol{v}}_k, t_k)\delta t$. Here $\delta \boldsymbol{x}_k = \boldsymbol{x}_k - \bar{\boldsymbol{x}}_k$, $\delta \boldsymbol{u}_k = \boldsymbol{u}_k - \bar{\boldsymbol{u}}_k$ and $\delta \boldsymbol{v}_k = \boldsymbol{v}_k - \bar{\boldsymbol{v}}_k$ are defined as the deviations from the nominal trajectory at time t_k. The state-dependent arguments in $\bar{\boldsymbol{A}}(t_k)$, $\bar{\boldsymbol{B}}_u(t_k)$, $\bar{\boldsymbol{B}}_v(t_k)$ are evaluated at $(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k, \bar{\boldsymbol{v}}_k)$, and are omitted for abbreviation. Henceforth, all the values at t_k are evaluated at $(\bar{\boldsymbol{x}}_k, \bar{\boldsymbol{u}}_k, \bar{\boldsymbol{v}}_k)$, unless otherwise specified. Similarly, values at t_{k+1} are evaluated at $(\bar{\boldsymbol{x}}_{k+1}, \bar{\boldsymbol{u}}_{k+1}, \bar{\boldsymbol{v}}_{k+1})$.

Similarly to the HJBI equation in the continuous-time case, our analysis for the discrete time case starts with the discrete HJBI equation with minimizing and maximizing controls $\boldsymbol{u}_k$ and $\boldsymbol{v}_k$, namely, $V(\boldsymbol{x}_k, t_k) = \min_{u_k} \max_{v_k} \{L(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{v}_k, t_k) + V(\boldsymbol{x}_{k+1}, t_{k+1})\}$, where $L(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{v}_k, t_k) = \mathcal{L}(\boldsymbol{x}(t_k), \boldsymbol{u}(t_k), \boldsymbol{v}(t_k), t_k)\delta t$. Let $\Theta(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{v}_k) = L(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{v}_k, t_k) + V(\boldsymbol{x}_{k+1}, t_{k+1})$ We construct a quadratic local model of the value function by expanding the function Θ up to second order as follows:

$$\begin{aligned}\Theta(\bar{\boldsymbol{x}}_k + \delta \boldsymbol{x}_k, \bar{\boldsymbol{u}}_k + \delta \boldsymbol{u}_k, \bar{\boldsymbol{v}}_k + \delta \boldsymbol{v}_k) &\approx \bar{\Theta} + \bar{\Theta}_x \delta \boldsymbol{x}_k + \bar{\Theta}_u \delta \boldsymbol{u}_k + \bar{\Theta}_v \delta \boldsymbol{v}_k \\ &+ \frac{1}{2}\begin{bmatrix} \delta \boldsymbol{x}_k \\ \delta \boldsymbol{u}_k \\ \delta \boldsymbol{v}_k \end{bmatrix}^T \begin{bmatrix} \bar{\Theta}_{xx} & \bar{\Theta}_{xu} & \bar{\Theta}_{xv} \\ \bar{\Theta}_{ux} & \bar{\Theta}_{uu} & \bar{\Theta}_{uv} \\ \bar{\Theta}_{vx} & \bar{\Theta}_{vu} & \bar{\Theta}_{vv} \end{bmatrix} \begin{bmatrix} \delta \boldsymbol{x}_k \\ \delta \boldsymbol{u}_k \\ \delta \boldsymbol{v}_k \end{bmatrix}\end{aligned} \tag{26}$$

where all the $\bar{\Theta}$-related terms are given as

$$\bar{\Theta}_x = \bar{V}_x(t_{k+1})^T \bar{\boldsymbol{A}}(t_k) + \bar{L}_x(t_k) \tag{27a}$$

$$\bar{\Theta}_u = \bar{V}_x(t_{k+1})^T \bar{\boldsymbol{B}}_u(t_k) + \bar{L}_u(t_k) \tag{27b}$$

$$\bar{\Theta}_v = \bar{V}_x(t_{k+1})^T \bar{\boldsymbol{B}}_v(t_k) + \bar{L}_v(t_k) \tag{27c}$$

$$\bar{\Theta}_{xx} = \bar{\boldsymbol{A}}(t_k)^T \bar{V}_{xx}(t_{k+1}) \bar{\boldsymbol{A}}(t_k) + \bar{L}_{xx}(t_k) \tag{27d}$$

$$\bar{\Theta}_{uu} = \bar{\boldsymbol{B}}_u(t_k)^T \bar{V}_{xx}(t_{k+1}) \bar{\boldsymbol{B}}_u(t_k) + \bar{L}_{uu}(t_k) \tag{27e}$$

$$\bar{\Theta}_{vv} = \bar{\boldsymbol{B}}_v(t_k)^T \bar{V}_{xx}(t_{k+1}) \bar{\boldsymbol{B}}_v(t_k) + \bar{L}_{vv}(t_k) \tag{27f}$$

$$\bar{\Theta}_{xu} = \bar{\boldsymbol{A}}(t_k)^T \bar{V}_{xx}(t_{k+1}) \bar{\boldsymbol{B}}_u(t_k) + \bar{L}_{xu}(t_k) \tag{27g}$$

$$\bar{\Theta}_{xv} = \bar{\boldsymbol{A}}(t_k)^T \bar{V}_{xx}(t_{k+1}) \bar{\boldsymbol{B}}_v(t_k) + \bar{L}_{xv}(t_k) \tag{27h}$$

$$\bar{\Theta}_{uv} = \bar{\boldsymbol{B}}_u(t_k)^T \bar{V}_{xx}(t_{k+1}) \bar{\boldsymbol{B}}_v(t_k) + \bar{L}_{uv}(t_k) \tag{27i}$$

$$\bar{\Theta}_{vu} = \bar{\Theta}_{uv}^T, \qquad \bar{\Theta}_{ux} = \bar{\Theta}_{xu}^T, \qquad \bar{\Theta}_{vx} = \bar{\Theta}_{xv}^T \tag{27j}$$

To find the optimal policies for $\delta \boldsymbol{u}_k$ and $\delta \boldsymbol{v}_k$ such that the second-order expansion of the function $\bar{\Theta}$ is optimized, we take the gradients of Eq. (26) with respect to $\delta \boldsymbol{u}_k$ and $\delta \boldsymbol{v}_k$ and set them to zero to obtain

$$\delta \boldsymbol{u}_k^* = -\bar{\Theta}_{uu}^{-1}(\bar{\Theta}_{ux}\delta \boldsymbol{x}_k + \bar{\Theta}_{uv}\delta \boldsymbol{v}_k^* + \bar{\Theta}_u) \tag{28a}$$

$$\delta \boldsymbol{v}_k^* = -\bar{\Theta}_{vv}^{-1}(\bar{\Theta}_{vx}\delta \boldsymbol{x}_k + \bar{\Theta}_{vu}\delta \boldsymbol{u}_k^* + \bar{\Theta}_v) \tag{28b}$$

Solving the system of equations (28) results in the expressions

$$\delta \boldsymbol{u}_k^* = l_u + K_u \delta \boldsymbol{x}_k \quad \text{and} \quad \delta \boldsymbol{v}_k^* = l_v + K_v \delta \boldsymbol{x}_k \tag{29}$$

where

$$l_u = -(\bar{\Theta}_{uu} - \bar{\Theta}_{uv}\bar{\Theta}_{vv}^{-1}\bar{\Theta}_{vu})^{-1}(\bar{\Theta}_u - \bar{\Theta}_{uv}\bar{\Theta}_{vv}^{-1}\bar{\Theta}_v) \tag{30}$$

$$l_v = -(\bar{\Theta}_{vv} - \bar{\Theta}_{vu}\bar{\Theta}_{uu}^{-1}\bar{\Theta}_{uv})^{-1}(\bar{\Theta}_v - \bar{\Theta}_{vu}\bar{\Theta}_{uu}^{-1}\bar{\Theta}_u) \tag{31}$$

$$K_u = -(\bar{\Theta}_{uu} - \bar{\Theta}_{uv}\bar{\Theta}_{vv}^{-1}\bar{\Theta}_{vu})^{-1}(\bar{\Theta}_{ux} - \bar{\Theta}_{uv}\bar{\Theta}_{vv}^{-1}\bar{\Theta}_{vx}) \tag{32}$$

$$K_v = -(\bar{\Theta}_{vv} - \bar{\Theta}_{vu}\bar{\Theta}_{uu}^{-1}\bar{\Theta}_{uv})^{-1}(\bar{\Theta}_{vx} - \bar{\Theta}_{vu}\bar{\Theta}_{uu}^{-1}\bar{\Theta}_{ux}) \tag{33}$$

By substituting the optimal control updates $\delta \boldsymbol{u}_k^*$ and $\delta \boldsymbol{v}_k^*$ from Eq. (29) into the value function, we can split the value function into zeroth-, first-, and second-order terms in $\delta \boldsymbol{x}_k$, such that $\bar{V}(\bar{\boldsymbol{x}}_k + \delta \boldsymbol{x}_k) = \bar{V}(t_k) + \bar{V}_x(t_k)^T \delta \boldsymbol{x}_k + (1/2)\delta \boldsymbol{x}_k^T \bar{V}_{xx}(t_k)\delta \boldsymbol{x}_k$, where $\bar{V}(t_k)$, $\bar{V}_x(t_k)$, and $\bar{V}_{xx}(t_k)$ are computed from

$$\begin{aligned}\bar{V}(t_k) = \bar{V}(t_{k+1}) &+ l_u^T \bar{\Theta}_u + l_v^T \bar{\Theta}_v + \frac{1}{2}(l_u^T \bar{\Theta}_{uu} l_u + l_v^T \bar{\Theta}_{vv} l_v \\ &+ l_u^T \bar{\Theta}_{uv} l_v + l_v^T \bar{\Theta}_{vu} l_u)\end{aligned} \tag{34}$$

$$\begin{aligned}\bar{V}_x(t_k) = \bar{\Theta}_x &+ K_u^T \bar{\Theta}_u + K_v^T \bar{\Theta}_v + \bar{\Theta}_{xu} l_u + \bar{\Theta}_{xv} l_v + K_u^T \bar{\Theta}_{uu} l_u \\ &+ K_v^T \bar{\Theta}_{vv} l_v + K_u^T \bar{\Theta}_{uv} l_v + K_v^T \bar{\Theta}_{vu} l_u\end{aligned} \tag{35}$$

$$\begin{aligned}\bar{V}_{xx}(t_k) = \bar{\Theta}_{xx} &+ K_u^T \bar{\Theta}_{ux} + \bar{\Theta}_{xu} K_u + K_v^T \bar{\Theta}_{vx} + \bar{\Theta}_{xv} K_v + K_u^T \bar{\Theta}_{uu} K_u \\ &+ K_v^T \bar{\Theta}_{vv} K_v + K_u^T \bar{\Theta}_{uv} K_v + K_v^T \bar{\Theta}_{vu} K_u\end{aligned} \tag{36}$$

Since the optimal cost-to-go is computed backward in time, boundary conditions need to be given at $t = t_f$. These boundary conditions are given by

$$\bar{V}(t_f) = \phi(\bar{\boldsymbol{x}}(t_f), t_f) \tag{37a}$$

$$\bar{V}_x(t_f) = \phi_x(\bar{\boldsymbol{x}}(t_f), t_f) \tag{37b}$$

$$\bar{V}_{xx}(t_f) = \phi_{xx}(\bar{\boldsymbol{x}}(t_f), t_f) \tag{37c}$$

In contrast to the backward difference equations (34) the corresponding equations in [5] lack all the terms involving $\bar{\Theta}_{uv}$.

