Fast Approximate Dynamic Programming for Input-Affine Dynamics

M. A. S. Kolarijani and P. Mohajerin Esfahani

Abstract. We propose two novel numerical schemes for approximate implementation of the dynamic programming (DP) operation concerned with finite-horizon optimal control of discrete-time systems with input-affine dynamics. The proposed algorithms involve discretization of the state and input spaces, and are based on an alternative path that solves the dual problem corresponding to the DP operation. We provide error bounds for the proposed algorithms, along with a detailed analyses of their computational complexity. In particular, for a specific class of problems with separable data in the state and input variables, the proposed approach can reduce the typical time complexity of the DP operation from $\mathcal{O}(XU)$ to $\mathcal{O}(X+U)$, where X and U denote the size of the discrete state and input spaces, respectively. This reduction in complexity is achieved by an algorithmic transformation of the minimization in DP operation to an addition via discrete conjugation.

Keywords: approximate dynamic programming, conjugate duality, input-affine dynamics, computational complexity

1. Introduction

Dynamic programming (DP) is one of the most common tools used for tackling sequential decision with applications in, e.g., optimal control, operation research, and machine learning. The basic idea of DP is to solve the Bellman equation

$$J_t(x_t) = \min_{u_t} \big\{ C(x_t, u_t) + J_{t+1}(x_{t+1}) \big\}, \tag{1}$$

backward in time t for the costs-to-go J_t, where $C(x_t, u_t)$ is the cost of taking the control action u_t at the state x_t (value iteration). Arguably, the most important drawback of DP is in its high computational cost in solving problems with a large scale *finite* state space, which are usually described as Markov decision processes (MDPs). Indeed, in [4], the authors show that for a finite-horizon MDP, the problem of determining whether a control action u_0 is an optimal action at a given initial state x_0 using value iteration is EXPTIME-complete. For problems with a *continuous* state space, which is commonly the case in engineering applications, solving the Bellman equation requires solving infinite number of optimization problems. This usually renders the exact implementation of the DP operation impossible, except for a few cases with an available closed form solution, e.g., linear quadratic regulator [7, Sec. 4.1]. To address this issue, various schemes have been introduced,

---

Date: May 11, 2021.

The authors are with Delft Center for Systems and Control, Delft University of Technology, Delft, The Netherlands. Email: {M.A.SharifiKolarijani, P.MohajerinEsfahani}@tudelft.nl.

The authors would like to thank G. F. Max for several fruitful discussions.

This research was supported by the European Research Council (ERC) under the European Unions Horizon 2020 research and innovation programme (TRUST-949796).

commonly known as *approximate* dynamic programming; see, e.g., [9, 29]. A common scheme is to use a sample-based approach accompanied by some form of function approximation. This usually amounts to deploying a brute force search over the discretizations/abstractions of the state and input spaces, leading to a time complexity of at least $\mathcal{O}(XU)$, where X and U are the cardinality of the discrete state and input spaces, respectively.

For some DP problems, it is possible to reduce this complexity by using duality, i.e., approaching the minimization problem in (1) in the conjugate domain. For instance, for the deterministic linear dynamics $x_{t+1} = Ax_t + Bu_t$ with the separable cost $C(x_t, u_t) = C_{\mathrm{s}}(x_t) + C_{\mathrm{i}}(u_t)$, we have

$$J_t(x_t) \geq C_{\mathrm{s}}(x_t) + \left[C_{\mathrm{i}}^*(-B^\top \cdot) + J_{t+1}^*\right]^* (Ax_t), \tag{2}$$

where the operator $[\cdot]^*$ denotes the Legendre-Fenchel transform, also known as (convex) conjugate transform. Under some technical assumptions (including, among others, convexity of the functions C_{i} and J_{t+1}), we have equality in (2); see, e.g., [8, Prop. 5.3.1]. Notice how the minimization operator in (1) transforms to a simple addition in (2). This observation signals the possibility of a significant reduction in the time complexity of solving the Bellman equation, at least for particular classes of DP problems.

Approaching the DP problem through the lens of the conjugate duality goes back to Bellman [5]. Further applications of this idea for reducing the computational complexity were later explored in [16] and [20]. Fundamentally, these approaches exploit the operational duality of infimal convolution and addition with respect to (w.r.t.) the conjugate transform [30]: For two functions $f_1, f_2 : \mathbb{R}^n \to [-\infty, \infty]$, we have $(f_1 \Box f_2)^* = f_1^* + f_2^*$, where $f_1 \Box f_2(x) := \inf\{f_1(x_1) + f_2(x_2) : x_1 + x_2 = x\}$ is the infimal convolution of f_1 and f_2. This is analogous to the well-known operational duality of convolution and multiplication w.r.t. the Fourier transform. Actually, Legendre-Fenchel transform plays a similar role as Fourier transform when the underlying algebra is the max-plus algebra, as opposed to the conventional plus-times algebra. Much like the extensive application of the latter operational duality upon introduction of the fast Fourier transform, "fast" numerical algorithms for conjugate transform can facilitate efficient applications of the former one. Interestingly, the first fast algorithm for computing (discrete) conjugate functions, known as fast Legendre transform, was inspired by fast Fourier transform, and enjoys the same *log-linear* complexity in the number of data points; see [13, 22] and the references therein. Later, this complexity was reduced by introducing a *linear-time* algorithm known as linear-time Legendre transform (LLT) [23]. We refer the interested reader to [25] for an extensive review of these algorithms (and other similar algorithms) and their applications. In this regard, we also note that recently, in [34], the authors introduced a quantum algorithm for computing the (discrete) conjugate of convex functions, which achieves a *poly-logarithmic* time complexity in the number of data points.

One of the first and most widespread applications of these fast algorithms has been in solving Hamilton-Jacobi equation [1, 13, 14]. Another interesting area of application is image processing, where the Legendre-Fenchel transform is commonly known as distance transform [17, 24]. Recently, in [18], the authors used these algorithms to tackle the optimal transport problem with strictly convex costs, with applications in image processing and in numerical methods for solving partial differential equations. However, surprisingly, the application of these fast algorithms in solving discrete-time optimal control problems seems to remain largely unexplored. An exception is [12], where the authors use LLT to propose the "fast value iteration" algorithm for computing the fixed-point of the Bellman operator arising from a specific class of infinite-horizon, discrete-time DP problems. Indeed, the setup in [12] corresponds to a subclass of problems considered in our study, which allows for a "perfect" transformation of the minimization in the DP operation in the primal domain to an addition in the dual (conjugate) domain; this connection will be discussed in detail in Section 7.1. Let us also note that the algorithms developed in [17, 24] for distance transform can also potentially tackle the (discretized) optimal control problems similar to the ones considered in this study. In particular, these algorithms require the stage cost to be reformulated as a convex distance function of the current and next states. While the this property might arise naturally, it can generally be restrictive as it is in our case.

Another line of work, closely related to ours, involves algorithms that utilize max-plus algebra in solving, continuous-time, continuous-space, deterministic optimal control problems; see, e.g., [27, 26, 2]. These works exploit the compatibility of the Bellman operation with max-plus operations, and approximate the value function as a max-plus linear combination. In particular, recently in [3, 6], the authors used this idea to propose an approximate value iteration algorithm for deteministic MDPs with continuous state space. In this regard, we note that the proposed algorithms in the current study also implicitly involve representing cost functions as max-plus linear combinations, yielding piece-wise affine approximations. The key difference of the proposed algorithms is however to choose a dynamic, grid-like (factorized) set of slopes in the dual space in order to control the error and reduce the computational cost; we will discuss this point in more details in Section 7.2

Paper organization and summary of main results. In this study, we consider the approximate implementation of the DP operation arising in the finite-horizon optimal control of discrete-time systems with continuous state and input spaces. The proposed approach involves discretization of the state space, and is based on an alternative path that solves the dual problem corresponding to the DP operation by utilizing the LLT algorithm for discrete conjugation. After presenting some preliminaries in Section 2, we provide the problem statement and its standard solution via the d-DP algorithm (in the primal domain) in Section 3. Sections 4 and 5 contain our main results on the proposed alternative approach for solving the DP problem in the conjugate domain:

(i) **From minimization in primal domain to addition in dual domain:** In Section 4, we introduce the discrete conjugate DP (d-CDP) algorithm (Algorithm 1) for problems with deterministic *input-affine* dynamics; see Figure 1a for the sketch of the algorithm. In particular, we use the linearity of the dynamics in the input to effectively incorporate the operational duality of addition and infimal convolution, and transform the minimization in the DP operation to a simple addition at the expense of three conjugate transforms. This, in turn, leads to transferring the computational cost from the input domain $\mathbb{U}$ to the state dual domain $\mathbb{Y}$ (Theorem 4.4). Moreover, the extension of this algorithm for stochastic dynamics is discussed in Section 4.3.1.

(ii) **From quadratic to linear time complexity:** In Section 5, we modify the proposed d-CDP algorithm (Algorithm 2) and reduce its time complexity (Theorem 5.2) for a subclass of problems with *separable* data in the state and input variables; see Figure 1b for the sketch of the algorithm. In particular, for this class, the time complexity of computing the costs-to-go at each step is of $\mathcal{O}(X + U)$, compared to the standard complexity of $\mathcal{O}(XU)$.

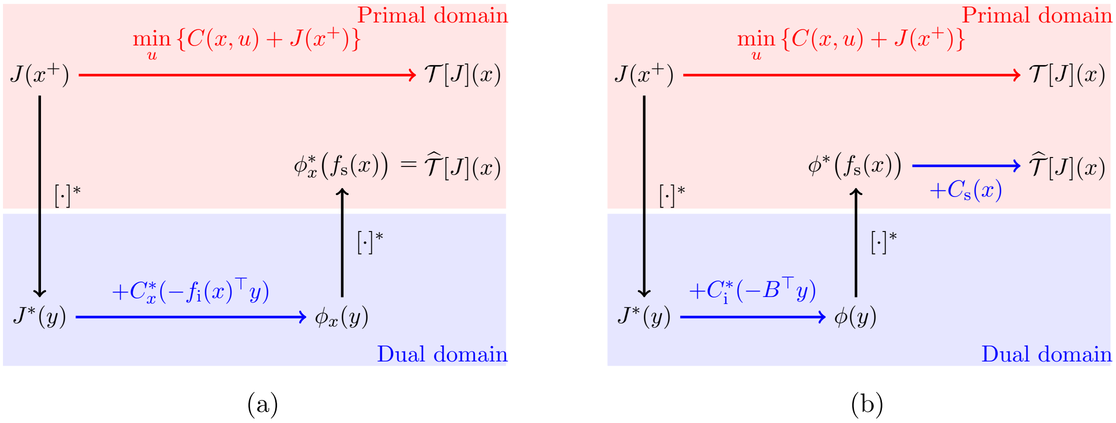


FIGURE 1. Sketch of the proposed algorithms – the standard DP operation in the primal domain (upper red paths) and the conjugate DP (CDP) operation through the dual domain (bottom blue paths): (a) Setting 1 with dynamics $x^+ = f_s(x) + f_i(x) \cdot u$ and generic convex cost $C(x,u)$; (b) Setting 2 with dynamics $x^+ = f_s(x) + B \cdot u$ and separable convex cost $C(x,u) = C_s(x) + C_i(u)$.

(iii) **Error bounds and construction of discrete dual domain:** We analyze the error of the proposed d-CDP algorithm and its modification (Theorems 4.6 and 5.3). The error analysis is based on two preliminary results on the error of discrete conjugation (Lemma 2.5) and approximate conjugation (Lemma 2.6 and Corollary 2.7). Moreover, we use the results of our error analysis to provide concrete guidelines for the construction of a dynamic discrete dual space in the proposed algorithms (Remark 4.7).

In Section 6, we validate our theoretical results and compare the performance of the proposed algorithms with the benchmark d-DP algorithm through multiple numerical examples. Further numerical examples (and descriptions of the extensions of the proposed algorithms) are provided in Appendix C. Moreover, in order to facilitate the application of the proposed algorithms, we provide a MATLAB package:

(iv) **The d-CDP MATLAB package:** The algorithms presented in this study and their extensions are available in the d-CDP MATLAB package [21]. A brief description of this package is provided in Appendix D. The numerical examples of this study are also included in the package and reproducible.

Section 7 concludes the paper by providing further remarks on the proposed algorithms such as its limitations and its relation to the existing schemes and algorithms in the literature. In particular, we discuss a potential significance of the conjugate dynamic programming framework proposed in this study towards quantum-mechanical implementation of DP:

(v) **Towards quantum dynamic programming:** Motivated by the recent quantum speedup for discrete conjugation [34], we envision that the proposed d-CDP Algorithm 2 paves the way for developing a quantum DP algorithm. In Section 7.6, we discuss how such an algorithm can potentially address the infamous "curse of dimensinality" in DP.

2. Notations and Preliminaries

2.1. General notations

We use $\mathbb{R}$ to denote the real line and $\overline{\mathbb{R}} = \mathbb{R} \cup \{\infty\}$, $\underline{\overline{\mathbb{R}}} = \mathbb{R} \cup \{\pm\infty\}$ to denote its extensions. The standard inner product in $\mathbb{R}^n$ and the corresponding induced 2-norm are denoted by $\langle \cdot, \cdot \rangle$ and $\|\cdot\|$, respectively. We also use $\|\cdot\|$ to denote the operator norm (w.r.t. the 2-norm) of a matrix; i.e., for $A \in \mathbb{R}^{m\times n}$, we denote $\|A\| = \sup\{\|Ax\| : \|x\| = 1\}$. We use the common convention in optimization whereby the optimal value of an infeasible minimization (resp. maximization) problem is set to $+\infty$ (resp. $-\infty$). Arbitrary sets (finite/infinite, countable/uncountable) are denoted as $\mathbb{X}, \mathbb{Y}, \ldots$. For *finite* (discrete) sets, we use the superscript d as in $\mathbb{X}^{\mathrm{d}}, \mathbb{Y}^{\mathrm{d}}, \ldots$ to differentiate them form infinite sets. Moreover, we use the superscript g to differentiate *grid-like* (factorized) finite sets. Precisely, a grid $\mathbb{X}^{\mathrm{g}} \subset \mathbb{R}^n$ is the Cartesian product $\mathbb{X}^{\mathrm{g}} = \Pi_{i=1}^{n} \mathbb{X}_i^{\mathrm{g}} = \mathbb{X}_1^{\mathrm{g}} \times \ldots \times \mathbb{X}_n^{\mathrm{g}}$, where $\mathbb{X}_i^{\mathrm{g}}$ is a finite set of real numbers $x_i^1 < x_i^2 < \ldots < x_i^{X_i}$. Assuming $X_i \geq 3$ for all $i = 1, \ldots, n$, we define $\mathbb{X}_{\mathrm{sub}}^{\mathrm{g}} := \Pi_{i=1}^{n} \mathbb{X}_{\mathrm{sub}i}^{\mathrm{g}}$, where $\mathbb{X}_{\mathrm{sub}i}^{\mathrm{g}} = \mathbb{X}_i^{\mathrm{g}} \setminus \{x_i^1, x_i^{X_i}\}$; that is, $\mathbb{X}_{\mathrm{sub}}^{\mathrm{g}}$ is the *sub-grid* derived by omitting the smallest and largest elements of $\mathbb{X}^{\mathrm{g}}$ in each dimension. The cardinality of a finite set $\mathbb{X}^{\mathrm{d}}$ (or $\mathbb{X}^{\mathrm{g}}$) is denoted by X. Let $\mathbb{X}, \mathbb{Y}$ be two arbitrary sets in $\mathbb{R}^n$. The convex hull of $\mathbb{X}$ is denoted by $\mathrm{co}(\mathbb{X})$. The diameter of $\mathbb{X}$ is defined as $\Delta_{\mathbb{X}} := \sup_{x,y\in\mathbb{X}} \|x - y\|$. We use $\mathrm{d}(\mathbb{X}, \mathbb{Y}) := \inf_{x\in\mathbb{X}, y\in\mathbb{Y}} \|x - y\|$ to denote the distance between $\mathbb{X}$ and $\mathbb{Y}$. The one-sided Hausdorff distance *from* $\mathbb{X}$ *to* $\mathbb{Y}$ is defined as $\mathrm{d}_{\mathrm{H}}(\mathbb{X}, \mathbb{Y}) := \sup_{x\in\mathbb{X}} \inf_{y\in\mathbb{Y}} \|x - y\|$. For an extended real-valued function $h : \mathbb{R}^n \to \overline{\mathbb{R}}$, the effective domain of h is defined by $\mathrm{dom}(h) := \{x \in \mathbb{R}^n : h(x) < +\infty\}$. The Lipschtiz constant of h over a set $\mathbb{X} \subset \mathrm{dom}(h)$ is denoted by $\mathrm{L}(h; \mathbb{X}) := \sup_{x,y\in\mathbb{X}} |h(x) - h(y)| / \|x - y\|$. We also denote $\mathrm{L}(h) := \mathrm{L}\left(h; \mathrm{dom}(h)\right)$ and $\mathbb{L}(h) := \Pi_{i=1}^{n} \left[\mathrm{L}_i^-(h), \mathrm{L}_i^+(h)\right]$, where $\mathrm{L}_i^+(h)$ (resp. $\mathrm{L}_i^-(h)$) is the maximum (resp. minimum) slope of the function h along the i-th dimension, i.e.,

$$\mathrm{L}_i^+(h) := \sup\left\{\frac{h(x) - h(y)}{x_i - y_i} : x, y \in \mathrm{dom}(h),\ x_i > y_i,\ x_j = y_j\ (j \neq i)\right\},$$

$$\mathrm{L}_i^-(h) := \inf\left\{\frac{h(x) - h(y)}{x_i - y_i} : x, y \in \mathrm{dom}(h),\ x_i > y_i,\ x_j = y_j\ (j \neq i)\right\}.$$

The subdifferential of h at a point $x \in \mathbb{R}^n$ is defined as $\partial h(x) := \big\{y \in \mathbb{R}^n : h(\tilde{x}) \geq h(x) + \langle y, \tilde{x} - x\rangle, \forall \tilde{x} \in \mathrm{dom}(h)\big\}$. We report the complexities using the standard big O notations $\mathcal{O}$ and $\widetilde{\mathcal{O}}$, where the latter hides the logarithmic factors. In this study, we are mainly concerned with the dependence of the computational complexities on *the size of the finite sets* involved (discretization of the primal and dual domains). In particular, we ignore the possible dependence of the computational complexities on the dimension of the variables, unless they appear in the power of the size of those discrete sets; e.g., the complexity of a single evaluation of an analytically available function is taken to be of $\mathcal{O}(1)$, regardless of the dimension of its input and output arguments. For reader's convenience, we also provide the list of the most important objects used throughout this article in the Table 1.