Remark 2: Note that in order to find the optimal policies for $\delta \boldsymbol{u}_k^*$ and $\delta \boldsymbol{v}_k^*$ from Eqs. (28a) and (28b), we need to invert the matrices $\bar{\Theta}_{uu}$ and $\bar{\Theta}_{vv}$ given by Eqs. (27e) and (27f), respectively. Therefore, it is required that $\bar{\Theta}_{uu}$ and $\bar{\Theta}_{vv}$ are invertible. Moreover, in order for $\boldsymbol{u}_k$ to be a minimizer of the value function, the matrix $\bar{\Theta}_{uu}$ must be positive definite. Likewise, $\bar{\Theta}_{vv}$ must be negative definite so that $\boldsymbol{v}_k$ is a maximizer. In the following, we present a technique to find new matrices $\tilde{\Theta}_{uu}$ and $\tilde{\Theta}_{vv}$ that are positive and negative definite, respectively, while resembling the original $\bar{\Theta}_{uu}$ and $\bar{\Theta}_{vv}$, using the following regularization scheme [14]. For $\bar{\Theta}_{uu}$, we first compute the eigenvalue decomposition $V_u D_u V_u^T = \bar{\Theta}_{uu}$ and we replace all elements of the diagonal matrix D_u that are smaller than ξ_u with ξ_u to obtain a new diagonal matrix $\tilde{D}_u$, where $\xi_u > 0$ is a prescribed small constant, and set $\tilde{\Theta}_{uu} = V_u \tilde{D}_u V_u^T$. Similarly, $\tilde{\Theta}_{vv}$ can be computed by $\tilde{\Theta}_{vv} = V_v \tilde{D}_v V_v^T$, where $V_v D_v V_v^T = \bar{\Theta}_{vv}$ is the eigenvalue decomposition of $\bar{\Theta}_{vv}$, and $\tilde{D}_u$ is generated by replacing all elements of the diagonal matrix D_u that are larger than $-\xi_v$ with $-\xi_v$, where $\xi_v > 0$ is a small constant.

IV. Comparison Between Continuous and Discrete GT-DDP

In this section, we discuss the differences between the continuous and discrete formulations of GT-DDP and explain why the continuous version may be preferable for most engineering applications. Besides the form of the backward differential equations, one of the major differences between the discrete and continuous time formulations lies in the specification of the terms $\bar{Q}_{uu}$ and $\bar{Q}_{vv}$ (continuous formulation) and $\bar{\Theta}_{uu}$ and $\bar{\Theta}_{vv}$ (discrete formulation). In the continuous case, these terms are specified by $\bar{\mathcal{L}}_{uu}$ and $\bar{\mathcal{L}}_{vv}$ from Eqs. (10e) and (10f), and therefore they are completely determined by the user. This is not the case with the discrete time formulation of the GT-DDP. To see the effect of the cost function in the feed-forward and feedback gains on continuous time GT-DDP, we recall that $\bar{Q}_{uu} = \bar{\mathcal{L}}_{uu}$ and $\bar{Q}_{vv} = \bar{\mathcal{L}}_{vv}$. Moreover, since $\bar{\mathcal{L}}_{uu}, \bar{\mathcal{L}}_{vv}$ are design parameters, we can choose them such that $\bar{\mathcal{L}}_{uu} > 0$ and $\bar{\mathcal{L}}_{vv} < 0$. Note also that the positive definiteness of $\bar{\mathcal{L}}_{uu}$ and negative definiteness of $\bar{\mathcal{L}}_{vv}$ are required since the role of the first controller/player is to minimize the cost while the role of the second controller/player is to maximize it. Given $\bar{Q}_{uu} > 0$ and $\bar{Q}_{vv} < 0$ we have the following expressions:

$$\bar{Q}_{uu} - \bar{Q}_{uv}\bar{Q}_{vv}^{-1}\bar{Q}_{vu} > 0 \Rightarrow (\bar{Q}_{uu} - \bar{Q}_{uv}\bar{Q}_{vv}^{-1}\bar{Q}_{vu})^{-1} > 0 \tag{38}$$

$$\bar{Q}_{vv} - \bar{Q}_{vu}\bar{Q}_{uu}^{-1}\bar{Q}_{uv} < 0 \Rightarrow (\bar{Q}_{vv} - \bar{Q}_{vu}\bar{Q}_{uu}^{-1}\bar{Q}_{uv})^{-1} < 0 \tag{39}$$

The matrix inequalities (38) and (39) show that the feed-forward and feedback parts of the control policies of the two players will operate such that the first player aims to reduce the cost while the second player aims to increase it.

On the other hand, in the discrete time GT-DDP case, we have

$$\bar{\Theta}_{uu} = \bar{\boldsymbol{B}}_u^T \bar{V}_{xx} \bar{\boldsymbol{B}}_u + \bar{\mathcal{L}}_{uu} \tag{40}$$

$$\bar{\Theta}_{vv} = \bar{\boldsymbol{B}}_v^T \bar{V}_{xx} \bar{\boldsymbol{B}}_v + \bar{\mathcal{L}}_{vv} \tag{41}$$

Hence, there is no guarantee that $\bar{\Theta}_{uu}$ is positive definite and $\bar{\Theta}_{vv}$ is negative definite at the same time since they depend on the sign definiteness of $\bar{V}_{xx}$, which is not known in advance. The modification proposed in Remark 2 can be implemented to maintain the definiteness of $\bar{\Theta}_{uu}$ and $\bar{\Theta}_{vv}$. Nonetheless, the convergence of the discrete time GT-DDP may not be easy to achieve.

In terms of running time, the continuous-time GT-DDP requires the usage of differential equation solvers, whereas only arithmetic calculations are needed in discrete time GT-DDP. Therefore, the time cost per iteration is less in the discrete time GT-DDP case, where each iteration is given by an update of the controls.

Algorithm 1 GT-DDP Algorithm

Input: Initial condition of the dynamics $\boldsymbol{x}_0$, initial minimizing control $\bar{\boldsymbol{u}}$ and maximizing control $\bar{\boldsymbol{v}}$, final time t_f, multiplier γ, a large integer N, and a small constant ϵ.
Output: Optimal minimizing control $\boldsymbol{u}^*$, optimal maximizing control $\boldsymbol{v}^*$, and the corresponding optimal gains $\boldsymbol{l}_u, \boldsymbol{K}_u, \boldsymbol{l}_v, \boldsymbol{K}_v$.

procedure Update_Control $\boldsymbol{x}_0, \bar{\boldsymbol{u}}, \bar{\boldsymbol{v}}, t_f$
  for $i = 1$ **to** N **do**
    while $|\delta \boldsymbol{u}| + |\delta \boldsymbol{v}| > \epsilon$ **do**
      Get the initial trajectory $\bar{\boldsymbol{x}}$ by integrating dynamics (1) forward with $\boldsymbol{x}_0, \bar{\boldsymbol{u}}, \bar{\boldsymbol{v}}$;
      Compute the value of V, V_x, V_{xx} at t_f according to Eq. (18);
      Integrate backward the Eqs. (17);
      Compute $\boldsymbol{l}_u, \boldsymbol{K}_u, \boldsymbol{l}_v, \boldsymbol{K}_v$ from Eqs. (13) to (16);
      Integrate Eq. (8) forward by replacing $\delta \boldsymbol{u}$ and $\delta \boldsymbol{v}$ with $(\boldsymbol{l}_u + \boldsymbol{K}_u \delta \boldsymbol{x})$ and $(\boldsymbol{l}_v + \boldsymbol{K}_v \delta \boldsymbol{x})$, respectively, to get $\delta \boldsymbol{x}(t)$;
      Compute $\delta \boldsymbol{u} = \boldsymbol{l}_u + \boldsymbol{K}_u \delta \boldsymbol{x}$ and $\delta \boldsymbol{v} = \boldsymbol{l}_v + \boldsymbol{K}_v \delta \boldsymbol{x}$;
      Update control $\bar{\boldsymbol{u}} = \bar{\boldsymbol{u}} + \gamma \delta \boldsymbol{u}$ and $\bar{\boldsymbol{v}} = \bar{\boldsymbol{v}} + \gamma \delta \boldsymbol{v}$, where $\gamma \in [0, 1]$;
      Set $\boldsymbol{u}^* = \bar{\boldsymbol{u}}$ and $\boldsymbol{v}^* = \bar{\boldsymbol{v}}$;
    end while
  end for
  return $\boldsymbol{u}^*, \boldsymbol{v}^*, \boldsymbol{l}_u, \boldsymbol{K}_u, \boldsymbol{l}_v, \boldsymbol{K}_v$.
end procedure

The continuous-time GT-DDP algorithm is provided in Algorithm 1. The discrete-time GT-DDP algorithm follows a similar approach and thus it is omitted for the sake of brevity.