2.2. Extension of discrete functions

Consider an extended real-valued function $h : \mathbb{R}^n \to \overline{\mathbb{R}}$, and its discretization $h^{\mathrm{d}} : \mathbb{X}^{\mathrm{d}} \to \overline{\mathbb{R}}$, where $\mathbb{X}^{\mathrm{d}}$ is a finite subset of $\mathbb{R}^n$. We use the superscript d, as in h^{d}, to denote the discretization of h. We particularly use this notation in combination with a second operation to emphasize that

TABLE 1. List of the most important notational conventions.

Notation & Description		Definition
LERP	Multilinear interpolation & extrapolation	–
LLT	Linear-time Legendre Transform	–
h^{d}	Discretization of the function h	–
$\widetilde{h^{\mathrm{d}}}$	Extension of the discrete function h^{d}	–
$\overline{h^{\mathrm{d}}}$	LERP extension of the discrete function h^{d} (with grid-like domain)	–
h^*	Conjugate of h	(3)
$h^{\mathrm{d}*}$	Discrete conjugate of h (conjugate of h^{d})	(4)
h^{**}	Biconjugate of h	(5)
$h^{\mathrm{d}*\mathrm{d}*}$	Discrete biconjugate of h	(6)
$\mathcal{T}$	Dynamic Programming (DP) operator	(16) & (29)
$\mathcal{T}^{\mathrm{d}}$	Discrete DP (d-DP) operator [for Setting 1]	(18)
$\widehat{\mathcal{T}}$	Conjugate DP (CDP) operator [for Setting 1]	(23)
$\widehat{\mathcal{T}}^{\mathrm{d}}$	Discrete CDP (d-CDP) operator	(24) & (30)
$\widehat{\mathcal{T}}^{\mathrm{d}}_{\mathrm{m}}$	Modified d-CDP operator [for Setting 2]	(31)

the second operation is applied on the discretized version of the operand. In particular, we use $\widetilde{h^{\mathrm{d}}} : \mathbb{R}^n \to \overline{\mathbb{R}}$ to denote the extension of the discrete function $h^{\mathrm{d}} : \mathbb{X}^{\mathrm{d}} \to \overline{\mathbb{R}}$. The extension can be realized, for example, as a linear approximation of the form $\widetilde{J^{\mathrm{d}}}(x) = \sum_{i=1}^{B} \alpha_i \cdot b_i(x)$, where b_i's are the basis functions, and α_i are the corresponding coefficients. These coefficients are determined by minimizing the squared error of the approximation over the discrete space, i.e., $\sum_{x\in\mathbb{X}^{\mathrm{d}}}[J^{\mathrm{d}}(x) - \widetilde{J^{\mathrm{d}}}(x)]^2$. Another possibility is to use kernel-based approximators with one kernel per sample, i.e., $\widetilde{J^{\mathrm{d}}}(x) = \sum_{y\in\mathbb{X}^{\mathrm{d}}} \alpha_y \cdot r(x, y)$, where r is the kernel function.

Remark 2.1 (Complexity of extension operation)**.** *We use E to denote the complexity of a generic extension operator. That is, for each $x \in \mathbb{R}^n$, the time complexity of the single evaluation $\widetilde{h^{\mathrm{d}}}(x)$ is assumed to be of $\mathcal{O}(E)$, with E (possibly) being a function of X.*

A kernel-based approximator of interest in this study is the *multilinear interpolation & extrapolation* (LERP) of a discrete function with a *grid-like* domain. Hence, we denote this operation with the different notation $\overline{h^{\mathrm{d}}} : \mathbb{R}^n \to \overline{\mathbb{R}}$ for the discrete function $h^{\mathrm{d}} : \mathbb{X}^{\mathrm{g}} \to \overline{\mathbb{R}}$. Notice that the LERP extension preserves the value of the function at the discrete points, i.e, $\overline{h^{\mathrm{d}}}(x) = h^{\mathrm{d}}(x)$ for all $x \in \mathbb{X}^{\mathrm{g}}$. In order to facilitate our complexity analysis in subsequent sections, we discusses the computational complexity of LERP in the following remark.

Remark 2.2 (Complexity of LERP)**.** *Given a discrete function $h^{\mathrm{d}} : \mathbb{X}^{\mathrm{g}} \to \mathbb{R}$ with a grid-like domain $\mathbb{X}^{\mathrm{g}} \subset \mathbb{R}^n$, the time complexity of a single evaluation of the LERP extension $\overline{h^{\mathrm{d}}}$ at a point $x \in \mathbb{R}^n$ is of $\mathcal{O}(2^n + \log X) = \widetilde{\mathcal{O}}(1)$ if $\mathbb{X}^{\mathrm{g}}$ is non-uniform, and of $\mathcal{O}(2^n) = \mathcal{O}(1)$ if $\mathbb{X}^{\mathrm{g}}$ is uniform. To see this, note that, in the case $\mathbb{X}^{\mathrm{g}}$ is non-uniform, LERP requires $\mathcal{O}(\log X)$ operations to find the position of x w.r.t. the grid points, using binary search. If $\mathbb{X}^{\mathrm{g}}$ is a uniform grid, this can be done in $\mathcal{O}(n)$ time. Upon finding the position of x, LERP then involves a series of one-dimensional linear interpolations or extrapolations along each dimension, which takes $\mathcal{O}(2^n)$ operations.*

For a convex function $h : \mathbb{R}^n \to \overline{\mathbb{R}}$, we have $\partial h(x) \neq \emptyset$ for all x in the relative interior of $\mathbb{X}$ [8, Prop. 5.4.1]. This characterization of convexity can be extended to discrete functions. A discrete function $h^{\mathrm{d}} : \mathbb{X}^{\mathrm{d}} \to \mathbb{R}$ is called convex-extensible if $\partial h^{\mathrm{d}}(x) \neq \emptyset$ for all $x \in \mathrm{dom}(h) = \mathbb{X}^{\mathrm{d}}$. Equivalently, h^{d} is convex-extensible, if it can be extended to a convex function $\widetilde{h^{\mathrm{d}}} : \mathbb{R}^n \to \overline{\mathbb{R}}$ such that $\widetilde{h^{\mathrm{d}}}(x) = h^{\mathrm{d}}(x)$ for all $x \in \mathbb{X}^{\mathrm{d}}$; we refer the reader to, e.g., [28] for different extensions of the notion of convexity to discrete functions.

2.3. Legendre-Fenchel Transform

Consider an extended-real-valued function $h : \mathbb{R}^n \to \overline{\mathbb{R}}$, with a nonempty effective domain $\mathrm{dom}(h) = \mathbb{X}$. The Legendre-Fenchel transform (convex conjugate) of h is the function

$$h^* : \mathbb{R}^n \to \overline{\mathbb{R}} : y \mapsto \sup_{x \in \mathbb{X}} \left\{ \langle y, x \rangle - h(x) \right\}. \tag{3}$$

Note that the conjugate function h^* is convex by construction. In this study, we particularly consider *discrete* conjugation, which involves computing the conjugate function using the discretized version $h^{\mathrm{d}} : \mathbb{X}^{\mathrm{d}} \to \overline{\mathbb{R}}$ of the function h, where $\mathbb{X}^{\mathrm{d}} \cap \mathbb{X} \neq \emptyset$. We use the notation $[\cdot]^{\mathrm{d}*}$, as opposed the standard notation $[\cdot]^*$, for discrete conjugation; that is,

$$h^{\mathrm{d}*} = [h^{\mathrm{d}}]^* : \mathbb{R}^n \to \mathbb{R} : y \mapsto \max_{x \in \mathbb{X}^{\mathrm{d}}} \left\{ \langle y, x \rangle - h(x) \right\}. \tag{4}$$

The biconjugate of h is the function

$$h^{**} = [h^*]^* : \mathbb{R}^n \to \overline{\mathbb{R}} : x \mapsto \sup_{y \in \mathbb{R}^n} \left\{ \langle x, y \rangle - h^*(y) \right\} = \sup_{y \in \mathbb{R}^n} \inf_{z \in \mathbb{X}} \left\{ \langle x - z, y \rangle + h(z) \right\}. \tag{5}$$

Using the notion of discrete conjugation $[\cdot]^{\mathrm{d}*}$, we also define the (doubly) discrete biconjugate

$$h^{\mathrm{d}*\mathrm{d}*} = [h^{\mathrm{d}*}]^{\mathrm{d}*} : \mathbb{R}^n \to \mathbb{R} : x \mapsto \max_{y \in \mathbb{Y}^{\mathrm{d}}} \left\{ \langle x, y \rangle - h^{\mathrm{d}*}(y) \right\} = \max_{y \in \mathbb{Y}^{\mathrm{d}}} \min_{z \in \mathbb{X}^{\mathrm{d}}} \left\{ \langle x - z, y \rangle + h(z) \right\}, \tag{6}$$

where $\mathbb{X}^{\mathrm{d}}$ and $\mathbb{Y}^{\mathrm{d}}$ are finite subsets of $\mathbb{R}^n$ such that $\mathbb{X}^{\mathrm{d}} \cap \mathbb{X} \neq \emptyset$.

The Linear-time Legendre Transform (LLT) is an efficient algorithm for computing the discrete conjugate over a finite *grid-like* dual domain. Precisely, to compute the conjugate of the function $h : \mathbb{X} \to \mathbb{R}$, LLT takes its discretization $h^{\mathrm{d}} : \mathbb{X}^{\mathrm{d}} \to \mathbb{R}$ as an input, and outputs $h^{\mathrm{d}*\mathrm{d}} : \mathbb{Y}^{\mathrm{g}} \to \mathbb{R}$, for the grid-like dual domain $\mathbb{Y}^{\mathrm{g}}$. That is, LLT is equivalent to the operation $[\cdot]^{\mathrm{d}*\mathrm{d}}$. We refer the interested reader to [23] for a detailed description of the LLT algorithm. We will use the following result for analyzing the computational complexity of the proposed algorithms.

Remark 2.3 (Complexity of LLT)**.** *Consider a function $h : \mathbb{R}^n \to \overline{\mathbb{R}}$ and its discretization over a grid-like set $\mathbb{X}^{\mathrm{g}} \subset \mathbb{R}^n$ such that $\mathbb{X}^{\mathrm{g}} \cap \mathrm{dom}(h) \neq \emptyset$. LLT computes the discrete conjugate function $h^{\mathrm{d}*\mathrm{d}} : \mathbb{Y}^{\mathrm{g}} \to \mathbb{R}$ using the data points $h^{\mathrm{d}} : \mathbb{X}^{\mathrm{g}} \to \overline{\mathbb{R}}$, with a time complexity of $\mathcal{O}\left(\Pi_{i=1}^{n}(X_i + Y_i)\right)$, where X_i (resp. Y_i) is the cardinality of the i-th dimension of the grid $\mathbb{X}^{\mathrm{g}}$ (resp. $\mathbb{Y}^{\mathrm{g}}$). In particular, if the grids $\mathbb{X}^{\mathrm{g}}$ and $\mathbb{Y}^{\mathrm{g}}$ have approximately the same cardinality in each dimension, then the time complexity of LLT is of $\mathcal{O}(X + Y)$ [23, Cor. 5].*

Hereafter, in order to simplify the exposition, we consider the following assumption.

Assumption 2.4 (Grid sizes in LLT)**.** *The primal and dual grids used for LLT operation have approximately the same cardinality in each dimension.*

2.4. **Preliminary results on conjugate transform**

In what follows, we provide two preliminary lemmas on the error of discrete conjugate transform and its approximate version. Although tailored for the error analysis of the proposed algorithms, we present these results in a generic format in order to facilitate their possible application/extension beyond this study. To this end, we recall some of the notations introduced so far. For a function $h : \mathbb{R}^n \to \overline{\mathbb{R}}$ with effective domain $\mathbb{X} = \mathrm{dom}(h)$, let $h^{\mathrm{d}} : \mathbb{X}^{\mathrm{d}} \to \mathbb{R}$ be its discretization of h, where $\mathbb{X}^{\mathrm{d}} \subset \mathbb{X}$. Also, let $h^* : \mathbb{R}^n \to \overline{\mathbb{R}}$ be the conjugate (3) of h, and $h^{\mathrm{d}*} : \mathbb{R}^n \to \mathbb{R}$ be the discrete conjugate (4) of h, using the primal discrete domain $\mathbb{X}^{\mathrm{d}}$.

Lemma 2.5 (Conjugate vs. discrete conjugate)**.** *Let $h : \mathbb{R}^n \to \overline{\mathbb{R}}$ be a proper, closed, convex function with a nonempty domain $\mathbb{X} = \mathrm{dom}(h)$. For each $y \in \mathbb{R}^n$, it holds that*

$$0 \le h^*(y) - h^{\mathrm{d}*}(y) \le \min_{x \in \partial h^*(y)} \left\{ \left[\, \|y\| + \mathrm{L}\left(h; \{x\} \cup \mathbb{X}^{\mathrm{d}}\right) \right] \cdot \mathrm{d}(x, \mathbb{X}^{\mathrm{d}}) \right\} =: \widetilde{e}_1(y, h, \mathbb{X}^{\mathrm{d}}). \tag{7}$$

If, moreover, $\mathbb{X}$ is compact and h is Lipschitz continuous, then

$$0 \le h^*(y) - h^{\mathrm{d}*}(y) \le \left[\, \|y\| + \mathrm{L}(h) \right] \cdot \mathrm{d_H}(\mathbb{X}, \mathbb{X}^{\mathrm{d}}) =: \widetilde{e}_2(y, h, \mathbb{X}^{\mathrm{d}}), \quad \forall y \in \mathbb{R}^n. \tag{8}$$

Proof. See Appendix B.1. □

The preceding lemma indicates that discrete conjugation leads to an under-approximation of the conjugate function, with the error depending on the discrete representation $\mathbb{X}^{\mathrm{d}}$ of the primal domain $\mathbb{X}$. In particular, the inequality (7) implies that for $y \in \mathbb{R}^n$, if $\mathbb{X}^{\mathrm{d}}$ contains $x \in \partial h^*(y)$, which is equivalent to $y \in \partial h(x)$ by the assumptions, then $h^{\mathrm{d}*}(y) = h^*(y)$.

We next present another preliminary however vital result on approximate conjugation. Let $h^{*\mathrm{d}} : \mathbb{Y}^{\mathrm{g}} \to \mathbb{R}$ be the discretization of h^* over the grid-like dual domain $\mathbb{Y}^{\mathrm{g}} \subset \mathbb{R}^n$. Also, let $\overline{h^{*\mathrm{d}}} : \mathbb{R}^n \to \mathbb{R}$ be the extension of $h^{*\mathrm{d}}$ using LERP. The approximate conjugation is then simply the approximation of $h^*(y)$ via $\overline{h^{*\mathrm{d}}}(y)$ for $y \in \mathbb{R}^n$. This approximation introduces a one-sided error:

Lemma 2.6 (Approximate conjugation using LERP)**.** *Consider a function h with a compact domain $\mathbb{X} = \mathrm{dom}(h)$. For all $y \in \mathrm{co}(\mathbb{Y}^{\mathrm{g}})$, it holds that*

$$0 \le \overline{h^{*\mathrm{d}}}(y) - h^*(y) \le \Delta_{\mathbb{X}} \cdot \mathrm{d}(y, \mathbb{Y}^{\mathrm{g}}). \tag{9}$$

If, moreover, the dual grid $\mathbb{Y}^{\mathrm{g}}$ is such that $\mathrm{co}(\mathbb{Y}^{\mathrm{g}}_{\mathrm{sub}}) \supseteq \mathbb{L}(h)$, then, for all $y \in \mathbb{R}^n$, it holds that

$$0 \le \overline{h^{*\mathrm{d}}}(y) - h^*(y) \le \Delta_{\mathbb{X}} \cdot \mathrm{d_H}\left(\mathrm{co}(\mathbb{Y}^{\mathrm{g}}), \mathbb{Y}^{\mathrm{g}}\right). \tag{10}$$

Proof. See Appendix B.2. □

As expected, the error due to the discretization $\mathbb{Y}^{\mathrm{g}}$ of the dual domain $\mathbb{Y}$ depends on the resolution of the discrete dual domain. We also note that the condition $\mathrm{co}(\mathbb{Y}^{\mathrm{g}}_{\mathrm{sub}}) \supseteq \mathbb{L}(h)$ in the second part of the preceding lemma (which implies that h is Lipschitz continuous), essentially requires the dual grid $\mathbb{Y}^{\mathrm{g}}$ to *more than cover the range of slopes* of the function h.

The algorithms developed in this study use LLT to compute discrete conjugate functions. However, as we will see, we sometimes require the value of the conjugate function at points other than the dual grid points used in LLT. To solve this issue, we use the same approximation described above, but now for discrete conjugation. In this regard, we note that the result of Lemme 2.6 also holds for discrete conjugation. To be precise, let $h^{\mathrm{d*d}} : \mathbb{Y}^{\mathrm{g}} \to \mathbb{R}$ be the discretization of $h^{\mathrm{d*}}$ over the grid-like dual domain $\mathbb{Y}^{\mathrm{g}} \subset \mathbb{R}^n$, and $\overline{h^{\mathrm{d*d}}} : \mathbb{R}^n \to \mathbb{R}$ be the extension of $h^{\mathrm{d*d}}$ using LERP.

Corollary 2.7 (Approximate discrete conjugation using LERP)**.** *Consider a discrete function $h^{\mathrm{d}} : \mathbb{X}^{\mathrm{d}} \to \mathbb{R}$. For all $y \in \mathrm{co}(\mathbb{Y}^{\mathrm{g}})$, it holds that*

$$0 \leq \overline{h^{\mathrm{d*d}}}(y) - h^{\mathrm{d*}}(y) \leq \Delta_{\mathbb{X}^{\mathrm{d}}} \cdot \mathrm{d}(y, \mathbb{Y}^{\mathrm{g}}). \tag{11}$$

If, moreover, the dual grid $\mathbb{Y}^{\mathrm{g}}$ is such that $\mathrm{co}(\mathbb{Y}^{\mathrm{g}}_{\mathrm{sub}}) \supseteq \mathbb{L}(h^{\mathrm{d}})$, then, for all $y \in \mathbb{R}^n$, it holds that

$$0 \leq \overline{h^{\mathrm{d*d}}}(y) - h^{\mathrm{d*}}(y) \leq \Delta_{\mathbb{X}^{\mathrm{d}}} \cdot \mathrm{d_H}\left(\mathrm{co}(\mathbb{Y}^{\mathrm{g}}), \mathbb{Y}^{\mathrm{g}}\right). \tag{12}$$

Proof. See Appendix B.3. □

3. Problem Statement and Standard Solution

In this study, we consider the optimal control of discrete-time systems

$$x_{t+1} = f(x_t, u_t), \quad t = 0, \dots, T-1, \tag{13}$$

where $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ describes the dynamics, and $T \in \mathbb{N}$ is the finite horizon. We also consider state and input constraints of the form

$$\begin{cases} x_t \in \mathbb{X} \subset \mathbb{R}^n & \text{for} \quad t \in \{0, \dots, T\}, \\ u_t \in \mathbb{U} \subset \mathbb{R}^m & \text{for} \quad t \in \{0, \dots, T-1\}. \end{cases} \tag{14}$$

Let $C : \mathbb{X} \times \mathbb{U} \to \overline{\mathbb{R}}$ and $C_T : \mathbb{X} \to \mathbb{R}$ be the stage and terminal cost functions, respectively. In particular, notice that we let the stage cost C take ∞ for $(x, u) \in \mathbb{X} \times \mathbb{U}$ so that it can embed the *state-dependent input constraints*. For an initial state $x_0 \in \mathbb{X}$, the cost incurred by the state trajectory $\mathbf{x} = (x_0, \dots, x_T)$ in response to the input sequence $\mathbf{u} = (u_0, \dots, u_{T-1})$ is given by

$$J(x_0, \mathbf{u}) = \textstyle\sum_{t=0}^{T-1} C(x_t, u_t) + C_T(x_T).$$

The problem of interest is then to find an optimal control sequence $\mathbf{u}^\star(x_0)$, that is, a solution to the minimization problem

$$J^\star(x_0) = \min_{\mathbf{u}} \left\{ J(x_0, \mathbf{u}) : (13) \ \& \ (14) \right\}. \tag{15}$$

In order to solve this problem using DP, we have to solve the Bellman equation

$$J_t(x_t) = \min_{u} \left\{ C(x_t, u_t) + J_{t+1}(x_{t+1}) : (13) \ \& \ (14) \right\}, \quad x_t \in \mathbb{X},$$

backward in time $t = T-1, \dots, 0$, initialized by $J_T = C_T$. The iteration finally outputs $J_0 = J^\star$ [7, Prop. 1.3.1]. In order to simplify the exposition, let us embed the state and input constraints in the cost functions (C and J_t) by extending them to infinity outside their effective domain. Let us also drop the time subscript t and focus on a single step of the recursion by defining the DP operator

$$\mathcal{T}[J](x) := \min_{u} \left\{ C(x, u) + J\big(f(x, u)\big) \right\}, \quad x \in \mathbb{X}, \tag{16}$$

so that $J_t = \mathcal{T}[J_{t+1}] = \mathcal{T}^{(T-t)}[J_T]$ for $t = T-1, \dots, 0$. *The main focus of this study is to use conjugate duality in order to develop "fast" schemes for the numerical implementation of the DP* *operator* (16) *for particular classes of optimal control problems.* Notice, however, the DP operation (16) requires solving an infinite number of optimization problems for all $x \in \mathbb{X}$. Except for a few cases with an available closed form solution, the exact implementation of DP operation is impossible. A standard approximation scheme is then to incorporate function approximation techniques and solve (16) for a finite sample (i.e., a discretization) of the underlying continuous state space. Precisely, we consider solving the optimization in (16) for a finite number of $x \in \mathbb{X}^{\mathrm{g}}$, where $\mathbb{X}^{\mathrm{g}} \subset \mathbb{X}$ is a grid-like discretization of the state space:

Problem 3.1 (Value iteration)**.** *Given the discretization $\mathbb{X}^{\mathrm{g}} \subset \mathbb{X}$ of the state space, find the discrete costs-to-go $J_t^{\mathrm{d}} : \mathbb{X}^{\mathrm{g}} \to \mathbb{R}$ for $t = 0, 1, \dots, T-1$.*

Notice that the DP operator $\mathcal{T}$ now takes the discrete function $J^{\mathrm{d}} : \mathbb{X}^{\mathrm{g}} \to \mathbb{R}$ as an input. However, in order to compute the output $[\mathcal{T}[J]]^{\mathrm{d}} : \mathbb{X}^{\mathrm{g}} \to \mathbb{R}$, we require evaluating J at points $f(x,u)$ for $(x,u) \in \mathbb{X}^{\mathrm{g}} \times \mathbb{U}$, which do not necessarily belong to the discrete state space $\mathbb{X}^{\mathrm{g}}$. Hence, along with the discretization of the state space, we also need to consider some form of function approximation for the cost-to-go function, that is, an extension $\widetilde{J^{\mathrm{d}}} : \mathbb{X} \to \mathbb{R}$ of the function $J^{\mathrm{d}} : \mathbb{X}^{\mathrm{g}} \to \mathbb{R}$.

What remains to be addressed is the issue of solving the minimization

$$\min_{u \in \mathbb{U}} \left\{ C(x,u) + \widetilde{J^{\mathrm{d}}}\big(f(x,u)\big) \right\}, \tag{17}$$

for each $x \in \mathbb{X}^{\mathrm{g}}$, where the next step cost-to-go is approximated by the extension $\widetilde{J^{\mathrm{d}}}$. Here, again, we consider an approximation that involves enumeration over a proper discretization $\mathbb{U}^{\mathrm{g}} \subset \mathbb{U}$ of the inputs space. These approximations introduce some error which, under some regularity assumptions, depends on the discretization of the state and input spaces and the extension operation; see Proposition A.1. Incorporating these approximations, we can introduce the *discrete* DP (d-DP) operator as follows

$$\mathcal{T}^{\mathrm{d}}[J^{\mathrm{d}}](x) := \min_{u \in \mathbb{U}^{\mathrm{g}}} \left\{ C(x,u) + \widetilde{J^{\mathrm{d}}}\big(f(x,u)\big) \right\}, \quad x \in \mathbb{X}^{\mathrm{g}}. \tag{18}$$

The d-DP operator/algorithm will be our benchmark for evaluating the performance of the alternative algorithms developed in this study. To this end, we discuss the time complexity of the d-DP operation in the following remark.

Remark 3.2 (Complexity of d-DP)**.** *Let the time complexity of a single evaluation of the extension operator $\widetilde{[\cdot]}$ in (18) be of $\mathcal{O}(E)$. Then, the time complexity of the d-DP operation (18) is of $\mathcal{O}\big(XUE\big)$. Moreover, for solving the T-step value iteration Problem 3.1, the time complexity of d-DP algorithm increases linearly with the horizon T.*

Let us clarify that the scheme described above essentially involves approximating a continuous-state/action MDP with a finite-state/action MDP, and then applying the value iteration algorithm. In this regard, we note that $\mathcal{O}(XU)$ is the best existing time-complexity in the literature for finite MDPs; see, e.g., [3, 32]. Indeed, regardless of the problem data, the d-DP algorithm involves solving a minimization problem for each $x \in \mathbb{X}^{\mathrm{g}}$, via enumeration over $u \in \mathbb{U}^{\mathrm{g}}$. However, as we will see in the subsequent sections, for certain classes of optimal control problems, it is possible to exploit the structure of the underlying continuous setup to avoid the minimization over the input and achieve a lower time complexity.

We finish this section with some remarks on using the output of the d-DP algorithm, for finding a suboptimal control sequence $\mathbf{u}^\star(x_0)$ for a given instance of the optimal control problem with initial state x_0. Upon solving the value iteration Problem 3.1, we have the discrete costs-to-go $J_t^{\mathrm{d}} : \mathbb{X}^{\mathrm{g}} \to \mathbb{R},\ t = 0, 1, \ldots, T-1$, at our disposal. Then, in order to find $\mathbf{u}^\star(x_0)$, we can use the greedy policy w.r.t. to computed costs J_t^{d} by solving T minimization problems forward in time, i.e.,

$$u_t^\star \in \operatorname*{argmin}_{u_t \in \mathbb{U}^{\mathrm{g}}} \left\{ C(x_t, u_t) + \widetilde{J_{t+1}^{\mathrm{d}}}\big(f(x_t, u_t)\big) \right\}, \quad t = 0, 1, \ldots, T-1. \tag{19}$$

This leads to an additional computational burden of $\mathcal{O}(TUE)$ in solving a T-step optimal control problem using the d-DP algorithm, where E represents the complexity of the extension operation used in (19). On the other hand, the backward value iteration using the d-DP algorithm also provides us with control laws $\mu_t^{\mathrm{d}} : \mathbb{X}^{\mathrm{g}} \to \mathbb{U}^{\mathrm{g}},\ t = 0, 1, \ldots, T-1$. Hence, alternatively, we can use these laws, accompanied by a proper extension operator, to produce a suboptimal control sequence, i.e.,

$$u_t^\star(x_t) = \widetilde{\mu_t^{\mathrm{d}}}(x_t), \quad t = 0, 1, \ldots, T-1. \tag{20}$$

The second method (using control laws) then has a time complexity of $\mathcal{O}(TE)$, where E represents the complexity of the extension operation used in (20). This complexity can be particularly lower than $\mathcal{O}(TUE)$ of the first method (using costs). However, generating control actions using the control laws has a higher memory complexity for systems with multiple inputs, and is also usually more sensitive to modelling errors due to its completely open-loop nature. Moreover, we note that the *total* time complexity of solving an instance of the optimal control problem, i.e., backward iteration for computing of costs J_t^{d} and control laws μ_t^{d}, and forward iteration for computing of control sequence $\mathbf{u}^\star(x_0)$, is in both methods of $\mathcal{O}(TXUE)$. That is, computationally, the backward value iteration is the dominating factor. We will see this effect in our numerical study in Section 6, where we consider both of the methods described above.

4. Alternative Solution: DP in Conjugate Domain

In this section, we introduce a general class of problems that allow us to employ conjugate duality for the DP problem and hence propose an alternative path for implementing the corresponding operator. In particular, we analyze the performance of the proposed algorithm by considering its time complexity and error. We also discuss the extensions of the proposed algorithm, which address two of the assumptions in our development.

4.1. The d-CDP algorithm

For now, we focus on deterministic systems described by (13) and (14), and postpone the discussion on the extension of the proposed scheme for stochastic systems to Section 4.3. Throughout this study, we assume that the problem data satisfy the following conditions.

Setting 1. *The dynamics, constraints and costs have the following properties:*

- *(i)* **Dynamics.** *The dynamics is input-affine, that is,* $f(x, u) = f_{\mathrm{s}}(x) + f_{\mathrm{i}}(x) \cdot u$, *where* $f_{\mathrm{s}} : \mathbb{R}^n \to \mathbb{R}^n$ *is the "state" dynamics, and* $f_{\mathrm{i}} : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ *is the "input" dynamics.*

(ii) **Constraints.** *The sets* $\mathbb{X}$ *and* $\mathbb{U}$ *are compact and convex. Moreover, for each* $x \in \mathbb{X}$*, the set of admissible inputs* $\mathbb{U}(x) := \{u \in \mathbb{U} : C(x,u) < \infty,\ f(x,u) \in \mathbb{X}\}$ *is nonempty.*

(iii) **Cost functions.** *The stage cost* C *is jointly convex in the state and input variables, with a compact effective domain. The terminal cost* C_T *is also convex.*

Note that the properties laid out in Setting 1 imply that the set of admissible inputs $\mathbb{U}(x)$ is a nonempty, compact, convex set for each $x \in \mathbb{X}$. Hence, the optimal value in (15) is achieved. Hereafter, we also assume that the joint discretization of the state-input space is "proper" in the sense that the feasibility condition of Setting 1-(ii) holds for the discrete state-input space:

Assumption 4.1 (Feasibile discretization)**.** *The discrete state space* $\mathbb{X}^{\mathrm{g}} \subset \mathbb{X}$ *and input space* $\mathbb{U}^{\mathrm{g}} \subset \mathbb{U}$ *are such that* $\mathbb{U}^{\mathrm{g}}(x) := \mathbb{U}(x) \cap \mathbb{U}^{\mathrm{g}} \neq \emptyset$ *for all* $x \in \mathbb{X}^{\mathrm{g}}$*.*

Alternatively, we can approach the optimization problem in the DP operation (16) in the dual domain. To this end, let us fix $x \in \mathbb{X}^{\mathrm{g}}$, and consider the following reformulation of the problem (16):

$$\mathcal{T}[J](x) = \min_{u,z} \{C(x,u) + J(z) : z = f(x,u)\}.$$

Notice how for input-affine dynamics, this formulation resembles the infimal convolution. In this regard, consider the corresponding dual problem

$$\widehat{\mathcal{T}}[J](x) := \max_{y}\ \min_{u,z}\ \{C(x,u) + J(z) + \langle y, f(x,u) - z\rangle\}, \tag{21}$$

where $y \in \mathbb{R}^n$ is the dual variable. Indeed, for input-affine dynamics, we can derive an equivalent formulation for the dual problem (21), which forms the basis for the proposed algorithms.

Lemma 4.2 (CDP operator)**.** *Let*

$$C_x^*(v) := \max_{u} \big\{ \langle v, u\rangle - C(x,u)\big\}, \quad v \in \mathbb{R}^m, \tag{22}$$

denote the partial conjugate of the stage cost w.r.t. the input variable u*. Then, for the input-affine dynamics of Setting 1-(i), the operator* $\widehat{\mathcal{T}}$ (21) *equivalently reads as*

$$\widehat{\mathcal{T}}[J](x) = \phi_x^*\big(f_{\mathrm{s}}(x)\big), \quad x \in \mathbb{X}^{\mathrm{g}}, \tag{23a}$$

$$\phi_x(y) := C_x^*(-f_{\mathrm{i}}(x)^\top y) + J^*(y), \quad y \in \mathbb{R}^n. \tag{23b}$$

Proof. See Appendix B.4. □

As we mentioned, the construction above suggests an alternative path for computing the output of the DP operator through the conjugate domain. We call this alternative approach *conjugate* DP (CDP). Figure 1a characterizes this alternative path schematically. Numerical implementation of CDP operation requires computation of conjugate functions. In particular, as shown in Figure 1a, CDP operation involves three conjugate transforms. For now, we assume that the partial conjugate C_x^* of the stage cost in (22) is analytically available. We will discuss the possibility of computing this object numerically in Section 4.3.

Assumption 4.3 (Conjugate of stage cost)**.** *For each* $x \in \mathbb{X}^{\mathrm{g}}$*, the conjugate function* C_x^* (22) *is analytically available. That is, the time complexity of evaluating* $C_x^*(v)$ *for each* $v \in \mathbb{R}^m$ *is of* $\mathcal{O}(1)$*.*

Algorithm 1 Implementation of the d-CDP operator (24) for Setting 1.

Input: dynamics $f_{\rm s} : \mathbb{R}^n \to \mathbb{R}^n$, $f_{\rm i} : \mathbb{R}^n \to \mathbb{R}^{n\times m}$;
discrete cost-to-go (at $t+1$) $J^{\rm d} : \mathbb{X}^{\rm g} \to \mathbb{R}$;
conjugate of stage cost $C_x^* : \mathbb{X}^{\rm g} \times \mathbb{R}^m \to \mathbb{R}$;
grid $\mathbb{Y}^{\rm g} \subset \mathbb{R}^n$;

Output: discrete cost-to-go (at t) $\widehat{\mathcal{T}}^{\rm d}[J^{\rm d}](x) : \mathbb{X}^{\rm g} \to \mathbb{R}$.

```
use LLT to compute J^{d*d} : Y^g → R from J^d : X^g → R;
for each x ∈ X^g do
    φ_x^d(y) ← C_x^*(−f_i(x)^⊤ y) + J^{d*d}(y) for y ∈ Y^g;
    T̂^d[J^d](x) ← max_{y∈Y^g} {⟨f_s(x), y⟩ − φ_x^d(y)}.
end for
```

The two remaining conjugate operations of the CDP path in Figure 1a are handled numerically. To be precise, first, for a grid-like discretization $\mathbb{Y}^{\rm g}$ of the dual domain, we employ LLT to compute $J^{\rm d*d} : \mathbb{Y}^{\rm g} \to \mathbb{R}$ using the data points $J^{\rm d} : \mathbb{X}^{\rm g} \to \mathbb{R}$. We will discuss the construction of $\mathbb{Y}^{\rm g}$ in Section 4.2.2. Now, let

$$\varphi_x^{\rm d}(y) = C_x^*(-f_{\rm i}(x)^\top y) + J^{\rm d*d}(y), \quad y \in \mathbb{Y}^{\rm g},$$

be an approximation of ϕ_x (23b), where we used the discrete conjugate $J^{\rm d*}$ instead of the conjugate J^*. Then, we can also handle the last conjugate transform numerically, and approximate $\phi_x^*\big(f_{\rm s}(x)\big)$ by

$$\varphi_x^{\rm d*}\big(f_{\rm s}(x)\big) = \max_{y\in\mathbb{Y}^{\rm g}} \{\langle f_{\rm s}(x), y\rangle - \varphi_x(y)\},$$

via enumeration over $y \in \mathbb{Y}^{\rm g}$. Based on the construction described above, we can introduce the *discrete* CDP (d-CDP) operator as follows

$$\widehat{\mathcal{T}}^{\rm d}[J^{\rm d}](x) := \varphi_x^{\rm d*}\big(f_{\rm s}(x)\big), \quad x \in \mathbb{X}^{\rm g}, \tag{24a}$$

$$\varphi_x^{\rm d}(y) := C_x^*(-f_{\rm i}(x)^\top y) + J^{\rm d*d}(y), \quad y \in \mathbb{Y}^{\rm g}. \tag{24b}$$

Algorithm 1 provides the pseudo-code for the numerical implementation of the d-CDP operation (24). In the next subsection, we analyze the complexity and error of Algorithm 1.

4.2. **Analysis of d-CDP algorithm**

4.2.1. *Complexity.* We begin with the computational complexity of Algorithm 1.

Theorem 4.4 (Complexity of d-CDP – Algorithm 1)**.** *Let Assumptions 2.4 and 4.3 hold. Then, the implementation of the d-CDP operator* (24) *via Algorithm 1 requires $\mathcal{O}(XY)$ operations.*

Proof. See Appendix B.5. □

Recall that the time complexity of the d-DP operator (18) is of $\mathcal{O}(XUE)$; see Remark 3.2. Comparing this complexity to the one reported in Theorem 4.4, points to a basic characteristic of CDP w.r.t. DP: CDP avoids the minimization over the control input in DP and casts it as a simple addition in the dual domain at the expense of three conjugate transforms. Consequently, the time complexity is transferred form the primal input domain $\mathbb{U}^{\rm g}$ into the state dual domain $\mathbb{Y}^{\rm g}$. This observation implies that if $Y < UE$, then d-CDP is expected to computationally outperform d-DP.