An interesting characteristic of trajectory optimization methods such as DDP is that they provide the locally optimal state trajectory, optimal feed-forward control, and locally optimal feedback gains. In Sec. V we demonstrate the effect of the cost function on the feed-forward and feedback parts of the minimizing control policy for different choices of $\bar{\mathcal{L}}_{vv}$.

V. Numerical Examples and Experimental Validation

In this section, we apply our algorithm to several min-max problems to test its performance. First, we apply the discrete and continuous time GT-DDP algorithms to the inverted pendulum problem under conflicting controls to compare between these two methods. We then apply the continuous time GT-DDP to a quadrotor system with conflicting controls. Afterward, we use the gains we obtain from the previous min-max problems to bring the inverted pendulum and the quadrotor to their respective desired states under the presence of stochastic disturbances in the control channel. Although the proposed GT-DDP algorithm is not designed for systems subject to stochastic disturbances, the numerical examples below reveal that the min-max formulation leads to feedback controllers that respond well to stochastic disturbances as well.

A. Min-Max Problems

1. *Inverted Pendulum Problem*

We first apply the proposed discrete and continuous time algorithms on a simple inverted pendulum with conflicting controls to get a better insight on the impact of the conflicting control in the overall system performance. The dynamics are given by

$$I\ddot{\theta} + b\dot{\theta} - mg\ell \sin\theta = \boldsymbol{u} + \boldsymbol{v} \tag{42}$$

where the parameters in the simulations are chosen as $m = 1$ kg, $\ell = 0.5$ m, $b = 0.1$ (N · s)/m, $I = m\ell^2$, $g = 9.81$ (kg · m)/s^2. The goal is to bring the pendulum from the initial state $[\theta, \dot{\theta}] = [\pi, 0]$ to $[\theta, \dot{\theta}] = [0, 0]$.

We cast this problem as a trajectory optimization problem with two conflicting controls. The cost function is given by

$$J = \int_0^{t_f} (\boldsymbol{x}^T Q \boldsymbol{x} + \boldsymbol{u}^T R_u \boldsymbol{u} - \boldsymbol{v}^T R_v \boldsymbol{v})\, \mathrm{d}t \tag{43}$$

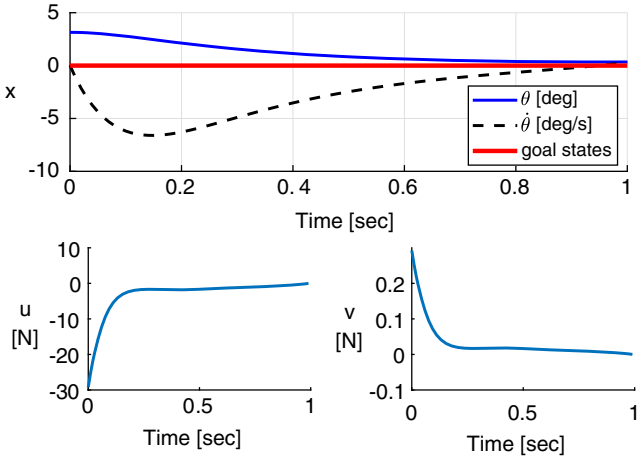
where $\boldsymbol{x} = [\theta, \dot{\theta}]^T$, $Q = \mathrm{diag}[1, 0.1]$, $R_u = 0.01$, and $R_v = 1$. Note that we are more interested about cases where R_v is not too large

compared with R_u. When R_v is relatively too large compared with R_u, the maximizing control is negligible. In the extreme case, when $R_v \to \infty$, we obtain an optimal control problem.

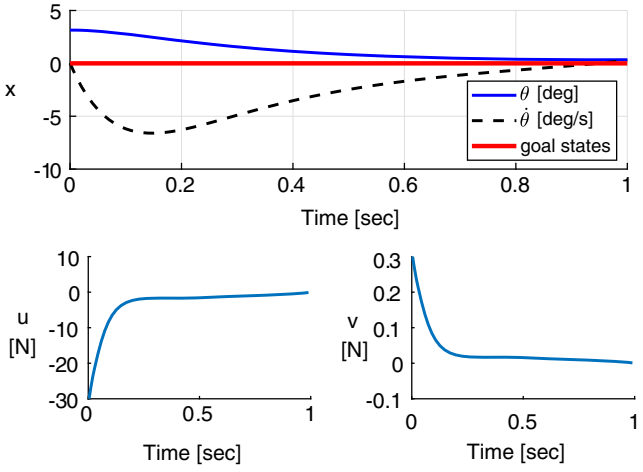
We set the initial control to be $\boldsymbol{u} \equiv 0$, $\boldsymbol{v} \equiv 0$ and the terminal time to be $t_f = 1$. The step multiplier $\gamma = 0.8$ for the continuous case. Initial controls and the corresponding initial trajectories of the states are identical in both cases. We ran the algorithms for 20 iterations and convergence was achieved in both cases. Figures 1a and 1b present the optimal controls of $\boldsymbol{u}$ and $\boldsymbol{v}$ for the discrete and continuous time cases, respectively, as well as the corresponding optimal trajectories of the states θ, $\dot{\theta}$. The two algorithms yield the same results, as expected.

As can be seen in Fig. 2, the cost converges in 11 iterations in the discrete-time case. On the other hand, the continuous-time GT-DDP converges in 4 iterations.

We also performed simulations with different choices of R_v for both continuous-time and discrete-time GT-DDP, and we altered the discretization step to see how it affects the outcome. It was observed that in the continuous GT-DDP case, R_v can be chosen to be $R_v \geq 0.011$ and the convergence is sustained, whereas in the discrete GT-DDP case, R_v can only be chosen to be $R_v \geq 0.31$ for the algorithm to converge when $\mathrm{d}t = 0.05$. Once we set $\mathrm{d}t = 0.01$, then $R_v = 0.53$ would cause the algorithm to diverge. These numerical observations seem to indicate that the continuous version of GT-DDP allows for a larger range of R_v than the discrete version, and that the discrete GT-DDP is somewhat sensitive to the choice of the discretization step, which essentially adds another tuning parameter and makes the algorithm harder to apply. Further investigation is needed, however, to better assess the impact of the discretization step to the convergence of the discrete GT-DDP.



a) Discrete GT-DDP case



b) Continuous GT-DDP case

Fig. 1 Comparison between discrete and continuous time GT-DDP.

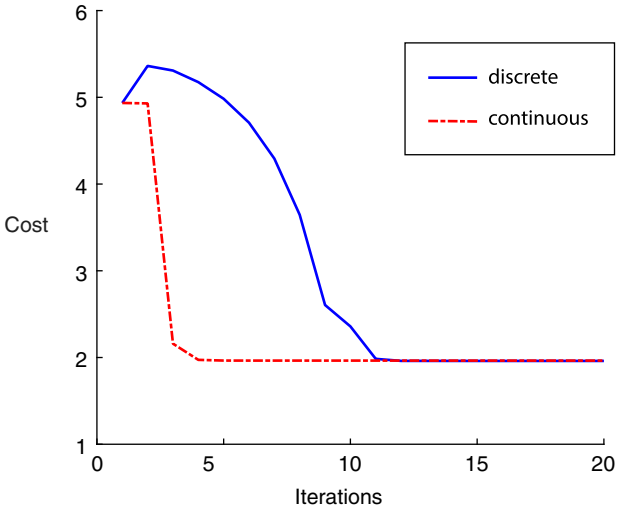


Fig. 2 Cost per iteration under continuous GT-DDP in dashed red and discrete GT-DDP in blue.

2. *Quadrotor*

In this section we apply the GT-DDP algorithm to a more complicated and realistic example, namely, a quadrotor system. In the next section we present the results from an experimental implementation of the algorithm. The dynamic model of the quadrotor includes 16 states: 3 for the position ($\boldsymbol{r} = (x, y, z)^T$), 3 for the Euler angles ($\Phi = (\phi, \theta, \psi)^T$), 3 for the velocity ($\dot{\boldsymbol{r}} = (\dot{x}, \dot{y}, \dot{z})^T$), 3 for the body angular rates ($\dot{\Phi} = (p, q, r)^T$), and 4 for the motor speeds ($\Omega = (\omega_1, \omega_2, \omega_3, \omega_4)^T$). The coordinate systems and free body diagram for the quadrotor are presented in Fig. 3. The inertial frame is defined by axes x_I, y_I, and z_I, where z_I points upward. The body frame B is attached to the quadrotor with x_B coinciding with the forward direction and z_B perpendicular to the plane of rotors pointing upward during perfect hover. The corresponding dynamics of the quadrotor with conflicting controls is given as follows:

$$\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = f(\boldsymbol{x}) + G(\boldsymbol{u} + \boldsymbol{v}) \tag{44}$$

where $\boldsymbol{x} = [\boldsymbol{r}, \Phi, \dot{\boldsymbol{r}}, \dot{\Phi}, \Omega]^T \in \mathbb{R}^{16}$. The form of $f(\boldsymbol{x})$ and the matrix $G \in \mathbb{R}^{16\times 4}$ can be found in [33].

The minimizing control is $\boldsymbol{u} = (u_1, u_2, u_3, u_4)^T \in \mathbb{R}^4$. Here u_1 results in a net force along the z_B axis and u_2, u_3, u_4 generate roll, pitch, and yaw moments, respectively, whereas $\boldsymbol{v} \in \mathbb{R}^4$ denotes the maximizing control. The corresponding cost function is defined as

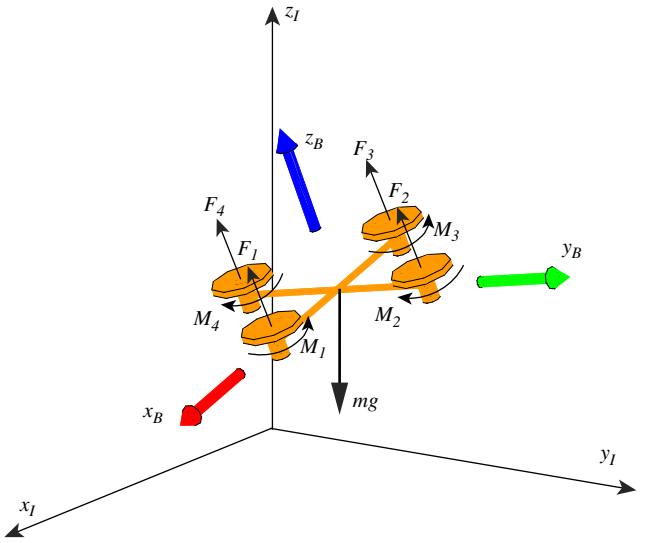
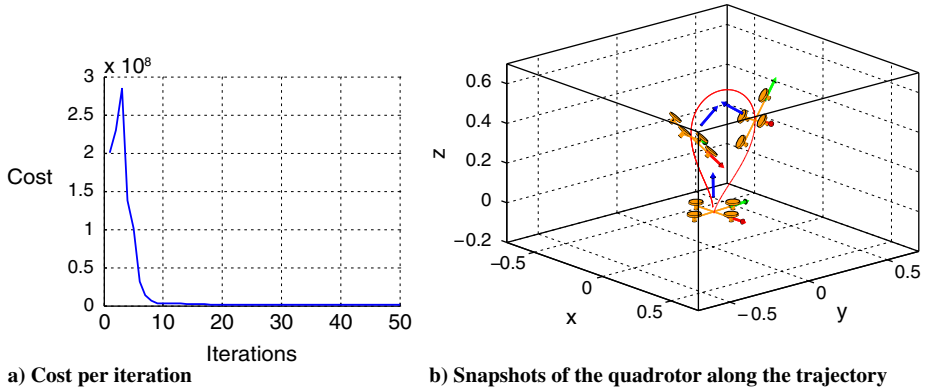
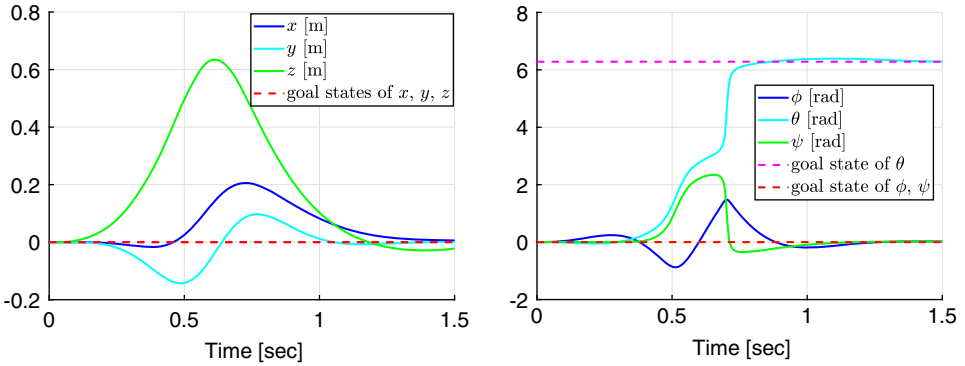


Fig. 3 Coordinate systems and forces/moments acting on a quadrotor.

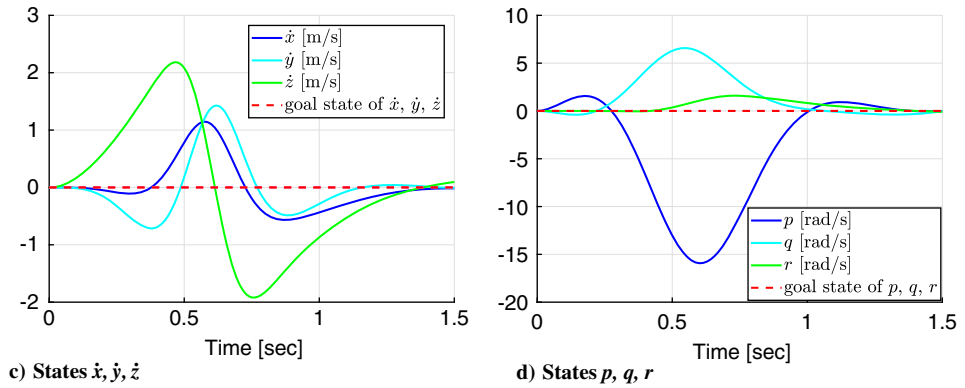


a) Cost per iteration b) Snapshots of the quadrotor along the trajectory

Fig. 4 Cost and snapshot of the quadrotor with conflicting controls.



a) States x, y, z b) States ϕ, θ, ψ



c) States $\dot{x}, \dot{y}, \dot{z}$ d) States p, q, r

Fig. 5 Optimal trajectories of the states and corresponding goal states.

$$J = \frac{1}{2}\int_0^{t_f} [(\boldsymbol{x}-\boldsymbol{x}_f)^T Q(\boldsymbol{x}-\boldsymbol{x}_f) + \boldsymbol{u}^T R_u \boldsymbol{u} - \boldsymbol{v}^T R_v \boldsymbol{v}] + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}_f)^T Q_f (\boldsymbol{x}-\boldsymbol{x}_f) \tag{45}$$

where $\boldsymbol{x}_f \in \mathbb{R}^{16}$ denotes the desired terminal states. The goal of this task is to steer the quadrotor to turn 180 deg about the y_I axis. The state-dependent running cost is included for better convergence. In the simulations, we set $\boldsymbol{x}_{f_5} = 2\pi$ and $\boldsymbol{x}_{f_i} = 0$ otherwise, and

$$[Q_f]_{i,i} = \begin{cases} 10^7, & i = 1,2,3,5; \\ 10^6 & i = 4,6,7,8,9; \\ 10^5 & i = 10,11,12; \\ 0 & \text{otherwise} \end{cases} \tag{46}$$

and all the off-diagonal terms are assigned to zero. We also chose $Q = 0.01Q_f$, $R_u = 0.0001I$, and $R_v = 0.0005I$. Finally, we let $\gamma = 0.5$.

The desired terminal state $\boldsymbol{x}_f$ is chosen such that the quadrotor changes its pitch angle by 2π and returns to its original hovering position. Fifty iterations were performed to ensure convergence. The cost per iteration is shown in Fig. 4a. A snapshot of the quadrotor along the optimal trajectory is depicted in Fig. 4b. The corresponding optimal state trajectories are shown in Fig. 5.

B. Experimental Results

The continuous version of the GT-DDP algorithm was also implemented on a Parrot AR.Drone 2.0 quadrotor. Two path-tracking flight tests were performed to validate the theory, especially the ability of GT-DDP to deal with disturbances. In the first test we employed a sling load (Fig. 6) and in the second test we applied an external wind field to create model discrepancies and external disturbances unknown to the system. We then compared the GT-DDP controller with a standard DDP controller to validate the robustness of the GT-DDP controller.

We first solved for the optimal trajectory that minimizes the tracking error from some nominal trajectory along with the control

Fig. 6 Sling load generates significant model error and pendulous oscillation disturbance in flight.

effort, and then used the DDP or GT-DDP solutions as reference state, input, and feedback gain trajectories in our low-level controllers.

For the controller implementation we had to translate the low-level inputs obtained via the DDP/GT-DDP to the high-level control inputs (e.g., attitude) tailored to the system operating frequency. The dynamics of the four propeller rotors are much faster than the dynamics of the quadrotor main body, and direct rotor control requires significantly higher frequency operating rate to stabilize the full-state of the drone [34]. We thus relied on the onboard attitude controller of AR.Drone 2.0 to command the high-level inputs from the DDP/GT-DDP controller to the individual rotor commands.

The system architecture consists of three main units: the path planner, the state estimator, and the GT-DDP unit, as shown in Fig. 7. The interconnection among these three units was done using the Robot Operating System (ROS) framework. The control command in the GT-DDP unit is calculated in the ground control source (GCS) running on an Intel Xeon W-2135 6-core processor with 32 GB RAM and commanded to AR.Drone 2.0 through Wi-Fi via the ardrone_autonomy ROS driver package [35]. We used a VICON motion capture system and the mocap_vicon ROS package [36] to estimate the full state of the drone at 100 Hz for direct feedback.

Within the GT-DDP controller block the workload is divided into three subunits: the controller node, the optimizer, and the controller. The controller node takes care of the communication with the other units by publishing and subscribing to ROS messaging, while managing the optimizer and the controller units to perform their own tasks. Once the target state trajectory $\boldsymbol{x}_d$ from the path planner is fed to the controller node, the node assigns the optimizer, the current state, and the target state trajectory to obtain the optimal state $\boldsymbol{x}^*$, input $\boldsymbol{u}^*$, and gain $\boldsymbol{K}_u$ histories based on the current state and the pre-assigned cost matrix.

The major functionality of the controller block is to compute the feedback control policy throughout the execution of the program. The controller stores the reference state $\boldsymbol{x}^*$, reference input $\boldsymbol{u}^*$, and gain $\boldsymbol{K}_u$, which are updated at ROS timer call over the time horizon. Once the optimal state, input, and gain trajectories are computed from the optimizer, we use the ROS timer to update the controller's internal data at each time segment. Specifically, the trajectories are stored as a function of time data type (e.g., using splines), and the timer callback generates a time t to compute the state, the input, and the gain from these trajectories. Concurrently, the controller node calls the controller's function to compute the corrective policy, that is, $\boldsymbol{u} = \boldsymbol{u}^* + \boldsymbol{K}_u(\boldsymbol{x} - \boldsymbol{x}^*)$, after receiving the latest state estimate $\boldsymbol{x}$. This multithread controller structure is adopted from the rotors_simulator ROS package [37].