We finish with some remarks on the application of the d-CDP algorithm for finding a suboptimal control sequence $\mathbf{u}^\star(x_0)$ for a given initial state x_0. In this regard, first notice that in the T-step application of Algorithm 1 for solving the value iteration Problem 3.1, the complexity increases linearly with the horizon T (assuming that the dual grid $\mathbb{Y}^{\mathrm{g}}$ can be constructed with at most $\mathcal{O}(X)$ operations; see also Remark 4.7). More importantly, note that, unlike the d-DP algorithm, the backward value iteration using the d-CDP algorithm only provides us with the costs-to-go $J_t^{\mathrm{d}} : \mathbb{X}^{\mathrm{g}} \to \mathbb{R}$, $t = 0, 1, \ldots, T-1$. Hence, in order to find $\mathbf{u}^\star(x_0)$, we can again use the greedy policy (19) w.r.t. the costs J_t^{d} computed using the d-CDP algorithm. Hence, an additional burden of $\mathcal{O}(TUE)$ must be considered for finding $\mathbf{u}^\star(x_0)$ using the d-CDP algorithm, where E represents the complexity of the extension operation used in (19). As a result, the *total* complexity of computing a control sequence for a given initial state using the d-CDP Algorithm 1 is of $\mathcal{O}\big(T(XY + UE)\big)$.

4.2.2. *Error.* We now consider the error introduced by Algorithm 1 w.r.t. the DP operator (16). Let us begin with presenting an alternative representation of the d-CDP operator that sheds some light on the main sources of error in the d-CDP operation.

Proposition 4.5 (d-CDP reformulation)**.** *The d-CDP operator* (24) *equivalently reads as*

$$\widehat{\mathcal{T}}^{\mathrm{d}}[J^{\mathrm{d}}](x) = \min_u \Big\{ C(x,u) + J^{\mathrm{d*d*}}\big(f(x,u)\big) \Big\}, \quad x \in \mathbb{X}^{\mathrm{g}}, \tag{25}$$

where $J^{\mathrm{d*d*}}$ is the discrete biconjugate of J, using the primal and dual grids $\mathbb{X}^{\mathrm{g}}$ and $\mathbb{Y}^{\mathrm{g}}$, respectively.

Proof. See Appendix B.6. □

Comparing the representations (16) and (25), we note that the d-CDP operator $\widehat{\mathcal{T}}^{\mathrm{d}}$ differs from the DP operator $\mathcal{T}$ in that it uses $J^{\mathrm{d*d*}}$ as an approximation of J. This observation points to two main sources of error in the proposed approach, namely, dualization and discretization. Indeed, $\widehat{\mathcal{T}}^{\mathrm{d}}$ is a discretized version of the dual problem (21). Regarding the dualization error, we note that the d-CDP operator is "blind" to non-convexity; that is, it essentially replaces the cost-to-go J by its convex envelop (the greatest convex function that supports J from below). The discretization error, on the other hand, depends on the choice of the finite primal and dual domains $\mathbb{X}^{\mathrm{g}}$ and $\mathbb{Y}^{\mathrm{g}}$. In particular, by a proper choice of $\mathbb{Y}^{\mathrm{g}}$, it is indeed possible to eliminate the corresponding error due to discretization of the dual domain. To illustrate, let J^{d} be a one-dimensional, discrete, convex-extensible function with domain $\mathbb{X}^{\mathrm{g}} = \{x^i\}_{i=1}^N \subset \mathbb{R}$, where $x^i < x^{i+1}$. (By convex-extensible, we mean that J^{d} can be extended to convex function $\widetilde{J^{\mathrm{d}}}$ such that $\widetilde{J^{\mathrm{d}}}(x) = J^{\mathrm{d}}(x)$ for all $x \in \mathbb{X}^{\mathrm{g}}$). Also, choose $\mathbb{Y}^{\mathrm{g}} = \{y^i\}_{i=1}^{N-1} \subset \mathbb{R}$ with $y^i = \frac{J^{\mathrm{d}}(x^{i+1}) - J^{\mathrm{d}}(x^i)}{x^{i+1} - x^i}$ as the discrete dual domain. Then, for all $x \in \mathrm{co}(\mathbb{X}^{\mathrm{g}}) = [x^1, x^N]$, we have $J^{\mathrm{d*d*}}(x) = \overline{J^{\mathrm{d}}}(x)$, where $\overline{[\cdot]}$ is the LERP extension. Hence, the only source of error under such construction is the discretization of the primal state space (i.e., approximation of the true J via $\overline{J^{\mathrm{d}}}$). However, a similar construction of $\mathbb{Y}^{\mathrm{g}}$ in dimensions $n \geq 2$ can lead to dual grids of size $Y = \mathcal{O}(X^n)$, which is computationally impractical; see Theorem 4.4. The following result provides us with specific bounds on the discretization error that point to a more practical way for construction of $\mathbb{Y}^{\mathrm{g}}$.

Theorem 4.6 (Error of d-CDP – Algorithm 1)**.** *Consider the DP operator $\mathcal{T}$* (16) *and the implementation of the d-CDP operator $\widehat{\mathcal{T}}^{\mathrm{d}}$* (24) *via Algorithm 1. Assume that $J : \mathbb{X} \to \mathbb{R}$ is a Lipschitz* *continuous, convex function. Then, for each $x \in \mathbb{X}^{\mathrm{g}}$, it holds that*

$$-e_2 \leq \mathcal{T}[J](x) - \widehat{\mathcal{T}}^{\mathrm{d}}[J^{\mathrm{d}}](x) \leq e_1(x), \tag{26}$$

where

$$\begin{aligned} e_1(x) &= \big[\, \|f_{\mathrm{s}}(x)\| + \|f_{\mathrm{i}}(x)\| \cdot \Delta_{\mathbb{U}} + \Delta_{\mathbb{X}} \big] \cdot \mathrm{d}\left(\partial \mathcal{T}[J](x), \mathbb{Y}^{\mathrm{g}}\right), \\ e_2 &= [\Delta_{\mathbb{Y}^{\mathrm{g}}} + \mathrm{L}(J)] \cdot \mathrm{d}_{\mathrm{H}}(\mathbb{X}, \mathbb{X}^{\mathrm{g}}). \end{aligned} \tag{27}$$

Proof. See Appendix B.7. □

Notice how the two terms e_1 and e_2 capture the errors due to the discretization of the state dual space ($\mathbb{Y}$) and the primal state space ($\mathbb{X}$), respectively. In particular, the first error term suggests that we choose choose $\mathbb{Y}^{\mathrm{g}}$ such that $\partial\mathcal{T}[J](x) \cap \mathbb{Y}^{\mathrm{g}} \neq \emptyset$ for all $x \in \mathbb{X}^{\mathrm{g}}$. Even if we had access to $\mathcal{T}[J]$, satisfying such a condition can again lead to dual grids of size $Y = \mathcal{O}(X^n)$. A more realistic objective is then to choose $\mathbb{Y}^{\mathrm{g}}$ such that $\mathrm{co}(\mathbb{Y}^{\mathrm{g}}) \cap \partial\mathcal{T}[J](x) \neq \emptyset$ for all $x \in \mathbb{X}^{\mathrm{g}}$. With such a construction, the distance $\mathrm{d}\left(\partial\mathcal{T}[J](x), \mathbb{Y}^{\mathrm{g}}\right)$ and hence e_1 decrease by using finer grids for the dual domain. To this end, we need to approximate "the range of slopes" of the function $\mathcal{T}[J]$ for $x \in \mathbb{X}^{\mathrm{g}}$. Notice, however, that we do not have access to $\mathcal{T}[J]$ since it is the *output* of the d-CDP operation in Algorithm 1. What we have at our disposal as *inputs* are the stage cost C and the next step (discrete) cost-to-go J^{d}. A coarse way to approximate the range of slopes of $\mathcal{T}[J]$ is to use the extrema of the functions C and J^{d}, and the diameter of $\mathbb{X}^{\mathrm{g}}$ in each dimension. The following remark explains such an approximation for the construction of $\mathbb{Y}^{\mathrm{g}}$.

Remark 4.7 (Construction of $\mathbb{Y}^{\mathrm{g}}$)**.** *Compute*

$$C^M = \max_{x \in \mathbb{X}^{\mathrm{g}}} \max_{u \in \mathbb{U}(x)} C(x,u), \;\; C^m = \min_{x \in \mathbb{X}^{\mathrm{g}}} \min_{u \in \mathbb{U}(x)} C(x,u), \;\; J^M = \max J^{\mathrm{d}}, \;\; J^m = \min J^{\mathrm{d}},$$

and then choose $\mathbb{Y}^{\mathrm{g}} = \Pi_{i=1}^n \mathbb{Y}_i^{\mathrm{g}} \subset \mathbb{R}^n$ such that for each dimension $i = 1, \ldots, n$, we have

$$\pm\alpha \cdot \frac{C^M + J^M - C^m - J^m}{\Delta_{\mathbb{X}_i^{\mathrm{g}}}} \in \mathrm{co}(\mathbb{Y}_i^{\mathrm{g}}).$$

Here, $\alpha > 0$ is a scaling factor mainly depending on the dimension n of the state space. This construction has a linear time complexity w.r.t. the size X of the state space.

4.3. **Extensions of d-CDP algorithm**

In this section, we consider the extensions of the proposed d-CDP algorithm and their implications on its complexity. In particular, the extension to stochastic systems with additive disturbance and the possibility of numerical computation of the conjugate of the stage cost are discussed. The pseudo-code for the multistep implementation of the extended d-CDP algorithm is provided in Appendix C.1.

4.3.1. *Stochastic systems.* Consider the stochastic version of the dynamics (13) described by

$$x_{t+1} = f(x_t, u_t) + w_t,$$

where $w_t,\ t = 0, \ldots, T-1$, are independent, *additive* disturbances. Then, the stochastic version of the CDP operator $\widehat{\mathcal{T}}$ (23) still reads the same, except its takes $J_{\mathrm{w}}(\cdot) := \mathbb{E}_w J(\cdot + w)$ as the input, where $\mathbb{E}_w$ is the expectation operator w.r.t. w. In other words, we need to first pass the cost-to-go J through the "expectation filter", and then feed it to the CDP operator. The extension of the d-CDP algorithm for handling this type of stochasticity involves similar considerations. To illustrate, assume that the disturbances belong to a finite set $\mathbb{W}^{\mathrm{d}} \subset \mathbb{R}^n$, with a known probability mass function (p.m.f.) $p : \mathbb{W}^{\mathrm{d}} \to [0, 1]$. The set $\mathbb{W}^{\mathrm{d}}$ can indeed be considered as a discretization of a bounded set of disturbances. Then, the corresponding extension of the d-CDP algorithm involves applying $\widehat{\mathcal{T}}^{\mathrm{d}}$ (24) to $J_{\mathrm{w}}^{\mathrm{d}}(x) : \mathbb{X}^{\mathrm{g}} \to \overline{\mathbb{R}}$ given by

$$J_{\mathrm{w}}^{\mathrm{d}}(x) = \textstyle\sum_{w \in \mathbb{W}^{\mathrm{d}}} p(w) \cdot \widetilde{J^{\mathrm{d}}}(x + w), \tag{28}$$

where $\widetilde{[\cdot]}$ is an extension operator such as LERP.

Assuming that a single evaluation of the employed extension operator in (28) requires $\mathcal{O}(E)$ operations, the stochastic version of the d-CDP operation that utilizes the scheme described above requires $\mathcal{O}\big(X(WE + Y)\big)$ operations. On the other hand, the stochastic version of the d-DP operation, described by

$$\mathcal{T}_{\mathrm{s}}^{\mathrm{d}}[J^{\mathrm{d}}](x) := \min_{u \in \mathbb{U}^{\mathrm{g}}} \left\{ C(x, u) + \mathbb{E}_w \left[\widetilde{J^{\mathrm{d}}}\big(f(x, u) + w\big) \right] \right\}, \quad x \in \mathbb{X}^{\mathrm{g}},$$

has a time complexity of $\mathcal{O}(XUWE)$.

4.3.2. *Numerical computation of C_x^*.* Assumption 4.3 on the availability of the conjugate function C_x^* (22) of the stage cost can be restrictive. Alternatively, we can use the approximate discrete conjugation to compute C_x^* numerically as described by the following scheme:

- **Step 1.** For each $x \in \mathbb{X}^{\mathrm{g}}$:
  1.a. compute/evaluate $C_x^{\mathrm{d}} = C^{\mathrm{d}}(x, \cdot) : \mathbb{U}^{\mathrm{g}} \to \overline{\mathbb{R}}$, where $\mathbb{U}^{\mathrm{g}}$ is a discretization of $\mathbb{U}$;
  1.b. construct the dual grid $\mathbb{V}^{\mathrm{g}}(x)$ using the method described below; and,
  1.c. apply LLT to compute $C_x^{\mathrm{d}*\mathrm{d}} : \mathbb{V}^{\mathrm{g}}(x) \to \mathbb{R}$ using the data points $C_x^{\mathrm{d}} : \mathbb{U}^{\mathrm{g}} \to \overline{\mathbb{R}}$.
- **Step 2.** For each $y \in \mathbb{Y}^{\mathrm{g}}$: use LERP to compute $\overline{C_x^{\mathrm{d}*\mathrm{d}}}(-f_{\mathrm{i}}(x)^\top y)$ from the data points $C_x^{\mathrm{d}*\mathrm{d}} : \mathbb{V}^{\mathrm{g}}(x) \to \mathbb{R}$, and use the result in Algorithm 1:3 as an approximation of $C_x^*(-f_{\mathrm{i}}(x)^\top y)$.

This scheme introduces some error that mainly depends on the grids $\mathbb{U}^{\mathrm{g}}$ and $\mathbb{V}^{\mathrm{g}}(x)$ used for the discretization of the input space and its dual domain, respectively. Indeed, we can use Lemmas 2.5 and Corollary 2.7 to bound this error. We now use those results to provide some guidelines on the construction of the dual grids $\mathbb{V}^{\mathrm{g}}(x)$ for each $x \in \mathbb{X}^{\mathrm{g}}$. By Corollary 2.7, we can either (i) construct $\mathbb{V}^{\mathrm{g}}(x)$ *dynamically* such that $\mathrm{co}\big(\mathbb{V}^{\mathrm{g}}(x)\big) \supseteq -f_{\mathrm{i}}(x)^\top \mathbb{Y}^{\mathrm{g}}$ at each iteration, or (ii) construct a *fixed* grid $\mathbb{V}^{\mathrm{g}}(x)$ such that $\mathrm{co}\big(\mathbb{V}_{\mathrm{sub}}^{\mathrm{g}}(x)\big) \supseteq \mathbb{L}(C_x^{\mathrm{d}})$. The former requires $\mathcal{O}(XY)$ operations *per iteration*, while the latter has a *one-time* computational cost of $\mathcal{O}(XU)$ (assuming $C_x^{\mathrm{d}} : \mathbb{U}^{\mathrm{g}} \to \mathbb{R}$ is real-valued; see also Remark 4.8). For this reason, we use the second construction. Then, the problem reduces to computing the range of slopes of C_x^{d}. In this regard, notice that the function $C(x, \cdot)$ is convex (see Setting 1-(iii)), and hence its discretization is convex-extensible. Indeed, for such functions, it is possible to compute the range of slopes with an acceptable cost as explained in the following remark.

Remark 4.8 (Construction of $\mathbb{V}^{\mathrm{g}}(x)$ for $x \in \mathbb{X}^{\mathrm{g}}$)**.** *For the convex-extensible function $C_x^{\mathrm{d}} = C^{\mathrm{d}}(x, \cdot) : \mathbb{U}^{\mathrm{g}} \to \overline{\mathbb{R}}$, take $\mathrm{L}_i^-(C_x^{\mathrm{d}})$ (resp. $\mathrm{L}_i^+(C_x^{\mathrm{d}})$) to be the minimum finite first forward (resp. maximum finite last backward) difference of C_x^{d} along each dimension $i = 1, 2, \ldots, m$. Then, construct the dual grid $\mathbb{V}^{\mathrm{g}}(x) = \Pi_{i=1}^m \mathbb{V}_i^{\mathrm{g}}(x) \subset \mathbb{R}^m$ such that in each dimension i, the set $\mathbb{V}_i^{\mathrm{g}}(x) \subset \mathbb{R}$ contains at least one* *element that is less (resp. greater) than* $\mathrm{L}_i^-(C_x^{\mathrm{d}})$ *(resp.* $\mathrm{L}_i^+(C_x^{\mathrm{d}})$*), so that* $\mathrm{co}\big(\mathbb{V}_{\mathrm{sub}}^{\mathrm{g}}(x)\big) \supseteq \mathbb{L}(C_x^{\mathrm{d}})$*. If* $C_x^{\mathrm{d}} : \mathbb{U}^{\mathrm{g}} \to \mathbb{R}$ *is real-valued, this construction of* $\mathbb{V}^{\mathrm{g}}(x)$ *has a complexity of* $\mathcal{O}(U)$*.*

The proposed numerical scheme also increases the computational cost of the d-CDP algorithm. However, considering the construction described above, the first step of the scheme is carried out once in a multistep implementation of the d-CDP algorithm. In particular, assuming the grids $\mathbb{V}^{\mathrm{g}}(x)$, $x \in \mathbb{X}^{\mathrm{g}}$, are all of the same size V, for the T-step implementation of d-CDP algorithm which uses the scheme described above to compute C_x^* numerically,

- Step 1 introduces a *one-time* computational cost of $\mathcal{O}(X(U+V))$, and,
- Step 2 increases the computational cost of the d-CDP algorithm to $\widetilde{\mathcal{O}}(XY)$ *per iteration.*

Hence, the extension of the d-CDP Algorithm 1 that computes C_x^* numerically has a complexity of at most $\widetilde{\mathcal{O}}\big(X(Y+U+V)\big)$ per iteration.

5. Reducing Complexity by Exploiting Structure

In this section, we focus on a specific subclass of the optimal control problems considered in this study. In particular, we exploit the problem structure in this subclass in order to reduce the computational cost of the d-CDP algorithm. In this regard, a closer look to Algorithm 1 reveals a computational bottleneck in its numerical implementation: computing $\varphi_x^{\mathrm{d}} : \mathbb{Y}^{\mathrm{g}} \to \mathbb{R}$, and its conjugate within the `for loop` over $x \in \mathbb{X}^{\mathrm{g}}$. This step is indeed the dominating factor in the time complexity of $\mathcal{O}(XY)$ of Algorithm 1; see Appendix B.5 for the proof of Theorem 4.4. Hence, if the structure of the problem allows for this computations to be carried out outside the `for loop` over $x \in \mathbb{X}^{\mathrm{g}}$, then a significant reduction in the time complexity is achievable. This is indeed possible for problems with separable data in the state and input variables. We again consider deterministic systems in our development and assume that the conjugate of (input-dependent) stage cost is analytically available (see Assumption 5.1 below). However, the same extensions discussed in Section 4.3 for handling additive noise in the dynamics and numerical computation the conjugate of (input-dependent) stage cost can be used for the modified d-CDP algorithm to be described. The pseudo-code for the multistep implementation of the modified d-CDP algorithm including these extensions is provided in Appendix C.1.

5.1. Modified d-CDP algorithm

Consider the following subclass of the problems described in Setting 1.