One important aspect of the controller architecture used, already briefly mentioned earlier, is that we need to map the low-level control inputs obtained with the DDP/GT-DDP algorithm to high-level

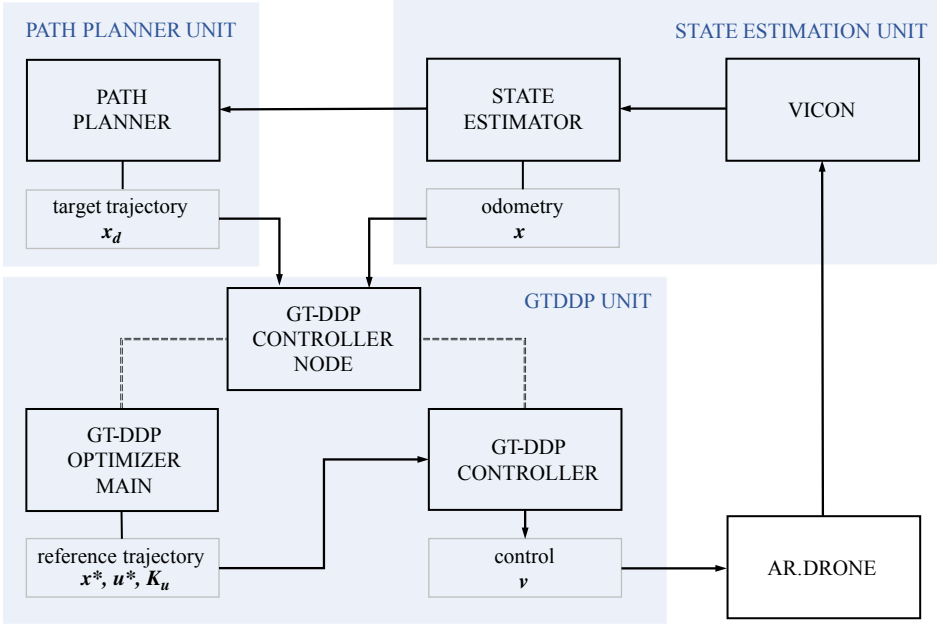


Fig. 7 System architecture.

control inputs to be fed to the onboard attitude controller. Specifically, the AR.Drone 2.0 only accepts $(\phi, \theta, r, \dot{z})$ as inputs, that is, roll, pitch, yaw rate, and vertical speed. We obtain these by integrating the equation of motion. Namely, we find the high-level input command $\boldsymbol{\nu} = (\phi, \theta, r, \dot{z})$ to the quadrotor by numerically integrating the dynamics model of the AR. Drone using the current state estimate $\boldsymbol{x}$ and the computed feedback control $\boldsymbol{u} = \boldsymbol{u}^* + \boldsymbol{K}_u(\boldsymbol{x} - \boldsymbol{x}^*)$. We denote this operation as $\boldsymbol{\nu} = h(\boldsymbol{x}, \boldsymbol{u}, \Delta t)$ and yields the predicted four state variables Δt seconds ahead. The prediction window Δt can be adjusted according to the controller operating frequency.

We use proportional + derivative loop around $\boldsymbol{\nu}$ with the VICON state estimate to ensure that the quadrotor follows the high-level command. The schematic of the interconnection of the various signals within the controller is shown in Fig. 8.

The GT-DDP optimizer program was coded in C/C++ to perform the tasks listed in Algorithm 1. We used the boost odeint C++ library to numerically integrate the quadrotor dynamics forward and the value function equations backward it time with the Dormand-Prince fifth-order method. The program stores and manages the state, the input, and the gain trajectories using the Spline class provided in the Eigen C++ library.

Two different trajectories, namely, a figure-eight and an inclined circle, were tested to validate the GT-DDP controller tracking performance. To help convergence, the initial trajectory was given in the differential flat output space $\boldsymbol{\sigma} = [x, y, z, \psi]^T$, which was then used to obtain $\boldsymbol{\nu}_d = [\phi, \theta, r, \dot{z}]^T$ as proposed in [38]. By introducing the intermediate inputs $\boldsymbol{u} = [\boldsymbol{u}_p, u_\psi]^T$ where $\boldsymbol{u}_p = [\ddot{\sigma}_1, \ddot{\sigma}_2, \ddot{\sigma}_3 + g]^T$ and $u_\psi = \dot{\sigma}_4$, we map the flat outputs $\boldsymbol{\sigma}$ to the inputs of interest $\boldsymbol{\nu}_d$. This mapping is restricted to twice differentiable trajectories with ϕ lying within $[-\pi/2, \pi/2]$, while not free-falling, that is, $\ddot{z} = -g$. Although not necessary, we use the target state trajectory $\boldsymbol{x}_d$ as $\boldsymbol{\sigma}$, $\dot{\boldsymbol{\sigma}}$, and $\boldsymbol{\nu}_d$ to accelerate the convergence of the DDP/GT-PPD optimizer.

The figure-eight trajectory was given as $[x, y, z, \psi] = [\alpha\cos(\omega/2t), \beta\sin(\omega t), \gamma, 0]$, where α and β are the amplitude of the figure-eight and γ is the height. The test results with $\alpha = 0.8$ m, $\beta = 0.6$ m, and $\gamma = 1.0$ m with $\omega = 1.256$ rad/s are shown in Fig. 9. In our second tracking test, the target trajectory was an inclined circle, such that $\boldsymbol{\sigma}_d = [x, y, z, \psi]^T = [\alpha\cos(\omega t), \beta\sin(\omega t), \gamma\cos(\omega t) + 2, 0]^T$, where α, β, and γ are the amplitude in the x, y, z directions accordingly. We set $\alpha = 0.9$ m, $\beta = 0.8$ m, and $\gamma = 0.45$ m with $\omega = 0.98$ rad/s. The tracking results for the inclined circular trajectory are shown in Fig. 10.

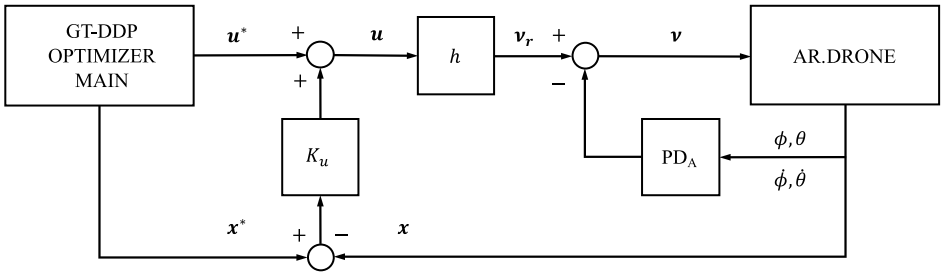


Fig. 8 GT-DDP controller signal flow schematic.

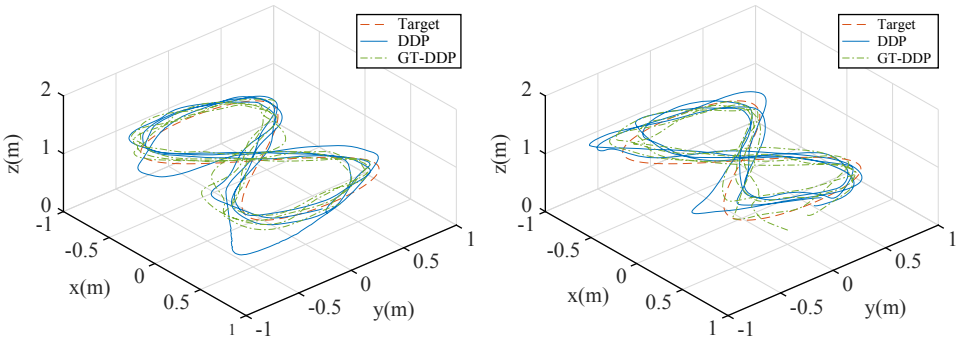


Fig. 9 GT-DDP (dash-dot) and DDP (solid) path tracking flight result for target path figure eight (dash) with sling load (left) and with external wind field (right).

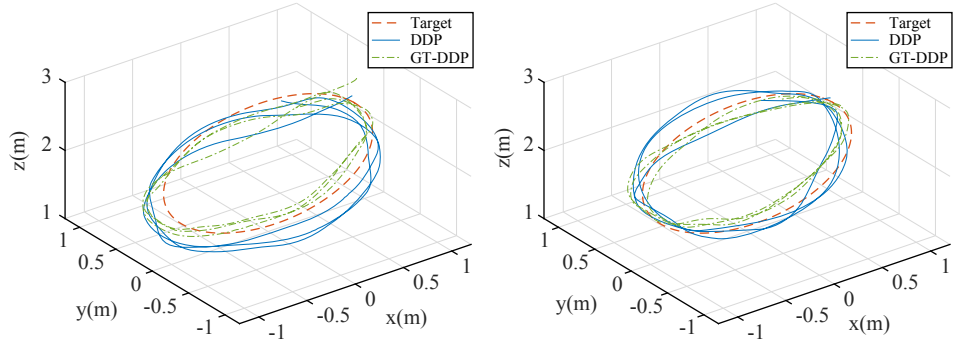


Fig. 10 GT-DDP (dash-dot) and DDP (solid) path tracking flight result for target path inclined circle (dash) with sling load (left) and with external wind field (right).

Two different types of external disturbances were investigated: the addition of a sling load and the effect of an external wind field. In the first case, we attached a 100 g mass 57 cm below the quadrotor center of mass with a flexible string to create nonnegligible model error and pendulous oscillation during the flight. The load adds 21% to the total mass of the quadrotor, and it freely swings during the flight. During the second test case, we applied a lateral wind with constant airspeed, using a 30-inch blade fan capable of 12,750 CFM. The approximate wind speed at the center of the trajectory is about 9.8 mph, as provided by the manufacturer.

The results for both cases are shown in Figs. 9 and 10. In both cases the GT-DDP performed better than the DDP. Error plots quantify the difference in performance but for the sake of brevity are not shown here, especially because the difference in performance is even noticeable when plotting the corresponding trajectories recorded by the VICON system.

VI. Conclusions

In this paper, a differential game problem with two conflicting controls has been considered. The problem has been formulated in continuous time by first considering a first-order expansion of the dynamics and a quadratic expansion of the cost function. By taking the Taylor series expansion of the associated Hamilton–Jacobi–Bellman–Isaacs equation around a nominal trajectory, the update law of both minimizing and maximizing controls, as well as the backward propagation equations of the zeroth-, first-, and second-order approximation terms of the value function, has been derived. By discretizing along the nominal trajectory, the derivation of the Game-Theoretic Differential Dynamic Programming (GT-DDP) algorithm in discrete time is then also presented. The advantages of the continuous-time GT-DDP over its discrete-time counterpart are discussed, and the performance of the two formulations is compared using several examples with conflicting controls. Specifically, it has been shown that the continuous-time formulation enjoys improved convergence properties compared with its discrete-time counterpart. The continuous-time GT-DDP algorithm is tested on a nonlinear control problem, namely, a quadrotor flight steering. The example showcases the ability of the GT-DDP algorithm to handle relatively high-degree-of-freedom systems. Lastly, GT-DDP on a quadrotor has been implemented with a sling load attached to it and subject to wind disturbances, and the superior performance of the GT-DDP algorithm compared with the standard DDP algorithm in terms or robustness has been shown, validating the benefits of a min-max formulation for handling disturbances.