Setting 2. *In addition to Setting 1, the dynamics and costs have the following properties:*

(i) **Dynamics.** *The input dynamics is state-independent, i.e.,* $f_{\mathrm{i}}(\cdot) = B \in \mathbb{R}^{n \times m}$*.*
(ii) **Cost functions.** *The stage cost is separable in the state and input variables, that is,* $C(x,u) = C_{\mathrm{s}}(x) + C_{\mathrm{i}}(u)$*, where* $C_{\mathrm{s}} : \mathbb{X} \to \mathbb{R}$ *and* $C_{\mathrm{i}} : \mathbb{U} \to \mathbb{R}$*.*

Note that the separability of the stage cost C implies that the constraints are also separable, i.e, there are no state-dependent input constraints. Under the conditions above, the state-dependent part of the stage cost (C_{s}) can be taken out of the minimization in the DP operator (16) as follows

$$\mathcal{T}[J](x) = C_{\mathrm{s}}(x) + \min_u \left\{ C_{\mathrm{i}}(u) + J\big(f(x,u)\big) \right\}, \quad x \in \mathbb{X}^{\mathrm{g}}. \tag{29}$$

Following the same dualization and then discretization procedure described in Section 4.1, we can derive the corresponding d-CDP operator

$$\widehat{\mathcal{T}}^{\mathrm{d}}[J^{\mathrm{d}}](x) = C_{\mathrm{s}}(x) + \varphi^{\mathrm{d}*}\big(f_{\mathrm{s}}(x)\big), \quad x \in \mathbb{X}^{\mathrm{g}}, \tag{30a}$$

$$\varphi^{\mathrm{d}}(y) := C_{\mathrm{i}}^*(-B^\top y) + J^{\mathrm{d}*\mathrm{d}}(y), \quad y \in \mathbb{Y}^{\mathrm{g}}. \tag{30b}$$

Here, again, we assume that the conjugate of the (input-dependent) stage cost is analytically available (similar to Assumption 4.3, now in the context posed by Setting 2).

Assumption 5.1 (Conjugate of input-dependent stage cost)**.** *The conjugate function $C_{\mathrm{i}}^*(v) = \max_u\{\langle v, u\rangle - C_{\mathrm{i}}(u)\}$ is analytically available; that is, the complexity of evaluating $C_{\mathrm{i}}^*(v)$ for each $v \in \mathbb{R}^m$ is of $\mathcal{O}(1)$.*

Notice how the function φ^{d} in (30b) is now independent of the state variable x. This means that the computation of φ^{d} requires $\mathcal{O}(Y)$ operations; cf. the `for loop` over $x \in \mathbb{X}^{\mathrm{g}}$ for the computation of φ_x^{d} in Algorithm 1. What remains to be addressed is the computation of the conjugate function $\varphi^{\mathrm{d}*}\big(f_{\mathrm{s}}(x)\big) = \max_{y \in \mathbb{Y}^{\mathrm{g}}}\{\langle f_{\mathrm{s}}(x), y\rangle - \varphi(y)\}$ for $x \in \mathbb{X}^{\mathrm{g}}$ in (30a). The straightforward maximization via enumeration over $y \in \mathbb{Y}^{\mathrm{g}}$ for each $x \in \mathbb{X}^{\mathrm{g}}$ (as in Algorithm 1) again leads to a time complexity of $\mathcal{O}(XY)$. The key idea here is to again use approximate discrete conjugatation:

- **Step 1.** Use LLT to compute $\varphi^{\mathrm{d}*\mathrm{d}} : \mathbb{Z}^{\mathrm{g}} \to \mathbb{R}$ from the data points $\varphi^{\mathrm{d}} : \mathbb{Y}^{\mathrm{g}} \to \mathbb{R}$, for a fixed grid $\mathbb{Z}^{\mathrm{g}}$.
- **Step 2.** For each $x \in \mathbb{X}^{\mathrm{g}}$, use LERP to compute $\overline{\varphi^{\mathrm{d}*\mathrm{d}}}\big(f_{\mathrm{s}}(x)\big)$ from the data points $\varphi^{\mathrm{d}*\mathrm{d}} : \mathbb{Z}^{\mathrm{g}} \to \mathbb{R}$, as an approximation of $\varphi^{\mathrm{d}*}\big(f_{\mathrm{s}}(x)\big)$.

We will discuss the construction of the grid $\mathbb{Z}^{\mathrm{g}}$ in Section 5.2.2. With such an approximation, the d-CDP operator (30) *modifies* to

$$\widehat{\mathcal{T}}_{\mathrm{m}}^{\mathrm{d}}[J^{\mathrm{d}}](x) := C_{\mathrm{s}}(x) + \overline{\varphi^{\mathrm{d}*\mathrm{d}}}\big(f_{\mathrm{s}}(x)\big), \quad x \in \mathbb{X}^{\mathrm{g}}, \tag{31a}$$

$$\varphi^{\mathrm{d}*\mathrm{d}}(z) := \max_{y \in \mathbb{Y}^{\mathrm{g}}}\{\langle z, y\rangle - \varphi(y)\}, \quad z \in \mathbb{Z}^{\mathrm{g}}, \tag{31b}$$

$$\varphi^{\mathrm{d}}(y) := C_{\mathrm{i}}^*(-B^\top y) + J^{\mathrm{d}*\mathrm{d}}(y), \quad y \in \mathbb{Y}^{\mathrm{g}}. \tag{31c}$$

Algorithm 2 provides the pseudo-code for the numerical scheme described above.

5.2. Analysis of modified d-CDP algorithm

5.2.1. *Complexity.*

We again begin with the time complexity of the proposed algorithm.

Theorem 5.2 (Complexity of modified d-CDP – Algorithm 2)**.** *Let Assumptions 2.4 and 5.1 hold. Then, the computation of the modified d-CDP operator (31) via Algorithm 2 has a time complexity of $\widetilde{\mathcal{O}}(X + Y + Z)$.*

Proof. See Appendix B.8. □

Once again, we note that in the T-step application of Algorithm 2 for solving the value iteration Problem 3.1, the time complexity increases linearly with the horizon T (assuming that the grids $\mathbb{Y}^{\mathrm{g}}$ and $\mathbb{Z}^{\mathrm{g}}$ can be constructed with at most $\mathcal{O}(X)$ operations; see also Remarks 4.7 and 5.4).

Algorithm 2 Implementation of the modified d-CDP operator (31) for Setting 2.

Input: dynamics $f_\mathrm{s} : \mathbb{R}^n \to \mathbb{R}^n,\ B \in \mathbb{R}^{n\times m}$;
discrete cost-to-go (at $t+1$) $J^\mathrm{d} : \mathbb{X}^\mathrm{g} \to \mathbb{R}$;
state-dependent stage cost $C_\mathrm{s}(x) : \mathbb{R}^n \to \mathbb{R}$;
conjugate of input-dependent stage cost $C_\mathrm{i}^* : \mathbb{R}^m \to \mathbb{R}$;
grids $\mathbb{Y}^\mathrm{g}, \mathbb{Z}^\mathrm{g} \subset \mathbb{R}^n$.
Output: discrete cost-to-go (at t) $\widehat{\mathcal{T}}_\mathrm{m}^\mathrm{d}[J^\mathrm{d}](x) : \mathbb{X}^\mathrm{g} \to \mathbb{R}$.

use LLT to compute $J^{\mathrm{d}*\mathrm{d}} : \mathbb{Y}^\mathrm{g} \to \mathbb{R}$ from $J^\mathrm{d} : \mathbb{X}^\mathrm{g} \to \mathbb{R}$;
$\varphi^\mathrm{d}(y) \leftarrow C_\mathrm{i}^*(-B^\top y) + J^{\mathrm{d}*\mathrm{d}}(y)$ for $y \in \mathbb{Y}^\mathrm{g}$;
use LLT to compute $\varphi^{\mathrm{d}*\mathrm{d}} : \mathbb{Z}^\mathrm{g} \to \mathbb{R}$ from $\varphi^\mathrm{d} : \mathbb{Y}^\mathrm{g} \to \mathbb{R}$;
for each $x \in \mathbb{X}^\mathrm{g}$ **do**
  use LERP to compute $\overline{\varphi^{\mathrm{d}*\mathrm{d}}}\big(f_\mathrm{s}(x)\big)$ from $\varphi^{\mathrm{d}*\mathrm{d}} : \mathbb{Z}^\mathrm{g} \to \mathbb{R}$;
  $\widehat{\mathcal{T}}_\mathrm{m}^\mathrm{d}[J^\mathrm{d}](x) \leftarrow C_\mathrm{s}(x) + \overline{\varphi^{\mathrm{d}*\mathrm{d}}}\big(f_\mathrm{s}(x)\big)$;
end for

We now compare the time complexity of d-CDP Algorithm 2 with that of d-DP algorithm and d-CDP Algorithm 1. In order to explicitly observe the reduction from quadratic complexity to (log-)linear complexity, let us assume that all the involved grids ($\mathbb{X}^\mathrm{g}$, $\mathbb{Y}^\mathrm{g}$, and $\mathbb{Z}^\mathrm{g}$) are of the same size, i.e., $Y, Z = X$ (this is also consistent with Assumption 2.4). Then, the complexity of d-CDP Algorithm 1 is of $\mathcal{O}(X^2)$, while the complexity of d-CDP Algorithm 2 is of $\widetilde{\mathcal{O}}(X)$. The same reduction can be seen in the complexity of finding a suboptimal control sequence for a given initial state. In particular, if we use the greedy policy (19) w.r.t. the discrete costs-to-go J_t^d computed using Algorithm 2, the total complexity of computing the control sequence is of $\widetilde{\mathcal{O}}\big(T(X+Y+Z+UE)\big) = \widetilde{\mathcal{O}}\big(T(X+UE)\big)$, where E represents the complexity of the extension operation used in (19). This is a reduction from quadratic to linear complexity compared to both d-DP algorithm and d-CDP Algorithm 1.

Finally, let us discuss the implications of the extensions of the Section 4.3 on the time complexity of d-CDP Algorithm 2. Again, assume $Y, Z = X$. First, regarding handling additive noise in the dynamics, we can follow the exact same procedure explained in Section 4.3.1. In particular, for a disturbance with the finite support $\mathbb{W}^\mathrm{d}$, the stochastic version of the modified d-CDP operator (31) requires $\widetilde{\mathcal{O}}(XWE + Y + Z) = \widetilde{\mathcal{O}}(XWE)$ operations in each iteration. Second, for the extension concerning numerical computation of the conjugate C_i^* of the input-dependent cost, we can also follow the scheme described in Section 4.3.2. However, since the function is now independent of the state, the two steps of that scheme also become independent of x. This leads to a one-time computational cost of $\mathcal{O}(U+V)$ for Step 1, and a per iteration computational cost of $\widetilde{\mathcal{O}}(Y)$ for Step 2. Hence, the extended version of the modified d-CDP Algorithm 2 that computes C_i^* numerically, has at most a complexity of $\widetilde{\mathcal{O}}(X+Y+Z+U+V) = \widetilde{\mathcal{O}}(X+U)$, assuming also $V = U$. Once again, in both cases, we see a reduction from quadratic complexity to linear complexity compared to both d-DP algorithm and d-CDP Algorithm 1.

5.2.2. *Error.* We next consider the error of the proposed algorithm by providing a bound on the difference between the modified d-CDP operator (31) and the DP operator (29).

Theorem 5.3 (Error of modified d-CDP – Algorithm 2)**.** *Consider the DP operator* $\mathcal{T}$ (29) *and the implementation of the modified d-CDP operator* $\widehat{\mathcal{T}}_{\mathrm{m}}^{\mathrm{d}}$ (31) *via Algorithm 2. Assume that* $J : \mathbb{X} \to \mathbb{R}$ *is a Lipschitz continuous, convex function, and that the grid* $\mathbb{Z}^{\mathrm{g}}$ *in Algorthim 2 is such that* $\mathrm{co}(\mathbb{Z}^{\mathrm{g}}) \supset f_{\mathrm{s}}(\mathbb{X}^{\mathrm{g}})$*. Then, for each* $x \in \mathbb{X}^{\mathrm{g}}$*, we have*

$$-\big(e_2 + e_3\big) \le \mathcal{T}[J](x) - \widehat{\mathcal{T}}_{\mathrm{m}}^{\mathrm{d}}[J^{\mathrm{d}}](x) \le e_1^m(x), \tag{32}$$

where

$$\begin{aligned} & e_1^m(x) := \big[\, \|f_{\mathrm{s}}(x)\| + \|B\| \cdot \Delta_{\mathbb{U}} + \Delta_{\mathbb{X}} \big] \cdot \mathrm{d}\left(\partial\big(\mathcal{T}[J] - C_{\mathrm{s}}\big)(x), \mathbb{Y}^{\mathrm{g}}\right), \\ & e_2 = [\Delta_{\mathbb{Y}^{\mathrm{g}}} + \mathrm{L}(J)] \cdot \mathrm{d_H}(\mathbb{X}, \mathbb{X}^{\mathrm{g}}), \\ & e_3 = \Delta_{\mathbb{Y}^{\mathrm{g}}} \cdot \mathrm{d_H}\left(f_{\mathrm{s}}(\mathbb{X}^{\mathrm{g}}), \mathbb{Z}^{\mathrm{g}}\right). \end{aligned} \tag{33}$$

Proof. See Appendix B.9. □

Once again, the three terms capture the errors due to discretization in y, x, and z, respectively. We now use this result to provide some guidelines on the construction of the required grids. Concerning the grid $\mathbb{Y}^{\mathrm{g}}$, because of the error term e_1^m, similar guidelines to the ones provided in Section 4.2.2 apply here. In particular, notice that the first error term e_1^m (33) now depends on $\mathrm{d}\left(\partial\big(\mathcal{T}[J] - C_{\mathrm{s}}\big)(x), \mathbb{Y}^{\mathrm{g}}\right)$, and hence in the construction of $\mathbb{Y}^{\mathrm{g}}$, we need to consider the range of slopes of $\mathcal{T}[J]-C_{\mathrm{s}}$. This essentially means using $C_{\mathrm{i}}^M = \max_{u\in\mathbb{U}} C_{\mathrm{i}}$ and $C_{\mathrm{i}}^m = \min_{u\in\mathbb{U}} C_{\mathrm{i}}$ instead of C^M and C^m, respectively, in Remark 4.7. Next to be addressed is the construction of the grid $\mathbb{Z}^{\mathrm{g}}$. Here, again, we are dealing with the issue of constructing the dual grid for approximate discrete conjugation. Then, by Corollary 2.7, we can either (i) construct a *fixed* grid $\mathbb{Z}^{\mathrm{g}}$ such that $\mathrm{co}(\mathbb{Z}^{\mathrm{g}}) \supset f_{\mathrm{s}}(\mathbb{X}^{\mathrm{g}})$, or (ii) construct $\mathbb{Z}^{\mathrm{g}}$ *dynamically* such that $\mathrm{co}(\mathbb{Z}^{\mathrm{g}}_{\mathrm{sub}}) \supseteq \mathbb{L}(\varphi^{\mathrm{d}})$ at each iteration. The former has a *one-time* computational cost of $\mathcal{O}(X)$, while the latter requires $\mathcal{O}(Y)$ operations *per iteration*. For this reason, as also assumed Theorem 5.3, we use the first method to construction $\mathbb{Z}^{\mathrm{g}}$. The following remark summarizes this discussion.

Remark 5.4 (Construction of $\mathbb{Z}^{\mathrm{g}}$)**.** *Construct the grid* $\mathbb{Z}^{\mathrm{g}}$ *such that* $\mathrm{co}(\mathbb{Z}^{\mathrm{g}}) \supset f_{\mathrm{s}}(\mathbb{X}^{\mathrm{g}})$*. This can be done by finding the vertices of the smallest hyper-rectangle that contains the set* $f_{\mathrm{s}}\big(\mathbb{X}^{\mathrm{g}}\big)$*. Such a construction has a* one-time *computational cost of* $\mathcal{O}(X)$*.*

6. **Numerical Experiments**

In this section, we examine the performance of the proposed d-CDP algorithms in comparison with the d-DP algorithm through two synthetic numerical examples. In particular, we use these numerical examples to verify our theoretical results on the complexity and error of the proposed algorithms. Here, we focus on the performance of the basic algorithms for deterministic systems for which the conjugate of the (input-dependent) stage cost is analytically available (see Assumptions 4.3 and 5.1). The results of the numerical simulations of the extended versions of these algorithms are provided in Appendix C.1. The application of these algorithms in solving the optimal control problem for a simple epidemic model and a noisy inverted pendulum are showcased in Appendix C.2. Finally, we note that all the simulations presented in this article were implemented via MATLAB version R2017b, on a PC with Intel Xeon 3.60 GHz processor and 16 GB RAM.

Throughout this section, we consider a linear system with two states and two inputs described by

$$x_{t+1} = \begin{bmatrix} -0.5 & 2 \\ 1 & 3 \end{bmatrix} x_t + \begin{bmatrix} 1 & 0.5 \\ 1 & 1 \end{bmatrix} u_t, \tag{34}$$

over the finite horizon $T = 10$, with the following state and input constraints

$$x_t \in \mathbb{X} = [-1, 1]^2 \subset \mathbb{R}^2, \quad u_t \in \mathbb{U} = [-2, 2]^2 \subset \mathbb{R}^2. \tag{35}$$

Moreover, we consider *quadratic state cost* and *exponential input cost* as follows

$$C_{\mathrm{s}}(x) = C_T(x) = \|x\|^2, \quad C_{\mathrm{i}}(u) = e^{|u_1|} + e^{|u_2|} - 2. \tag{36}$$

We use *uniform* grid-like discretizations $\mathbb{X}^{\mathrm{g}} \subset \mathbb{X}$ and $\mathbb{U}^{\mathrm{g}} \subset \mathbb{U}$ for the state and input spaces, respectively. All of the other grids ($\mathbb{Y}^{\mathrm{g}}$ and $\mathbb{Z}^{\mathrm{g}}$) involved in the proposed d-CDP algorithms are also constructed *uniformly*, according to the guidelines provided in Remarks 4.7 and 5.4. We are particularly interested in the performance (error and time complexity) of the d-CDP algorithms in comparison with the d-DP algorithm, as the size of these discrete sets increases. Considering the fact that all the involved grids are constructed uniformly, and all the extension operations are handled via LERP, we can summarize the computational complexities as follows.

Remark 6.1 (Comparison of complexities)**.** *Assume that all the finite sets involved in the d-DP and d-CDP algorithms* ($\mathbb{X}^{\mathrm{g}}, \mathbb{U}^{\mathrm{g}}, \mathbb{Y}^{\mathrm{g}}, \mathbb{Z}^{\mathrm{g}}$) *are* uniform *grids, and that all the extension operations in these algorithms are handled via* LERP*. Then, the complexity of finding a suboptimal input sequence in a T-step optimal control problem for a given initial state is of*

- *(i)* $\mathcal{O}(TXU)$ *for d-DP algorithm,*
- *(ii)* $\mathcal{O}\big(T(XY + U)\big)$ *for d-CDP Algorithm 1,*
- *(iii)* $\mathcal{O}\big(T(X + Y + Z + U)\big)$ *for d-CDP Algorithm 2.*

6.1. **Numerical study of Algorithm 1**

In this section, in addition to the constraints (35), we also consider the following constraint

$$h(x_t, u_t) = x_t + u_t - (2, 2)^\top \le 0, \tag{37}$$

where $h : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$ describes the state-dependent input constraints. Corresponding to the notation of Section 4, the stage and terminal costs are respectively given by (see (36))

$$C(x, u) = \|x\|^2 + e^{|u_1|} + e^{|u_2|} - 2, \quad C_T(x) = \|x\|^2,$$

subject to (35) and (37). The conjugate of the stage cost is also analytically available and given by

$$C_x^*(v) = 1 - \|x\|^2 + \langle \hat{u}, v \rangle - e^{|\hat{u}_1|} - e^{|\hat{u}_2|}, \quad v \in \mathbb{R}^2,$$

where

$$\hat{u}_i = \begin{cases} \max\big\{ -2,\ \min\{2 - x_i,\ \mathrm{sgn}(v_i) \ln |v_i|\} \big\}, & v_i \neq 0, \\ 0, & v_i = 0, \end{cases} \qquad i = 1, 2.$$

We note that, for construction of $\mathbb{Y}^{\mathrm{g}}$, we use the method described in Remark 4.7 with $\alpha = 1$.

We begin with examining the error in the d-DP and d-CDP algorithms w.r.t. the "reference" costs-to-go $J_t^\star : \mathbb{X} \to \mathbb{R}$. These reference costs $J_t^\star$ are computed numerically via a high resolution application of the d-DP algorithm with $X, U = 81^2$. Figure 2 depicts the maximum absolute error

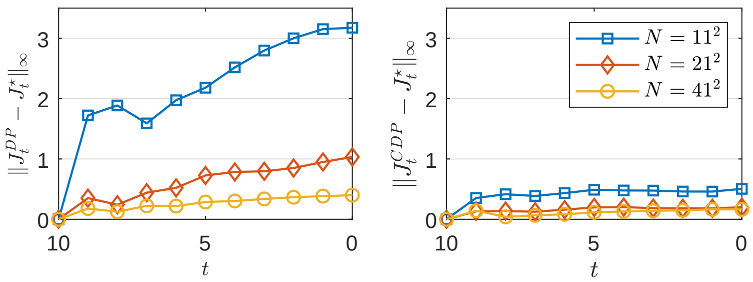


FIGURE 2. Error of Algorithm 1: the maximum absolute error (over $x \in \mathbb{X}^{\mathrm{g}}$) in the computed discrete costs-to-go for different grid sizes $(X, Y, U = N)$. Notice the time axis is backward.

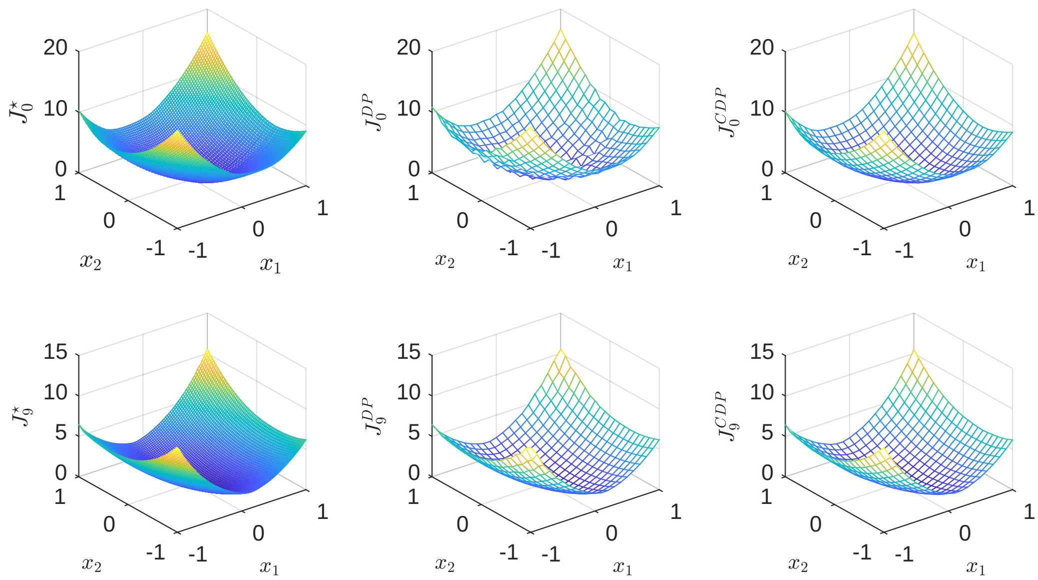


FIGURE 3. Error of Algorithm 1: the costs-to-go with $N = 21^2$ at $t = 0$ (top) and $t = 9$ (bottom).

in the discrete cost functions J_t^{d} computed using these algorithms over the horizon. As expected and in line with our error analysis (Theorem 4.6 and Proposition A.1), using a finer discretization scheme with larger $N = X, Y, U$, leads to a smaller error. Moreover, for each gird size, a general increase is seen in the error, over the time steps in the backward iteration, due to accumulation of error. For further illustration, Figure 3 shows the corresponding costs-to-go at $t = 0$ and $t = 9$, with $N = 21^2$. Notice that, since the stage and terminal costs are convex and the dynamics is linear, the costs-to-go are also convex. As can be seen in Figure 3, while the d-CDP algorithm preserves the convexity, the d-DP algorithm outputs non-convex costs-to-go (due to the application of LERP). In particular, notice how J_0^{DP} is not a convex-extensible function, while J_0^{CDP} is convex-extensible

We next compare the performance of the d-DP and d-CDP algorithms for solving instances of the optimal control problem, using the cost functions derived from the backward value iteration. In this regards, recall that the d-DP algorithm also provides us with the control laws $\mu_t^{\mathrm{d}} : \mathbb{X}^{\mathrm{g}} \to \mathbb{U}^{\mathrm{g}}$. Hence, we report the performance of the d-DP algorithm for the two following cases:

(i) d-DP (J) denoting the case where the control sequence $\mathbf{u}^\star(x_0)$ is derived according to (19), i.e., using the greedy policy w.r.t. the costs $J_t^{\rm d}: \mathbb{X}^{\rm g} \to \mathbb{R}$.

(ii) d-DP (μ) denoting the case where the control sequence $\mathbf{u}^\star(x_0)$ is derived according to (20), i.e., via LERP extension of the control laws $\mu_t^{\rm d}: \mathbb{X}^{\rm g} \to \mathbb{U}^{\rm g}$.

Table 2 reports the average relative cost (w.r.t. the trajectory cost of d-DP (μ) with $N = 81^2$) and the average total run-time (the running time of the backward value iteration plus the running time of the forward computation of control sequence for each initial condition) for different grid sizes. The average is taken over ten instances of the optimal control problem with random initial conditions, chosen uniformly from $\mathbb{X} = [-1, 1]^2$. Comparing the first two rows of Table 2, we see that d-CDP has a slightly better performance compared to d-DP (J) w.r.t. both the trajectory cost and the running time. Indeed, by Remark 6.1, the time complexity of both algorithms is of $\mathcal{O}(TN^2)$, which matches the reported running times. In this regard, we also note that the backward value iteration is the absolutely dominant factor in the reported running times. (Effectively, the reported numbers can be taken to be the run-time of the backward value iteration).

On the other hand, as reported in the last row of Table 2, d-DP (μ) achieves a significantly better performance with regard to the cost of the controlled trajectories. This is because the control input computed using this approach is smooth, while the control input computed using the cost functions (in the first two rows of Table 2) is limited to the discrete input space considered in the forward minimization problems. The d-CDP algorithm, however, gives us an extra degree of freedom for the size Y of the dual grid. Then, if the the cost functions are "compactly representable" in the dual domain via their slopes, we can reduce the time complexity by using a more coarse grid $\mathbb{Y}^{\rm g}$, with a limited effect on the "quality" of computed cost functions. Indeed, as reported in the first

TABLE 2. Performance of the d-CDP Algorithm 1 and the d-DP algorithm for different grid sizes ($X, Y, U = N$): The reported numbers are the average relative trajectory cost (left – blue), and the average total running time (right – red); see the text for more details.

Relative trajectory cost / Running time (seconds)				
Alg. \ N	11^2	21^2	41^2	81^2
d-CDP	2.06 / 3.4e0	1.49 / 3.8e1	1.13 / 5.5e2	1.04 / 8.1e3
d-DP (J)	2.30 / 6.3e0	1.56 / 8.0e1	1.15 / 1.2e3	1.04 / 1.8e4
d-DP (μ)	1.55 / 6.2e0	1.20 / 8.0e1	1.04 / 1.2e3	1 / 1.8e4

TABLE 3. Performance of the d-CDP Algorithm 1 using modified grid sizes (X, Y, U); cf. Table 2.

Relative trajectory cost / Running time (seconds)			
(X, Y, U)	$(11^2, 7^2, 11^2)$	$(21^2, 11^2, 21^2)$	$(41^2, 21^2, 41^2)$
d-CDP	2.07 / 1.9e0	1.52 / 1.5e1	1.13 / 1.9e2
(X, Y, U)	$(11^2, 7^2, 21^2)$	$(21^2, 11^2, 41^2)$	$(41^2, 21^2, 81^2)$
d-CDP	1.48 / 2.6e0	1.12 / 2.5e1	1.02 / 3.5e2

three cases in Table 3, for solving the same ten instances of the considered optimal control problem, we can reduce the size of the dual grid by a factor of 4, and hence reduce the running time of d-CDP algorithm, while achieving approximately the same average relative cost in the controlled trajectories. This, in turn, allows us to increase the size U of the discrete input space $\mathbb{U}^{\mathrm{g}}$ in order to generate a finer control input. Doing so, we can achieve a smaller cost, while keeping the total running time at approximately the same level. This effect can also be seen in Table 3, where the d-CDP algorithm achieves a similar performance to that of d-DP (μ); cf. the last row of Table 2. However, we note that such a modification in the size of the grids leads to an increase in the running time of the forward iteration.

6.2. **Numerical study of Algorithm 2**

We now focus on the performance of the d-CDP Algorithm 2 for solving the optimal control problem described by the dynamics (34), constraints (35), and costs (36). The conjugate of the input-dependent stage cost now reads as

$$C_{\mathrm{i}}^{*}(v) = 1 + \langle \hat{u}, v \rangle - e^{|\hat{u}_1|} - e^{|\hat{u}_2|}, \quad v \in \mathbb{R}^2, \tag{38}$$

where

$$\hat{u}_i = \begin{cases} \max\big\{ -2,\ \min\{2,\ \mathrm{sgn}(v_i)\ln|v_i|\}\big\}, & v_i \neq 0, \\ 0, & v_i = 0, \end{cases} \qquad i = 1, 2.$$

For construction of $\mathbb{Y}^{\mathrm{g}}$, we again use the scheme described in Remark 4.7 with $\alpha = 1$, taking into account the modification mentioned in Section 5.2.2. Also, the grid $\mathbb{Z}^{\mathrm{g}}$ is constructed according to Remark 5.4. We use the same set up as before for examining the performance of the d-CDP algorithm in comparison with the d-DP algorithm. Figure 4 depicts the maximum absolute error in the discrete costs-to-go J_t^{d} computed using these algorithms over the horizon w.r.t. the reference $J_t^{\star}$ (i.e., the output of the d-DP algorithm with $X, U = 81^2$). In Table 4, we report the average relative cost and the average total run-time for different grid sizes (the average is taken over ten randomly chosen instances of the optimal control problem). Once again, in line with our error analysis (Theorem 5.3 and Proposition A.1), using finer grids leads to smaller errors. Also, the running times reported in Table 4 correspond to the the complexities given in Remark 6.1, i.e., $\mathcal{O}(TN^2)$ for d-DP algorithm and $\mathcal{O}(TN)$ for d-CDP Algorithm 2.

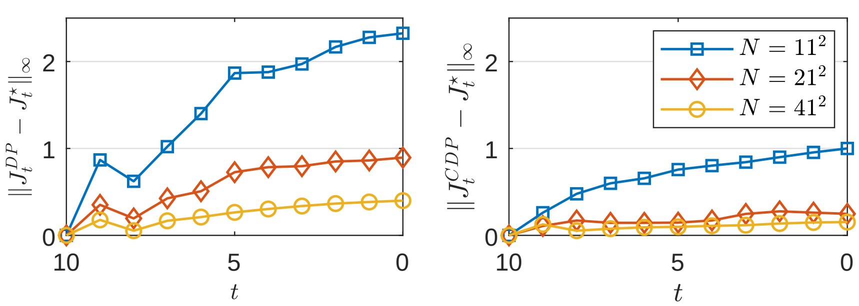


FIGURE 4. Error of Algorithm 2: the maximum absolute error (over $x \in \mathbb{X}^{\mathrm{g}}$) in the computed discrete costs-to-go for different grid sizes ($X, Y, Z, U = N$). Notice that the time axis is backward.

Table 4. Performance of the d-CDP Algorithm 2 and the d-DP algorithm for different grid sizes ($X, Y, Z, U = N$); see the caption of Table 2 and the text for more details.

	Relative trajectory cost / Running time (seconds)			
Alg. \ N	11^2	21^2	41^2	81^2
d-CDP	2.02 / 1.8e − 1	1.47 / 3.9e − 1	1.13 / 1.5e + 0	1.02 / 7.5e + 0
d-DP (J)	2.30 / 3.3e + 0	1.55 / 4.2e + 1	1.15 / 6.2e + 2	1.04 / 1.0e + 4
d-DP (μ)	1.56 / 3.3e + 0	1.20 / 4.2e + 1	1.03 / 6.2e + 2	1 / 1.0e + 4

Comparing the first two rows of Table 4, we see that the d-CDP algorithm achieves a lower average cost compared to the d-DP algorithm, with a significant reduction in the running time. In particular, notice how the lower complexity of d-CDP allows us to increase the size of the grids to $N = 81^2$, while keeping the running time at the same order as the d-DP algorithm with $N = 11^2$. Moreover, the lower time complexity of the d-CDP algorithm also allows us to achieve a lower average cost in comparison with the d-DP (μ) by using larger grid sizes in the d-CDP algorithm; e.g., compare the performance of d-CDP with $N = 41^2$ with that of d-DP (μ) with $N = 21^2$ in Table 4. However, such an increase in the grid sizes in the d-CDP algorithm implies an increase in the running time of the forward iteration, and also a higher memory requirement in comparison with the d-DP algorithm.

7. Further Remarks

In this final section, we discuss some of the algorithms available in the literature and their connection to the d-CDP algorithms. The limitations of the proposed algorithms and possible remedies to alleviate them are also discussed. We also mention possible extensions of the current work as future research directions.

7.1. Value iteration in the conjugate domain

Let us first note that the algorithms developed in this study involve two LLT transforms at the beginning and end of each step (see, e.g., lines 1 and 3 in Algorithm 2). Hence, the possibility of a perfect transformation of the minimization in the primal domain to a simple addition in the conjugate domain is interesting since it allows for performing the value iteration completely in the conjugate domain for the *conjugate* of the costs-to-go. In other word, we can stay in the conjugate domain over multiple steps in time, and avoid the conjugate operation at the beginning of the intermediate steps. This, in turn, leads to a lower computational cost in multistep implementations. However, for such a perfect transformation to be possible, we need to impose further restrictions on the problem data. To be precise, on top of the properties laid out in Setting 2, we need

(i) the dynamics to be *linear*, i.e., $f(x, u) = Ax + Bu$, where the state matrix A is *invertible*,
(ii) and the stage cost to be *state-independent*, i.e., $C_s(x) = 0$, and hence $C(x, u) = C_i(u)$.

For systems satisfying these conditions, the DP operator reads as

$$\mathcal{T}[J](x) = \min_u \{C_i(u) + J(Ax + Bu)\}, \quad x \in \mathbb{X},$$

and its conjugate can be shown to be given by

$$\mathcal{T}[J]^*(y) = C_{\mathrm{i}}^*(-B^\top A^{-\top} y) + J^*(A^{-\top} y), \quad y \in \mathbb{R}^n.$$

Notice how the minimization in the DP operator in the primal domain perfectly transforms to an addition in the dual domain. This property indeed allows us to stay in the dual domain over multiple steps in time, while only computing the conjugate of the costs in the intermediate steps. The possibility of such a perfect transformation, accompanied by application of LLT for better time complexity, was first noticed in [12]. Indeed, there, the authors introduced the fast value iteration algorithm for a more restricted class of DP problems (besides the properties discussed above, they required, among other conditions, the state matrix A to be non-negative and monotone). In this regard, we also note that, as in [12], the possibility of staying in the conjugate domain over multiple steps is particularly interesting for infinite-horizon problems.

7.2. Relation to max-plus linear approximations

Recall the d-CDP reformulation

$$\widehat{\mathcal{T}}^{\mathrm{d}}[J^{\mathrm{d}}](x) = \min_u \left\{ C(x,u) + J^{\mathrm{d}*\mathrm{d}*}\big(f(x,u)\big) \right\},$$

in Proposition 4.5. Now, note that

$$J^{\mathrm{d}*\mathrm{d}*}(x) = \max_{y \in \mathbb{Y}^{\mathrm{g}}} \left\{ \langle x, y \rangle - J^{\mathrm{d}*}(y) \right\},$$

is a max-plus linear combination using the basis functions $\{\langle \cdot, y \rangle : y \in \mathbb{Y}^{\mathrm{g}}\}$ and coefficients $\{J^{\mathrm{d}*}(y) : y \in \mathbb{Y}^{\mathrm{g}}\}$. That is, d-CDP algorithm, similarly to the approximate value iteration algorithms in [3, 6], employs a max-plus approximation of J as a piece-wise affine function. The key difference in the proposed algorithms is however that by choosing a grid-like dual domain $\mathbb{Y}^{\mathrm{g}}$ (i.e., set of slopes for the basis functions used for approximations), we can incorporate the linear-time complexity of the LLT in our advantage in computing the coefficients $\{J^{\mathrm{d}*}(y) : y \in \mathbb{Y}^{\mathrm{g}}\}$. Moreover, instead of using a fixed basis, we incorporate a dynamic basis by updating the grid $\mathbb{Y}^{\mathrm{g}}$ at each iteration in order to reduce the error of the algorithm.