Appendix A: Derivation of the Expansion of the HJB Equation

Starting with the left-hand side of Eq. (5) we have:

$$\frac{\partial V(\boldsymbol{x},t)}{\partial t} = \frac{\partial V(\bar{\boldsymbol{x}}+\delta\boldsymbol{x},t)}{\partial t} \approx \frac{\partial \bar{V}}{\partial t} + \frac{\partial \bar{V}_{\boldsymbol{x}}^T}{\partial t}\delta\boldsymbol{x} + \frac{1}{2}\delta\boldsymbol{x}^T\frac{\partial \bar{V}_{xx}}{\partial t}\delta\boldsymbol{x} \tag{A1}$$

We also have

$$\frac{\mathrm{d}\bar{V}}{\mathrm{d}t} = \frac{\partial \bar{V}}{\partial t} + \bar{V}_x^T\frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}t} = \frac{\partial \bar{V}}{\partial t} + \bar{V}_x^T\bar{F} \tag{A2}$$

Thus, we get the expression

$$\frac{\partial \bar{V}}{\partial t} = \frac{\mathrm{d}\bar{V}}{\mathrm{d}t} - \bar{V}_x^T\bar{F} \tag{A3}$$

Similarly,

$$\frac{\partial \bar{V}_x}{\partial t} = \frac{\mathrm{d}\bar{V}_x}{\mathrm{d}t} - \bar{V}_{xx}\bar{F} \tag{A4}$$

Finally, the partial time derivative of the Hessian of the value function takes the form:

$$\frac{\partial \bar{V}_{xx}}{\partial t} = \frac{\mathrm{d}\bar{V}_{xx}}{\mathrm{d}t} - \sum_{i=1}^{n}\bar{V}_{xxx}^{(i)}\bar{F}^{(i)} \tag{A5}$$

where $\bar{V}_{xxx}^{(i)}$ denotes the Hessian matrix of the ith element of $\bar{V}_x$ and $\bar{F}^{(i)}$ denotes the ith element of $\bar{F}$. Using the expressions (A2–A4), the left-hand side of Eq. (5) thus becomes

$$\begin{aligned}-\frac{\partial V(\boldsymbol{x},t)}{\partial t} \approx{}& -\frac{\mathrm{d}\bar{V}}{\mathrm{d}t} - \frac{\mathrm{d}\bar{V}_x^T}{\mathrm{d}t}\delta\boldsymbol{x} - \frac{1}{2}\delta\boldsymbol{x}^T\frac{\mathrm{d}\bar{V}_{xx}}{\mathrm{d}t}\delta\boldsymbol{x} + \bar{V}_x^T\bar{F} + \delta\boldsymbol{x}^T\bar{V}_{xx}\bar{F}\\ &+ \frac{1}{2}\delta\boldsymbol{x}^T\left(\sum_{i=1}^{n}\bar{V}_{xxx}^{(i)}\bar{F}^{(i)}\right)\delta\boldsymbol{x}\end{aligned} \tag{A6}$$

We now turn our attention to the expansion of the right-hand side of Eq. (5). We start by writing

$$\begin{aligned}\mathcal{L}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{v},t) &= \mathcal{L}(\bar{\boldsymbol{x}}+\delta\boldsymbol{x},\bar{\boldsymbol{u}}+\delta\boldsymbol{u},\bar{\boldsymbol{v}}+\delta\boldsymbol{v},t)\\ &\approx \bar{\mathcal{L}} + \bar{\mathcal{L}}_x^T\delta\boldsymbol{x} + \bar{\mathcal{L}}_u^T\delta\boldsymbol{u} + \bar{\mathcal{L}}_v^T\delta\boldsymbol{v}\\ &\quad + \frac{1}{2}\begin{bmatrix}\delta\boldsymbol{x}\\ \delta\boldsymbol{u}\\ \delta\boldsymbol{v}\end{bmatrix}^T\begin{bmatrix}\bar{\mathcal{L}}_{xx} & \bar{\mathcal{L}}_{xu} & \bar{\mathcal{L}}_{xv}\\ \bar{\mathcal{L}}_{ux} & \bar{\mathcal{L}}_{uu} & \bar{\mathcal{L}}_{uv}\\ \bar{\mathcal{L}}_{vx} & \bar{\mathcal{L}}_{vu} & \bar{\mathcal{L}}_{vv}\end{bmatrix}\begin{bmatrix}\delta\boldsymbol{x}\\ \delta\boldsymbol{u}\\ \delta\boldsymbol{v}\end{bmatrix}\end{aligned} \tag{A7}$$

By expanding V_x around $\bar{\boldsymbol{x}}$ we have

$$V_x(\boldsymbol{x},t) = V_x(\bar{\boldsymbol{x}}+\delta\boldsymbol{x},t) \approx \bar{V}_x + \bar{V}_{xx}\delta\boldsymbol{x} + \frac{1}{2}\bar{W} \tag{A8}$$

where $\bar{W}\in\mathbb{R}^n$ is defined as

$$\bar{W}^{(i)} = \delta\boldsymbol{x}^T\bar{V}_{xxx}^{(i)}\delta\boldsymbol{x}, \qquad i = 1,\ldots,n$$

The system Eq. (1) is also expanded up to the first order, that is,

$$\begin{aligned}F(\boldsymbol{x},\boldsymbol{u},\boldsymbol{v},t) &= F(\bar{\boldsymbol{x}}+\delta\boldsymbol{x},\bar{\boldsymbol{u}}+\delta\boldsymbol{u},\bar{\boldsymbol{v}}+\delta\boldsymbol{v},t)\\ &\approx \bar{F} + \bar{F}_x\delta\boldsymbol{x} + \bar{F}_u\delta\boldsymbol{u} + \bar{F}_v\delta\boldsymbol{v}\end{aligned} \tag{A9}$$

Therefore, the right-hand side of Eq. (5) can be approximated as

$$\begin{aligned}&\min_u\max_v\{\mathcal{L}(\boldsymbol{x},\boldsymbol{u},\boldsymbol{v},t) + V_x^TF(\boldsymbol{x},\boldsymbol{u},\boldsymbol{v},t)\}\\ &\approx \min_{\delta u}\max_{\delta v}\Bigg\{\bar{\mathcal{L}} + \bar{\mathcal{L}}_x^T\delta\boldsymbol{x} + \bar{\mathcal{L}}_u^T\delta\boldsymbol{u} + \bar{\mathcal{L}}_v^T\delta\boldsymbol{v}\\ &\quad + \frac{1}{2}\begin{bmatrix}\delta\boldsymbol{x}\\ \delta\boldsymbol{u}\\ \delta\boldsymbol{v}\end{bmatrix}^T\begin{bmatrix}\bar{\mathcal{L}}_{xx} & \bar{\mathcal{L}}_{xu} & \bar{\mathcal{L}}_{xv}\\ \bar{\mathcal{L}}_{ux} & \bar{\mathcal{L}}_{uu} & \bar{\mathcal{L}}_{uv}\\ \bar{\mathcal{L}}_{vx} & \bar{\mathcal{L}}_{vu} & \bar{\mathcal{L}}_{vv}\end{bmatrix}\begin{bmatrix}\delta\boldsymbol{x}\\ \delta\boldsymbol{u}\\ \delta\boldsymbol{v}\end{bmatrix} + \bar{V}_x^T\bar{F} + \bar{V}_x^T\bar{F}_x\delta\boldsymbol{x}\\ &\quad + \bar{V}_x^T\bar{F}_u\delta\boldsymbol{u} + \bar{V}_x^T\bar{F}_v\delta\boldsymbol{v} + \delta\boldsymbol{x}^T\bar{V}_{xx}\bar{F} + \delta\boldsymbol{x}^T\bar{V}_{xx}\bar{F}_x\delta\boldsymbol{x}\\ &\quad + \delta\boldsymbol{x}^T\bar{V}_{xx}\bar{F}_u\delta\boldsymbol{u} + \delta\boldsymbol{x}^T\bar{V}_{xx}\bar{F}_v\delta\boldsymbol{v} + \frac{1}{2}\bar{W}^T\bar{F}\Bigg\}\end{aligned} \tag{A10}$$

Next, note that the term $(1/2)\bar{W}^T\bar{F}$ can be written as follows:

$$\frac{1}{2}\bar{W}^T\bar{F} = \frac{1}{2}\sum_{i=1}^{n}\left(\delta\boldsymbol{x}^T\bar{V}_{xxx}^{(i)}\delta\boldsymbol{x}\bar{F}^{(i)}\right) = \frac{1}{2}\delta\boldsymbol{x}^T\left(\sum_{i=1}^{n}\bar{V}_{xxx}^{(i)}\bar{F}^{(i)}\right)\delta\boldsymbol{x}$$

After equating Eq. (A6) with Eq. (A10), and canceling repeated terms, one obtains

$$
\begin{aligned}
&-\frac{\mathrm{d}\bar{V}}{\mathrm{d}t}-\delta\boldsymbol{x}^T\frac{\mathrm{d}\bar{V}_x}{\mathrm{d}t}-\frac{1}{2}\delta\boldsymbol{x}^T\frac{\mathrm{d}\bar{V}_{xx}}{\mathrm{d}t}\delta\boldsymbol{x}\\
&=\min_{\delta u}\max_{\delta v}\Bigg\{\bar{\mathcal{L}}+\bar{\mathcal{L}}_x^T\delta\boldsymbol{x}+\bar{\mathcal{L}}_u^T\delta\boldsymbol{u}+\bar{\mathcal{L}}_v^T\delta\boldsymbol{v}\\
&+\frac{1}{2}\begin{bmatrix}\delta\boldsymbol{x}\\ \delta\boldsymbol{u}\\ \delta\boldsymbol{v}\end{bmatrix}^T\begin{bmatrix}\bar{\mathcal{L}}_{xx} & \bar{\mathcal{L}}_{xu} & \bar{\mathcal{L}}_{xv}\\ \bar{\mathcal{L}}_{ux} & \bar{\mathcal{L}}_{uu} & \bar{\mathcal{L}}_{uv}\\ \bar{\mathcal{L}}_{vx} & \bar{\mathcal{L}}_{vu} & \bar{\mathcal{L}}_{vv}\end{bmatrix}\begin{bmatrix}\delta\boldsymbol{x}\\ \delta\boldsymbol{u}\\ \delta\boldsymbol{v}\end{bmatrix}\\
&+\bar{V}_x^T\bar{F}_x\delta\boldsymbol{x}+\bar{V}_x^T\bar{F}_u\delta\boldsymbol{u}+\bar{V}_x^T\bar{F}_v\delta\boldsymbol{v}+\delta\boldsymbol{x}^T\bar{V}_{xx}\bar{F}_x\delta\boldsymbol{x}\\
&+\delta\boldsymbol{x}^T\bar{V}_{xx}\bar{F}_u\delta\boldsymbol{u}+\delta\boldsymbol{x}^T\bar{V}_{xx}\bar{F}_v\delta\boldsymbol{v}\Bigg\}\\
&=\min_{\delta u}\max_{\delta v}\Bigg\{\bar{\mathcal{L}}+\delta\boldsymbol{x}^T\bar{Q}_x+\delta\boldsymbol{u}^T\bar{Q}_u+\delta\boldsymbol{v}^T\bar{Q}_v+\frac{1}{2}\delta\boldsymbol{x}^T\bar{Q}_{xx}\delta\boldsymbol{x}\\
&+\frac{1}{2}\delta\boldsymbol{u}^T\bar{Q}_{uu}\delta\boldsymbol{u}+\frac{1}{2}\delta\boldsymbol{v}^T\bar{Q}_{vv}\delta\boldsymbol{v}+\delta\boldsymbol{u}^T\bar{Q}_{ux}\delta\boldsymbol{x}+\delta\boldsymbol{v}^T\bar{Q}_{vx}\delta\boldsymbol{x}\\
&+\delta\boldsymbol{u}^T\bar{Q}_{uv}\delta\boldsymbol{v}\Bigg\}
\end{aligned}
\tag{A11}
$$

where

$$\bar{Q}_x=\bar{F}_x^T\bar{V}_x+\bar{\mathcal{L}}_x \tag{A12a}$$

$$\bar{Q}_u=\bar{F}_u^T\bar{V}_x+\bar{\mathcal{L}}_u \tag{A12b}$$

$$\bar{Q}_v=\bar{F}_v^T\bar{V}_x+\bar{\mathcal{L}}_v \tag{A12c}$$

$$\bar{Q}_{xx}=\bar{\mathcal{L}}_{xx}+\bar{V}_{xx}\bar{F}_x+\bar{F}_x^T\bar{V}_{xx} \tag{A12d}$$

$$\bar{Q}_{uu}=\bar{\mathcal{L}}_{uu} \tag{A12e}$$

$$\bar{Q}_{vv}=\bar{\mathcal{L}}_{vv} \tag{A12f}$$

$$\bar{Q}_{ux}=\bar{F}_u^T\bar{V}_{xx}+\bar{\mathcal{L}}_{ux} \tag{A12g}$$

$$\bar{Q}_{vx}=\bar{F}_v^T\bar{V}_{xx}+\bar{\mathcal{L}}_{vx} \tag{A12h}$$

$$\bar{Q}_{uv}=\bar{\mathcal{L}}_{uv} \tag{A12i}$$

Appendix B: Proof of Proposition II.1

Proof: To find the update law of the value function and its first- and second-order partial derivatives, we substitute the minimizing control (11a) and maximizing control (11b) to the HJBI Eq. (5).

Specifically, we have

$$
\begin{aligned}
&-\frac{\mathrm{d}\bar{V}}{\mathrm{d}t}-\delta\boldsymbol{x}^T\frac{\mathrm{d}\bar{V}_x}{\mathrm{d}t}-\frac{1}{2}\delta\boldsymbol{x}^T\frac{\mathrm{d}\bar{V}_{xx}}{\mathrm{d}t}\delta\boldsymbol{x}=\bar{\mathcal{L}}+\delta\boldsymbol{x}^T\bar{Q}_x+\delta\boldsymbol{u}^{*T}\bar{Q}_u\\
&+\delta\boldsymbol{v}^{*T}\bar{Q}_v+\delta\boldsymbol{u}^{*T}Q_{ux}\delta\boldsymbol{x}+\frac{1}{2}\delta\boldsymbol{u}^{*T}\bar{Q}_{uu}\delta\boldsymbol{u}^*+\delta\boldsymbol{u}^{*T}\bar{Q}_{uv}\delta\boldsymbol{v}^*\\
&+\frac{1}{2}\delta\boldsymbol{v}^{*T}\bar{Q}_{vv}\delta\boldsymbol{v}^*+\delta\boldsymbol{v}^{*T}\bar{Q}_{vx}\delta\boldsymbol{x}+\frac{1}{2}\delta\boldsymbol{x}^T\bar{Q}_{xx}\delta\boldsymbol{x}\\
&=\bar{\mathcal{L}}+\delta\boldsymbol{x}^T\bar{Q}_x+(\boldsymbol{l}_u+\boldsymbol{K}_u\delta\boldsymbol{x})^T\bar{Q}_u+(\boldsymbol{l}_v+\boldsymbol{K}_v\delta\boldsymbol{x})^T\bar{Q}_v\\
&+(\boldsymbol{l}_u+\boldsymbol{K}_u\delta\boldsymbol{x})^T\bar{Q}_{ux}\delta\boldsymbol{x}+(\boldsymbol{l}_v+\boldsymbol{K}_v\delta\boldsymbol{x})^T\bar{Q}_{vx}\delta\boldsymbol{x}\\
&+\frac{1}{2}\delta\boldsymbol{x}^T\bar{Q}_{xx}\delta\boldsymbol{x}+(\boldsymbol{l}_u+\boldsymbol{K}_u\delta\boldsymbol{x})^T\bar{Q}_{uv}(\boldsymbol{l}_v+\boldsymbol{K}_v\delta\boldsymbol{x})\\
&+\frac{1}{2}(\boldsymbol{l}_u+\boldsymbol{K}_u\delta\boldsymbol{x})^T\bar{Q}_{uu}(\boldsymbol{l}_u+\boldsymbol{K}_u\delta\boldsymbol{x})\\
&+\frac{1}{2}(\boldsymbol{l}_v+\boldsymbol{K}_v\delta\boldsymbol{x})^T\bar{Q}_{vv}(\boldsymbol{l}_v+\boldsymbol{K}_v\delta\boldsymbol{x})
\end{aligned}
\tag{B1}
$$

After collecting terms in the right-hand side of Eq. (B1) as zeroth-, first-, and second-order expressions of $\delta\boldsymbol{x}$, we can equate the coefficients of $\delta\boldsymbol{x}$ in the left-hand side and right-hand side of Eq. (B1) and readily obtain the backward propagation equations with respect to the value function and its first- and second-order partial derivatives as follows:

$$-\frac{\mathrm{d}\bar{V}}{\mathrm{d}t}=\bar{\mathcal{L}}+\boldsymbol{l}_u^T\bar{Q}_u+\boldsymbol{l}_v^T\bar{Q}_v+\frac{1}{2}\boldsymbol{l}_u\bar{Q}_{uu}\boldsymbol{l}_u+\boldsymbol{l}_u^T\bar{Q}_{uv}\boldsymbol{l}_v+\frac{1}{2}\boldsymbol{l}_v^T\bar{Q}_{vv}\boldsymbol{l}_v \tag{B2a}$$

$$
\begin{aligned}
-\frac{\mathrm{d}\bar{V}_x}{\mathrm{d}t}&=\bar{Q}_x+\boldsymbol{K}_u^T\bar{Q}_u+\boldsymbol{K}_v^T\bar{Q}_v+\bar{Q}_{ux}^T\boldsymbol{l}_u+\bar{Q}_{vx}^T\boldsymbol{l}_v+\boldsymbol{K}_u^T\bar{Q}_{uu}\boldsymbol{l}_u\\
&+\boldsymbol{K}_u^T\bar{Q}_{uv}\boldsymbol{l}_v+\boldsymbol{K}_v^T\bar{Q}_{vu}\boldsymbol{l}_u+\boldsymbol{K}_v^T\bar{Q}_{vv}\boldsymbol{l}_v
\end{aligned}
\tag{B2b}
$$

$$
\begin{aligned}
-\frac{\mathrm{d}\bar{V}_{xx}}{\mathrm{d}t}&=\boldsymbol{K}_u^T\bar{Q}_{ux}+\bar{Q}_{ux}^T\boldsymbol{K}_u+\boldsymbol{K}_v^T\bar{Q}_{vx}+\bar{Q}_{vx}^T\boldsymbol{K}_v+\boldsymbol{K}_v^T\bar{Q}_{vu}\boldsymbol{K}_u\\
&+\boldsymbol{K}_u^T\bar{Q}_{uv}\boldsymbol{K}_v+\boldsymbol{K}_u^T\bar{Q}_{uu}\boldsymbol{K}_u+\boldsymbol{K}_v^T\bar{Q}_{vv}\boldsymbol{K}_v+\bar{Q}_{xx}
\end{aligned}
\tag{B3c}
$$

At the final time, we have the boundary condition (6). By taking the Taylor series expansion around $\bar{\boldsymbol{x}}(t_f)$ we get the expressions

$$
\begin{aligned}
\phi(\boldsymbol{x}(t_f),t_f)&=\phi(\bar{\boldsymbol{x}}(t_f)+\delta\boldsymbol{x}(t_f),t_f)\\
&\approx\phi(\bar{\boldsymbol{x}}(t_f),t_f)+\delta\boldsymbol{x}(t_f)^T\phi_x(\bar{\boldsymbol{x}}(t_f),t_f)\\
&+\delta\boldsymbol{x}(t_f)^T\phi_{xx}(\bar{\boldsymbol{x}}(t_f),t_f)\delta\boldsymbol{x}(t_f)
\end{aligned}
\tag{B4}
$$

Therefore, the boundary conditions at $t=t_f$ for the backward differential equations (19) are

$$\bar{V}(t_f)=\phi(\bar{\boldsymbol{x}}(t_f),t_f) \tag{B5a}$$

$$\bar{V}_x(t_f)=\phi_x(\bar{\boldsymbol{x}}(t_f),t_f) \tag{B5b}$$

$$\bar{V}_{xx}(t_f)=\phi_{xx}(\bar{\boldsymbol{x}}(t_f),t_f) \tag{B5c}$$

Acknowledgments

Support for this work was provided by NSF awards CMMI-1160780 and CMMI-1662542 and AFOSR award FA9550-13-1-0029.