7.3. Gird-like discretization and curse of dimensionality

In this study, we exclusively considered grid-like discretizations of both primal and dual domains. This is particularly suitable for problem with (almost) boxed constraints on the state and input spaces (as illustrated in the numerical examples in Section 6). More importantly, such discretizations suffer from the so-called curse of dimensionality; that is, the size of the finite representations of the corresponding spaces increases exponentially with the dimension of those spaces. In this regard, we note that in order to enjoy the linear-time complexity of LLT, we are *only* required to choose a grid-like *dual* grid [23, Rem. 5]; that is, the discretization of the state and input spaces in the primal domain need not be grid-like. However, since the grid-like dual domain is usually chosen to include the same number of points as the primal domain in each dimension (see Assumption 2.4), we still face the curse of dimensionality. This, in particular, impairs the performance of the d-CDP Algorithm 1 for problems in which the dimension of the state space is greater than that of the input space; see the numerical example of Appendix C.2.1 for an illustration. Proper exploitation of the aforementioned property of LLT in such cases calls for a more efficient construction of the dual grid based on the provided data points in the primal domain.

7.4. **Preserving convexity in multistep implementation**

Recall that the proposed approach involves solving the dual problem corresponding to the minimization of the DP operation. As discussed before, due to this dualization, the d-CDP algorithm is essentially blind to non-convexity. In this regard, note that the convexity of the stage and terminal costs (as assumed in this article) is necessary but not sufficient for the costs-to-go J_t to be convex for all $t = 1, \dots, T$. In particular, for preservation of convexity in the proposed d-CDP algorithms, we also need the composition $J_t\big(f(x,u)\big)$ to be jointly convex in x and u, given that J_t is convex. This is, for example, the case when (the extended-value extension of) J_t is monotonic, and the dynamics f is convex in each argument [10, Sec. 3.2.4].

For Algorithm 2, there is another issue that further complicates the preservation of convexity in the multistep implementation: the approximate conjugation in the last step, particularly, the LERP extension in Algorithm 2:5. To see this, note that the mapping $x \mapsto \overline{\varphi^{\mathrm{d*d}}}\big(f_{\mathrm{s}}(x)\big)$ is not necessarily convex, despite the fact that the underlying discrete function $\varphi^{\mathrm{d*d}} : \mathbb{Z}^{\mathrm{g}} \to \mathbb{R}$ is convex-extensible by construction. A possible remedy for this last issue is to employ an operator (instead of LERP) that leads to convex extensions of convex-extensible functions. Another possibility is to avoid approximate conjugation by incorporating the ability of LLT to provide us with the optimizer map as described below.

7.5. **The optimizer map in LLT**

Consider a discrete function $h^{\mathrm{d}} : \mathbb{X}^{\mathrm{d}} \to \mathbb{R}$ and its discrete conjugate $h^{\mathrm{d*d}} : \mathbb{Y}^{\mathrm{g}} \to \mathbb{R}$ computed using LLT for some finite set $\mathbb{Y}^{\mathrm{g}}$. LLT is, in principle, capable of providing us with the optimizer mapping $x^\star : \mathbb{Y}^{\mathrm{g}} \to \mathbb{X}^{\mathrm{d}} : y \mapsto \operatorname{argmax}\{\langle x, y\rangle - h^{\mathrm{d}}(x)\}$, where for each $y \in \mathbb{Y}^{\mathrm{g}}$, we have $h^{\mathrm{d*d}}(y) = \langle x^\star(y), y\rangle - h^{\mathrm{d}}\big(x^\star(y)\big)$. This capability of LLT can be employed to address some of the drawbacks of the proposed d-CDP algorithm:

(i) Avoiding approximate conjugation: Let us first recall that by approximate (discrete) conjugation we mean that we first compute the conjugate function $h^{\mathrm{d*d}} : \mathbb{Y}^{\mathrm{g}} \to \mathbb{R}$ for some grid $\mathbb{Y}^{\mathrm{g}}$ using the data points $h^{\mathrm{d}} : \mathbb{X}^{\mathrm{d}} \to \mathbb{R}$, and then for any $\tilde{y}$ (not necessarily belonging to $\mathbb{Y}^{\mathrm{g}}$) we use the LERP extension $\overline{h^{\mathrm{d*d}}}(\tilde{y})$ as an approximation for $h^{\mathrm{d*}}(\tilde{y})$. This approximation scheme is used in Algorithm 2 (and all the extended algorithms in Appendix C.1 for computing the conjugate of the stage cost numerically). Indeed, it is possible to avoid this approximation and compute $h^{\mathrm{d*}}(\tilde{y})$ exactly by incorporating a smart search for the corresponding optimizer $\tilde{x} \in \mathbb{X}^{\mathrm{d}}$ for which $h^{\mathrm{d*}}(\tilde{y}) = \langle \tilde{x}, \tilde{y}\rangle - h(\tilde{x})$. To be precise, if $\tilde{y} \in \operatorname{co}(\widetilde{\mathbb{Y}}^{\mathrm{d}})$ for some subset $\widetilde{\mathbb{Y}}^{\mathrm{d}}$ of $\mathbb{Y}^{\mathrm{g}}$, then $\tilde{x} \in \operatorname{co}\big(x^\star(\widetilde{\mathbb{Y}}^{\mathrm{d}})\big)$, where $x^\star : \mathbb{Y}^{\mathrm{g}} \to \mathbb{X}^{\mathrm{d}}$ is the corresponding optimizer mapping. That is, in order to find the optimizer $\tilde{x} \in \mathbb{X}^{\mathrm{d}}$ corresponding to $\tilde{y}$, it suffices to search in the set $\mathbb{X}^{\mathrm{d}} \cap \operatorname{co}\big(x^\star(\widetilde{\mathbb{Y}}^{\mathrm{d}})\big)$, instead of the whole discrete primal domain $\mathbb{X}^{\mathrm{d}}$. This, in turn, can lead to lower time requirement for computing the exact discrete conjugate function.

(ii) Extracting the optimal policy within the d-CDP algorithm: As shown by our theoretical results and also confirmed by the numerical examples of Section 6, the d-CDP algorithms computationally outperforms the d-DP algorithm in solving the value iteration problem, i.e., computing the costs

$J_t^{\mathrm{d}} : \mathbb{X}^{\mathrm{g}} \to \mathbb{R},\ t = 0, 1, \dots, T-1$. On the other hand, the backward value iteration using the d-DP algorithm also provides us with control laws $\mu_t^{\mathrm{d}} : \mathbb{X}^{\mathrm{g}} \to \mathbb{U}^{\mathrm{g}},\ t = 0, 1, \dots, T-1$. This can potentially render the computation of the control sequence for a given initial condition less costly. In order to address this issue, we have to look at the possibility of extracting the optimal policy within the d-CDP algorithm. A promising approach is to keep track of the dual pairs in each conjugate transform, i.e., the pairs (x, y) for which $\langle x, y \rangle = h(x) + h^*(y)$. This indeed seems possible considering the capability of LLT in providing the optimizer mapping $x^\star : \mathbb{Y}^{\mathrm{g}} \to \mathbb{X}^{\mathrm{d}}$.

7.6. **Towards quantum dynamic programming**

Application of quantum computing for solving optimal control problems has attracted a lot of attentions recently. In particular, in [31], a quantum algorithm is proposed for solving the finite-horizon DP problem with a time complexity of $\mathcal{O}(X^{1/2} \cdot U^{9/2})$. Such a complexity is particularly attractive for problems with a huge state space and a relatively small action space, such as the travelling salesman problem. More related to our work is the recent introduction of the quantum mechanical implementation of the LLT algorithm for the discrete conjugate transform, which enjoys a poly-logarithmic complexity in the size of the discrete primal and dual domains [34]. An interesting feature of the d-CDP Algorithm 2 is that one can readily leverage this algorithm and develop a quantum mechanical version of the modified d-CDP algorithm for problems of Setting 2. In this regard, we note that Algorithm 2 consists of three main operations: (i) LLT (lines 1 and 3), (ii) addition (lines 2 and 6), and (iii) composite extended function query (line 5). In particular, by choosing $Z, Y = X$, all these operations can be handled with a log-linear complexity in the size of the discrete state space X, leading to a log-linear time complexity for the d-CDP Algorithm 2 (Theorem 5.2). The quantum algorithm proposed in [34], on the other hand, reduces the complexity of the LLT operations to $\mathcal{O}\big(\mathrm{poly}(\log X, \log Y)\big)$. To the best of our knowledge, the quantum-mechanical implementation of addition also has a poly-logarithmic complexity in the size of the input vectors. Thus, assuming that composite function query can also be handled quantum-mechanically with a similar logarithmic complexity, we envision a quantum version of the d-CDP Algorithm 2 with a poly-logarithmic complexity $\mathcal{O}\big(\mathrm{poly}(\log X)\big)$ in the size of the discrete state space. Such a reduction in the time complexity is particularly interesting since it can effectively address the infamous curse of dimensionality in the DP literature.

Appendix A. **Error of d-DP**

In this section, we consider the error in the d-DP operator w.r.t. the DP operator.

Proposition A.1 (Error of d-DP)**.** *Consider the DP operator $\mathcal{T}$ (16) and the d-DP operator $\mathcal{T}^{\mathrm{d}}$ (18). Assume that the functions J, $\widetilde{J^{\mathrm{d}}}$, and C are Lipschtiz continuous, and $\widetilde{J^{\mathrm{d}}}(x) = J(x)$ for all $x \in \mathbb{X}^{\mathrm{g}}$. Also, assume that the set of admissible inputs $\mathbb{U}(x)$ is compact for each $x \in \mathbb{X}^{\mathrm{g}}$, and denote $\mathbb{U}^{\mathrm{g}}(x) = \mathbb{U}(x) \cap \mathbb{U}^{\mathrm{g}}$. Then, for each $x \in \mathbb{X}^{\mathrm{g}}$, it holds that*

$$-e_1 \le \mathcal{T}^{\mathrm{d}}[J^{\mathrm{d}}](x) - \mathcal{T}[J](x) \le e_1 + e_2(x),$$

where

$$e_1 = \big[\,\mathrm{L}(J) + \mathrm{L}(\widetilde{J^{\mathrm{d}}})\big] \cdot \mathrm{d_H}(\mathbb{X}, \mathbb{X}^{\mathrm{g}}),$$

$$e_2(x) = \big[\,\mathrm{L}(J) + \mathrm{L}(C)\big] \cdot \mathrm{d_H}\left(\mathbb{U}(x), \mathbb{U}^{\mathrm{g}}(x)\right).$$

Proof. Define $Q_x(u) \coloneqq C(x,u) + J\big(f(x,u)\big)$ and $\widetilde{Q}_x(u) \coloneqq C(x,u) + \widetilde{J^{\mathrm{d}}}\big(f(x,u)\big)$. Let us fix $x \in \mathbb{X}^{\mathrm{g}}$. In what follows, we consider the effect of (i) replacing J with $\widetilde{J^{\mathrm{d}}}$, and (ii) minimizing over $\mathbb{U}^{\mathrm{g}}$ instead of $\mathbb{U}(x)$, separately. To this end, we define the *intermediate* DP operator

$$\mathcal{T}^{\mathrm{i}}[J](x) \coloneqq \min_{u}\ \widetilde{Q}_x(u), \quad x \in \mathbb{X}^{\mathrm{g}}.$$

(i) Difference between $\mathcal{T}$ and $\mathcal{T}^{\mathrm{i}}$: Let $u^\star \in \operatorname{argmin}_u Q(x,u) \subseteq \mathbb{U}(x)$, so that $\mathcal{T}[J](x) = Q(x,u^\star)$ and $\mathcal{T}^{\mathrm{i}}[J](x) \le \widetilde{Q}(x,u^\star)$. Also, let $z^\star \in \operatorname{argmin}_{z\in\mathbb{X}^{\mathrm{g}}} \|z - f(x,u^\star)\|$. Then,

$$\begin{aligned}\mathcal{T}^{\mathrm{i}}[J](x) - \mathcal{T}[J](x) &\le \widetilde{Q}(x,u^\star) - Q(x,u^\star) \\ &= \widetilde{J^{\mathrm{d}}}\big(f(x,u^\star)\big) - \widetilde{J^{\mathrm{d}}}(z^\star) + J(z^\star) - J\big(f(x,u^\star)\big),\end{aligned}$$

where we used the assumption that $\widetilde{J^{\mathrm{d}}}(z^\star) = J(z^\star)$ for $z^\star \in \mathbb{X}^{\mathrm{g}}$. Hence,

$$\begin{aligned}\mathcal{T}^{\mathrm{i}}[J](x) - \mathcal{T}[J](x) &\le \big[\,\mathrm{L}(J) + \mathrm{L}(\widetilde{J^{\mathrm{d}}})\big] \cdot \|z^\star - f(x,u^\star)\| \\ &= \big[\,\mathrm{L}(J) + \mathrm{L}(\widetilde{J^{\mathrm{d}}})\big] \cdot \min_{z\in\mathbb{X}^{\mathrm{g}}} \|z - f(x,u^\star)\| \\ &\le \big[\,\mathrm{L}(J) + \mathrm{L}(\widetilde{J^{\mathrm{d}}})\big] \cdot \max_{z'\in\mathbb{X}}\ \min_{z\in\mathbb{X}^{\mathrm{g}}} \left\|z - z'\right\| \\ &= \big[\,\mathrm{L}(J) + \mathrm{L}(\widetilde{J^{\mathrm{d}}})\big] \cdot \mathrm{d_H}(\mathbb{X}, \mathbb{X}^{\mathrm{g}}) = e_1,\end{aligned}$$

where for the second inequality we used the fact that $f(x,u^\star) \in \mathbb{X}$. We can use the same line of arguments by defining $\tilde{u}^\star \in \operatorname{argmin}_u \widetilde{Q}(x,u)$, and $\tilde{z}^\star \in \operatorname{argmin}_{z\in\mathbb{X}^{\mathrm{g}}} \|z - f(x,\tilde{u}^\star)\|$ to show that $\mathcal{T}^{\mathrm{i}}[J](x) - \mathcal{T}[J](x) \le e_1$. Combining these results, we have

$$-e_1 \le \mathcal{T}^{\mathrm{i}}[J](x) - \mathcal{T}[J](x) \le e_1. \tag{39}$$

(ii) Difference between $\mathcal{T}^{\mathrm{i}}$ and $\mathcal{T}^{\mathrm{d}}$: First note that, by construction, we have $\mathcal{T}^{\mathrm{i}}[J](x) \le \mathcal{T}^{\mathrm{d}}[J^{\mathrm{d}}](x)$. Now, let $\tilde{u}^\star \in \operatorname{argmin}_u \widetilde{Q}(x,u) \subseteq \mathbb{U}(x)$, so that $\mathcal{T}^{\mathrm{i}}[J](x) = \widetilde{Q}(x,\tilde{u}^\star)$. Also, let

$$\bar{u}^\star \in \operatorname*{argmin}_{u\in\mathbb{U}^{\mathrm{g}}(x)} \|u - \tilde{u}^\star\|\,,$$

and note that $\mathcal{T}^{\mathrm{d}}[J^{\mathrm{d}}](x) \le \widetilde{Q}(x,\bar{u}^\star)$. Then, using the fact that $\widetilde{Q}$ is Lipschitz continuous, we have

$$\begin{aligned}0 \le \mathcal{T}^{\mathrm{d}}[J^{\mathrm{d}}](x) - \mathcal{T}^{\mathrm{i}}[J](x) &\le \widetilde{Q}(x,\bar{u}^\star) - \widetilde{Q}(x,\tilde{u}^\star) \le \mathrm{L}(\widetilde{Q}_x) \cdot \|\bar{u}^\star - \tilde{u}^\star\| \\ &\le \big[\,\mathrm{L}(J) + \mathrm{L}(C)\big] \cdot \min_{u\in\mathbb{U}^{\mathrm{g}}(x)} \|u - \tilde{u}^\star\| \\ &\le \big[\,\mathrm{L}(J) + \mathrm{L}(C)\big] \cdot \max_{u'\in\mathbb{U}(x)}\ \min_{u\in\mathbb{U}^{\mathrm{g}}(x)} \left\|u - u'\right\| \\ &= \big[\,\mathrm{L}(J) + \mathrm{L}(C)\big] \cdot \mathrm{d_H}\left(\mathbb{U}(x), \mathbb{U}^{\mathrm{g}}(x)\right) = e_2(x),\end{aligned}$$

Combining this last result with the inequality (39), we derive the bounds of the proposition. ☐

Appendix B. Technical Proofs

B.1. Proof of Lemma 2.5

Let $y \in \mathbb{R}^n$, and observe that

$$h^{\mathrm{d}*}(y) = \max_{x \in \mathbb{X}^{\mathrm{d}}} \{\langle y, x\rangle - h(x)\} \leq \max_{x \in \mathbb{R}^n} \{\langle y, x\rangle - h(x)\} = h^*(y).$$

This settles the first inequality in (7) and (8). Now, assume that $\partial h^*(y) \neq \emptyset$, and let $x \in \partial h^*(y)$ so that $h(x) + h^*(y) = \langle y, x\rangle$ [8, Prop. 5.4.3]. Also, let $\tilde{x} \in \operatorname{argmin}_{z \in \mathbb{X}^{\mathrm{d}}} \|x - z\|$, and note that $h^{\mathrm{d}*}(y) \geq \langle y, \tilde{x}\rangle - h(\tilde{x})$. Then,

$$\begin{aligned} h^*(y) - h^{\mathrm{d}*}(y) &\leq \langle y, x - \tilde{x}\rangle - h(x) + h(\tilde{x}) \\ &\leq \left[\, \|y\| + \mathrm{L}\left(h; \{x\} \cup \mathbb{X}^{\mathrm{d}}\right)\right] \cdot \|x - \tilde{x}\| \\ &= \left[\, \|y\| + \mathrm{L}\left(h; \{x\} \cup \mathbb{X}^{\mathrm{d}}\right)\right] \cdot \mathrm{d}(x, \mathbb{X}^{\mathrm{d}}). \end{aligned}$$

Hence, by minimizing over the choice $x \in \partial h^*(y)$, we derive the upper bound provided in (7). In particular, note that if $\partial h^*(y) = \emptyset$, then the upper bound becomes trivial, i.e., $\widetilde{e}_1 = \infty$. Finally, the additional constraints of compactness of $\mathbb{X} = \mathrm{dom}(h)$ implies that $\partial h^*(y) \cap \mathbb{X} \neq \emptyset$. Hence, we can choose $x \in \partial h^*(y) \cap \mathbb{X}$ and use Lipschitz-continuity of h to write

$$\begin{aligned} h^*(y) - h^{\mathrm{d}*}(y) &\leq \left[\, \|y\| + \mathrm{L}\left(h; \{x\} \cup \mathbb{X}^{\mathrm{d}}\right)\right] \cdot \mathrm{d}(x, \mathbb{X}^{\mathrm{d}}) \\ &\leq \left[\, \|y\| + \mathrm{L}(h)\right] \cdot \max_{z \in \mathbb{X}} \mathrm{d}(z, \mathbb{X}^{\mathrm{d}}) = \widetilde{e}_2(y, h, \mathbb{X}^{\mathrm{d}}). \end{aligned}$$