References

[1] Zhou, K., Doyle, J. C., and Glover, K., *Robust and Optimal Control*, Vol. 40, Prentice Hall, New Jersey, 1996, Chap. 16.

[2] Morimoto, J., and Doya, K., "Robust Reinforcement Learning," *Neural Computation*, Vol. 17, No. 2, 2005, pp. 335–359. doi:10.1162/0899766053011528

[3] Mihatsch, O., and Neuneier, R., "Risk-Sensitive Reinforcement Learning," *Machine Learning*, Vol. 49, Nos. 2–3, 2002, pp. 267–290. doi:10.1023/A:1017940631555

[4] Coraluppi, S. P., and Marcus, S. I., "Risk-Sensitive and Minimax Control of Discrete-Time, Finite-State Markov Decision Processes," *Automatica*, Vol. 35, No. 2, 1999, pp. 301–309. doi:10.1016/S0005-1098(98)00153-8

[5] Morimioto, J., Zeglin, G., and Atkeson, C. G., "Minimax Differential Dynamic Programming: Application to a Biped Walking Robot," *2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE Publ., Piscataway, NJ, Vol. 2, 2003, pp. 1927–1932. doi:10.1109/IROS.2003.1248926

[6] Mayne, D., "A Second-Order Gradient Method for Determining Optimal Trajectories of Non-Linear Discrete-Time Systems," *International Journal of Control*, Vol. 3, No. 1, 1966, pp. 85–95. doi:10.1080/00207176608921369

[7] Jacobson, D. H., and Mayne, D. Q., *Differential Dynamic Programming*, American Elsevier Pub. Co., New York, 1970, Chap. 2.
[8] Liao, L. Z., and Shoemaker, C. A., "Convergence in Unconstrained Discrete-Time Differential Dynamic Programming," *IEEE Transactions on Automatic Control*, Vol. 36, No. 6, 1991, pp. 692–706. doi:10.1109/9.86943
[9] Ng, C. K., Liao, L.-Z., and Li, D., "A Globally Convergent and Efficient Method for Unconstrained Discrete-Time Optimal Control," *Journal of Global Optimization*, Vol. 23, Nos. 3–4, 2002, pp. 401–421. doi:10.1023/A:1016595100139
[10] Todorov, E., and Li, W., "Optimal Control Methods Suitable for Biomechanical Systems," *Proceedings of the 25th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Vol. 2, IEEE Publ., Piscataway, NJ, Sept. 2003, pp. 1758–1761. doi:10.1109/IEMBS.2003.1279748
[11] Pantoja, J., "Differential Dynamic Programming and Newton's Method," *International Journal of Control*, Vol. 47, No. 5, 1988, pp. 1539–1553. doi:10.1080/00207178808906114
[12] Atkeson, C. G., and Morimoto, J., "Nonparametric Representation of Policies and Value Functions: A Trajectory-Based Approach," *Advances in Neural Information Processing Systems*, MIT Press, Cambridge, MA, 2003, pp. 1643–1650.
[13] Abbeel, P., Coates, A., Quigley, M., and Ng, A. Y., "An Application of Reinforcement Learning to Aerobatic Helicopter Flight," *Advances in Neural Information Processing Systems*, Vol. 19, MIT Press, Vancouver, Dec. 2007, pp. 1–8.
[14] Todorov, E., and Li, W., "A Generalized Iterative LQG Method for Locally-Optimal Feedback Control of Constrained Nonlinear Stochastic Systems," *Proceedings of the 2005 American Control Conference*, IEEE Publ., Piscataway, NJ, June 2005, pp. 300–306. doi:10.1109/ACC.2005.1469949
[15] Theodorou, E., Tassa, Y., and Todorov, E., "Stochastic Differential Dynamic Programming," *Proceedings of the 2010 American Control Conference*, IEEE Publ., Piscataway, NJ, 2010, pp. 1125–1132. doi:10.1109/ACC.2010.5530971
[16] Tassa, Y., Erez, T., and Smart, W. D., "Receding Horizon Differential Dynamic Programming," *Advances in Neural Information Processing Systems*, Curran Associates, Inc., Red Hook, NY, Dec. 2007, pp. 1465–1472.
[17] Tassa, Y., Erez, T., and Todorov, E., "Synthesis and Stabilization of Complex Behaviors Through Online Trajectory Optimization," *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE Publ., Piscataway, NJ, Oct. 2012, pp. 4906–4913. doi:10.1109/IROS.2012.6386025
[18] Atkeson, C., and Stephens, B., "Random Sampling of States in Dynamic Programming," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, Vol. 38, No. 4, 2008, pp. 924–929. doi:10.1109/TSMCB.2008.926610
[19] Erez, T., Tassa, Y., and Todorov, E., "Infinite-Horizon Model Predictive Control for Periodic Tasks with Contacts," *Proceedings of Robotics: Science and Systems VII*, MIT Press, Cambridge, MA, 2012, pp. 73–80. doi:10.15607/RSS.2011.VII.010
[20] Tassa, Y., Mansard, N., and Todorov, E., "Control-Limited Differential Dynamic Programming," *2014 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE Publ., Piscataway, NJ, May–June 2014, pp. 1168–1175. doi:10.1109/ICRA.2014.6907001
[21] Lin, T. C., and Arora, J. S., "Differential Dynamic Programming Technique for Constrained Optimal Control," *Computational Mechanics*, Vol. 9, No. 1, 1991, pp. 27–40. doi:10.1007/BF00369913
[22] Mitrovic, D., Klanke, S., and Vijayakumar, S., "Adaptive Optimal Feedback Control with Learned Internal Dynamics Models," *From Motor Learning to Interaction Learning in Robots*, Springer, Berlin, 2010, pp. 65–84. doi:10.1007/978-3-642-05181-4_4
[23] Bertsekas, D. P., and Tsitsiklis, J. N., "Neuro-Dynamic Programming: An Overview," *Proceedings of the 34th IEEE Conference on Decision and Control*, Vol. 1, IEEE Publ., Piscataway, NJ, Dec. 1995, pp. 560–564. doi:10.1109/CDC.1995.478953
[24] Pan, Y., and Theodorou, E., "Probabilistic Differential Dynamic Programming," *Advances in Neural Information Processing Systems*, Curran Associates, Inc., Red Hook, NY, Dec. 2014, pp. 1907–1915.
[25] Sun, W., Theodorou, E. A., and Tsiotras, P., "Game Theoretic Continuous Time Differential Dynamic Programming," *American Control Conference*, IEEE Publ., Piscataway, NJ, July 2015, pp. 5593–5598. doi:10.1109/ACC.2015.7172215
[26] Sun, W., Theodorou, E. A., and Tsiotras, P., "Continuous-Time Differential Dynamic Programming with Terminal Constraints," *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, IEEE Publ., Piscataway, NJ, Dec. 2014, pp. 1–6. doi:10.1109/ADPRL.2014.7010647
[27] Denham, W., "Differential Dynamic Programming," *IEEE Transactions on Automatic Control*, Vol. 16, No. 4, 1971, pp. 389–390. doi:10.1109/TAC.1971.1099760
[28] Isaacs, R., *Differential Games: A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization*, Courier Dover Publ., Mineola, NY, 1999, Chap. 2.
[29] Başar, T., and Olsder, G. J., *Dynamic Noncooperative Game Theory*, Vol. 23, SIAM, Philadelphia, PA, 1999, p. 300, Chap. 6.
[30] Crandall, M. G., and Lions, P.-L., "Viscosity Solutions of Hamilton-Jacobi Equations," *Transactions of the American Mathematical Society*, Vol. 277, No. 1, 1983, pp. 1–42. doi:10.1090/S0002-9947-1983-0690039-8
[31] Bardi, M., and Capuzzo-Dolcetta, I., *Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman Equations*, Springer Science & Business Media, Berlin, 2008, Chap. 2.
[32] Melikyan, A., *Generalized Characteristics of First Order PDEs: Applications in Optimal Control and Differential Games*, Springer Science & Business Media, Berlin, 2012, Chap. 2.
[33] Michael, N., Mellinger, D., Lindsey, Q., and Kumar, V., "The GRASP Multiple Micro-UAV Testbed," *IEEE Robotics & Automation Magazine*, Vol. 17, No. 3, 2010, pp. 56–65. doi:10.1109/MRA.2010.937855
[34] Bangura, M., and Mahony, R., "Real-Time Model Predictive Control for Quadrotors," *Proceedings of the 19th World Congress*, The International Federation of Automatic Control, Laxenburg, Austria, Vol. 47, Aug. 2014, pp. 11773–11780. doi:10.3182/20140824-6-ZA-1003.00203
[35] AutonomyLab, "Ardrone_Autonomy," Dec. 2016, https://github.com/AutonomyLab/ardrone_autonomy [accessed 06 June 2018].
[36] KumarRobotics, "Motion_Capture_System," March 2015, https://github.com/KumarRobotics/motion_capture_system [accessed 06 June 2018].
[37] Furrer, F., Burri, M., Achtelik, M., and Siegwart, R., "Robot Operating System (ROS): The Complete Reference (Volume 1)," *Studies Comp. Intelligence Volume Number:625*, Springer International Publ., Cham, Switzerland, 2016, Chap. 23.
[38] Ferrin, J., Leishman, R., Beard, R., and McLain, T., "Differential Flatness Based Control of a Rotorcraft for Aggressive Maneuvers," *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE Publ., Piscataway, NJ, Sept. 2011, pp. 2688–2693. doi:10.1109/IROS.2011.6095098