B.2. Proof of Lemma 2.6

Let us first consider the case $y \in \mathrm{co}(\mathbb{Y}^{\mathrm{g}})$. The value of the multi-linear interpolation $\overline{h^{*\mathrm{d}}}(y)$ is a convex combination of $h^{*\mathrm{d}}(y^{(k)}) = h^*(y^{(k)})$ over the grid points $y^{(k)} \in \mathbb{Y}^{\mathrm{g}},\ k \in 1, \ldots, 2^n$, located at the vertices of the hyper-rectangular cell that contains y. That is, $\overline{h^{*\mathrm{d}}}(y) = \sum_k \alpha^{(k)}\ h^*(y^{(k)})$, where $\sum_k \alpha^{(k)} = 1$ and $\alpha^{(k)} \in [0, 1]$. Note that, since we are using LERP, we also have $y = \sum_k \alpha^{(k)}\ y^{(k)}$. Then,

$$h^*(y) = h^*\left(\textstyle\sum_k \alpha^{(k)}\ y^{(k)}\right) \leq \textstyle\sum_k \alpha^{(k)}\ h^*(y^{(k)}) = \overline{h^{*\mathrm{d}}}(y), \tag{40}$$

where the inequality follows from the convexity of h^*. Also, notice that

$$\begin{aligned} \overline{h^{*\mathrm{d}}}(y) &= \textstyle\sum_k \alpha^{(k)}\ h^*(y^{(k)}) = \sum_k \alpha^{(k)}\ \max_{x \in \mathbb{X}} \left\{\langle y^{(k)}, x\rangle - h(x)\right\} \\ &= \textstyle\sum_k \alpha^{(k)}\ \max_{x \in \mathbb{X}} \left\{\langle y, x\rangle - h(x) + \langle y^{(k)} - y, x\rangle\right\} \\ &\leq \textstyle\sum_k \alpha^{(k)}\ \max_{x \in \mathbb{X}} \left\{\langle y, x\rangle - h(x) + \left\|y^{(k)} - y\right\| \cdot \|x\|\right\} \\ &\leq \textstyle\sum_k \alpha^{(k)}\ \max_{x \in \mathbb{X}} \left\{\langle y, x\rangle - h(x) + \Delta_{\mathbb{X}} \cdot \mathrm{d}(y, \mathbb{Y}^{\mathrm{g}})\right\}. \end{aligned}$$

Then, using $\sum_k \alpha^k = 1$, we have

$$\overline{h^{*\mathrm{d}}}(y) \leq \max_{x \in \mathbb{X}} \left\{\langle y, x\rangle - h(x)\right\} + \Delta_{\mathbb{X}} \cdot \mathrm{d}(y, \mathbb{Y}^{\mathrm{g}}) \leq h^*(y) + \Delta_{\mathbb{X}} \cdot \mathrm{d}(y, \mathbb{Y}^{\mathrm{g}}). \tag{41}$$

Combining the two inequalities (40) and (41) gives us the inequality (9) in the lemma.

We next consider the case $y \notin \mathrm{co}(\mathbb{Y}^{\mathrm{g}})$ under the extra assumption $\mathrm{co}(\mathbb{Y}^{\mathrm{g}}_{\mathrm{sub}}) \supset \mathbb{L}(h)$. Note that this assumption implies that (consult the notation preceding the lemma):

- $\mathbb{L}(h)$ is bounded (h is Lipschitz continuous); and,
- $y_i^1 < y_i^2 \leq \mathrm{L}_i^-(h)$ and $\mathrm{L}_i^+(h) \leq y_i^{Y_i-1} < y_i^{Y_i}$ for all $i \in \{1, \dots, n\}$.

In order to simplify the exposition, we consider the two-dimensional case ($n = 2$), while noting that the provided arguments can be generalized to higher dimensions. So, let $\mathbb{Y}^{\mathrm{g}} = \mathbb{Y}_1^{\mathrm{g}} \times \mathbb{Y}_2^{\mathrm{g}}$, where $\mathbb{Y}_i^{\mathrm{g}}$ ($i = 1, 2$) is the finite set of real numbers $y_i^1 < y_i^2 < \ldots < y_i^{Y_i}$ with $Y_i \geq 3$. Let us further simplify the argument by letting $y = (y_1, y_2) \notin \mathrm{co}(\mathbb{Y}^{\mathrm{g}})$ be such that $y_1 < y_1^1$ and $y_2^1 \leq y_2 \leq y_2^2$, so that computing $\overline{h^{*\mathrm{d}}}(y)$ involves extrapolation in the first dimension and interpolation in the second dimension; see Figure 5a for a visualization of this instantiation. Since the extension uses LERP, using the points depicted in Figure 5a, we can write

$$\overline{h^{*\mathrm{d}}}(y) = \alpha \ \overline{h^{*\mathrm{d}}}(y') + (1 - \alpha) \ \overline{h^{*\mathrm{d}}}(y''), \tag{42}$$

where $\alpha = (y_1^2 - y_1)/(y_1^2 - y_1^1)$, and

$$\begin{aligned} \overline{h^{*\mathrm{d}}}(y') &= \beta \ h^*(y^{1,1}) + (1 - \beta) \ h^*(y^{1,2}), \\ \overline{h^{*\mathrm{d}}}(y'') &= \beta \ h^*(y^{1,2}) + (1 - \beta) \ h^*(y^{2,2}), \end{aligned} \tag{43}$$

where $\beta = (y_2^2 - y_2)/(y_2^2 - y_2^1)$. In Figure 5a, we have also paired each of the points of interest in the dual domain with its corresponding maximizer in the primal domain. That is, for $\xi = y, y', y'', y^{1,1}, y^{1,2}, y^{1,2}, y^{2,2}$, we have respectively identified $\eta = x, x', x'', x^{1,1}, x^{1,2}, x^{1,2}, x^{2,2} \in \mathbb{X}$, where $\xi \in \partial h(\eta)$ so that

$$h^*(\xi) = \langle \eta, \xi \rangle - h(\eta). \tag{44}$$

We now list the *implications* of the assumption $y_1^1 < y_1^2 \leq \mathrm{L}_1^-(h)$; Figure 5b illustrates these implications in the one-dimensional case:

I.1. We have $h^*(y) = \alpha \ h^*(y') + (1 - \alpha) \ h^*(y'')$.
I.2. We can choose the maximizers in the primal domain such that
  I.2.1. $x^{1,1} = x^{2,1}$, $x^{1,2} = x^{2,2}$, and $x = x' = x''$;
  I.2.2. $x_1^{1,1} = x_1^{1,2} = x_1 = \min_{(z_1,z_2)\in\mathbb{X}} z_1$.

With these preparatory discussions, we can now consider the error of extrapolative discrete conjugation at the point y. In this regard, first note that $\{y', y''\} \subset \mathrm{co}(\mathbb{Y}^{\mathrm{g}})$, and hence we can use the result of first part of the lemma to write

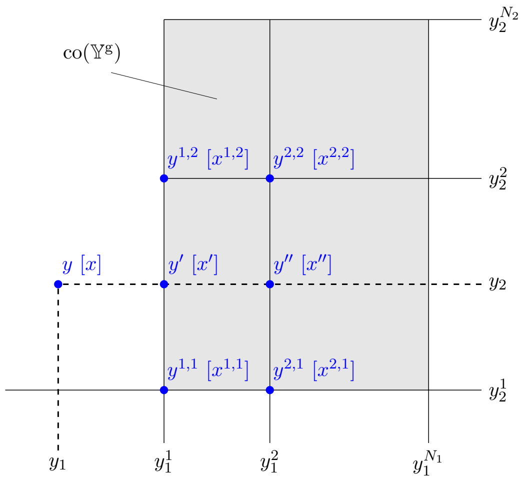
$$\overline{h^{*\mathrm{d}}}(y') = h^*(y') + e', \quad \overline{h^{*\mathrm{d}}}(y'') = h^*(y'') + e'', \tag{45}$$

where $\{e', e''\} \subset [0, \Delta_{\mathbb{X}} \cdot \mathrm{d_H}(\{y', y''\}, \mathbb{Y}^{\mathrm{g}})]$. We claim that these error terms are equal. Indeed, from (43) and (45), we have

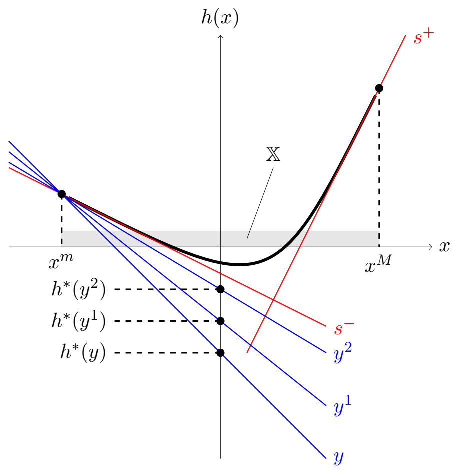
$$e' - e'' = \beta \left[h^*(y^{1,1}) - h^*(y^{2,1})\right] + (1 - \beta) \left[h^*(y^{1,2}) - h^*(y^{2,2})\right] + h^*(y'') - h^*(y').$$

Then, using the pairings in (44) and the implication I.2, we can write

$$\begin{aligned} e' - e'' &\overset{(I.2.1)}{=} \beta \left\langle x^{1,1}, y^{1,1} - y^{2,1} \right\rangle + (1 - \beta) \left\langle x^{1,2}, y^{1,2} - y^{2,2} \right\rangle + \left\langle x, y'' - y' \right\rangle \\ &= \beta \left\langle x^{1,1}, (y_1^1 - y_1^2, 0) \right\rangle + (1 - \beta) \left\langle x^{1,2}, (y_1^1 - y_1^2, 0) \right\rangle + \left\langle x, (y_1^2 - y_1^1, 0) \right\rangle \\ &= \Big(\beta x_1^{1,1} + (1 - \beta) x_1^{1,2} - x_1\Big) (y_1^1 - y_1^2) \overset{(I.2.2)}{=} 0. \end{aligned}$$



(a) Position of the point y w.r.t. the grid $\mathbb{Y}^{\mathrm{g}}$



(b) Implications of the assumption

FIGURE 5. Illustration of the proof of Lemma 2.6. (a) The dual grid $\mathbb{Y}^{\mathrm{g}}$ and the position of the point y w.r.t. the grid. The blue dots show the points of interest and their corresponding maximizer in the primal domain. E.g., “y [x]” implies that $y \in \partial h(x)$, where $x \in \mathbb{X}$, so that $\langle x, y\rangle = h(x) + h^*(y)$. (b) Illustration of the implications of the assumption $y^1 < y^2 \leq s^- = \mathrm{L}^-(h)$ in the one-dimensional case. The colored (red and blue) variables denote the slope of the corresponding lines. Note that $\{y, y^1, y^2\} \subset \partial h(x^m)$, where $x^m = \min_{x\in\mathbb{X}} x$. Indeed, for all $y \leq s^-$, the conjugate $h^*(y) = \langle x^m, y\rangle - h(x^m)$ is a linear function with slope x^m. In particular, for $y < y^1$, we have $h^*(y) = \alpha h^*(y^1) + (1-\alpha)h^*(y^2)$, where $\alpha = (y^2 - y)/(y^2 - y^1)$.

With this result at hand, we can employ the equality (42) and the implication I.1 to write

$$\overline{h^{*\mathrm{d}}}(y) - h^*(y) = \alpha\left[\overline{h^{*\mathrm{d}}}(y') - h^*(y')\right] + (1-\alpha)\left[\overline{h^{*\mathrm{d}}}(y'') - h^*(y'')\right] = \alpha e' + (1-\alpha)e'' = e'.$$

That is,

$$0 \leq \overline{h^{*\mathrm{d}}}(y) - h^*(y) \leq \Delta_{\mathbb{X}} \cdot \mathrm{d_H}(\{y', y''\}, \mathbb{Y}^{\mathrm{g}}) \leq \Delta_{\mathbb{X}} \cdot \mathrm{d_H}\left(\mathrm{co}(\mathbb{Y}^{\mathrm{g}}), \mathbb{Y}^{\mathrm{g}}\right),$$

where for the last inequality we used the fact that $\{y', y''\} \subset \mathrm{co}(\mathbb{Y}^{\mathrm{g}})$.

B.3. **Proof of Corollary 2.7**

The first statement immediately follows from Lemma 2.6 since the finite set $\mathbb{X}^{\mathrm{d}}$ is compact. For the second statement, the extra condition $\mathrm{co}(\mathbb{Y}^{\mathrm{g}}_{\mathrm{sub}}) \supseteq \mathbb{L}(h)$ has the same implications as the ones provided in the proof of Lemma 2.6 in Appendix B.2. Hence, following the same arguments, we can show that provided bounds hold for all $y \in \mathbb{R}^n$ under the aforementioned condition.

B.4. **Proof of Lemma 4.2**

Using the definition of conjugate transform, we have

$$\begin{aligned}\widehat{\mathcal{T}}[J](x) &= \max_{y\in\mathbb{R}^n} \min_{u,z\in\mathbb{R}^n} \{C(x,u) + J(z) + \langle y, f_{\mathrm{s}}(x) + f_{\mathrm{i}}(x)u - z\rangle\} \\ &= \max_{y}\left\{\langle y, f_{\mathrm{s}}(x)\rangle - \max_{u}\left[\left\langle -f_{\mathrm{i}}(x)^\top y, u\right\rangle - C(x,u)\right] - \max_{z}\left[\langle y, z\rangle - J(z)\right]\right\}\end{aligned}$$

$$= \max_{y} \left\{ \langle y, f_{\mathrm{s}}(x) \rangle - C_x^*(-f_{\mathrm{i}}(x)^\top y) - J^*(y) \right\}$$
$$= \max_{y} \left\{ \langle y, f_{\mathrm{s}}(x) \rangle - \phi_x(y) \right\} = \phi_x^*\big(f_{\mathrm{s}}(x)\big).$$

B.5. **Proof of Theorem 4.4**

In what follows, we provide the time complexity of each line of Algorithm 1. The LLT of line 1 requires $\mathcal{O}(X+Y)$ operations; see Remark 2.3. By Assumption 4.3, computing φ_x^{d} in line 3 has a complexity of $\mathcal{O}(Y)$. The minimization via enumeration in line 4 also has a complexity of $\mathcal{O}(Y)$. This, in turn, implies that the `for loop` over $x \in \mathbb{X}^{\mathrm{g}}$ requires $\mathcal{O}(XY)$ operations. Hence, the time complexity of the whole algorithm is of $\mathcal{O}(XY)$.

B.6. **Proof of Proposition 4.5**

We can use the representation (24) and the definition (22) to obtain

$$\begin{aligned}
\widehat{\mathcal{T}}^{\mathrm{d}}[J^{\mathrm{d}}](x) &= \max_{y \in \mathbb{Y}^{\mathrm{g}}} \left\{ \langle f_{\mathrm{s}}(x), y \rangle - \varphi_x^{\mathrm{d}}(y) \right\} \\
&= \max_{y \in \mathbb{Y}^{\mathrm{g}}} \left\{ \langle f_{\mathrm{s}}(x), y \rangle - C_x^*(-f_{\mathrm{i}}(x)^\top y) - J^{\mathrm{d*d}}(y) \right\} \\
&= \max_{y \in \mathbb{Y}^{\mathrm{g}}} \left\{ \langle f_{\mathrm{s}}(x), y \rangle - \max_{u \in \operatorname{dom} C(x,\cdot)} \left[\left\langle -f_{\mathrm{i}}(x)^\top y, u \right\rangle - C(x,u) \right] - J^{\mathrm{d*d}}(y) \right\} \\
&= \max_{y \in \mathbb{Y}^{\mathrm{g}}} \min_{u \in \operatorname{dom} C(x,\cdot)} \left\{ C(x,u) + \langle y, f(x,u) \rangle - J^{\mathrm{d*d}}(y) \right\},
\end{aligned}$$

By the properties laid out in Setting 1, the objective function of this maximin problem is convex in u, with $\operatorname{dom}\big(C(x,\cdot)\big)$ being compact. Also, the objective function is Ky Fan concave in y, which follows from the convexity of $J^{\mathrm{d*}}$. Then, by the Ky Fan's Minimax Theorem (see, e.g., [19, Thm. A]), we can swap the maximization and minimization operators to obtain

$$\begin{aligned}
\widehat{\mathcal{T}}^{\mathrm{d}}[J^{\mathrm{d}}](x) &= \min_{u \in \operatorname{dom} C(x,\cdot)} \max_{y \in \mathbb{Y}^{\mathrm{g}}} \left\{ C(x,u) + \langle y, f(x,u) \rangle - J^{\mathrm{d*}}(y) \right\} \\
&= \min_{u} \left\{ C(x,u) + J^{\mathrm{d*d*}}\big(f(x,u)\big) \right\}.
\end{aligned}$$

B.7. **Proof of Theorem 4.6**

Let us first note that the convexity of $J : \mathbb{X} \to \mathbb{R}$ implies that the duality gap is zero. Indeed, following a similar argument as the one provided in the proof of Proposition 4.5 in Appendix B.6, and using Sion's Minimax Theorem (see, e.g., [33, Thm. 3]), we can show that the CDP operator (23) equivalently reads as

$$\widehat{\mathcal{T}}[J](x) = \min_{u} \left\{ C(x,u) + J^{**}\big(f(x,u)\big) \right\}, \quad x \in \mathbb{X}^{\mathrm{g}}.$$

Then, since J is a proper, closed, convex function, we have $J^{**} = J$, and hence $\widehat{\mathcal{T}}[J] = \mathcal{T}[J]$.

We next consider the disctretization error in $\widehat{\mathcal{T}}^{\mathrm{d}}$ (24) w.r.t. $\widehat{\mathcal{T}}$ (23). First, we can use Lemma 2.5, and the fact that $\operatorname{dom}(J) = \mathbb{X}$ is compact, to write

$$0 \le \phi_x(y) - \varphi_x(y) = J^*(y) - J^{\mathrm{d*}}(y) \le \widetilde{e}_2(y, J, \mathbb{X}^{\mathrm{g}}) \le \max_{y \in \mathbb{Y}^{\mathrm{g}}} \widetilde{e}_2(y, J, \mathbb{X}^{\mathrm{g}}) = e_2, \quad \forall y \in \mathbb{Y}^{\mathrm{g}}.$$

The preceding inequality captures the error due to discretization of the primal domain $\mathbb{X}$, i.e., using $J^{\mathrm{d}*}$ in (24b) instead of J^* in (23b). Using this inequality and the definition of discrete conjugate, we can write

$$0 \le \varphi_x^{\mathrm{d}*}\big(f_{\mathrm{s}}(x)\big) - \phi_x^{\mathrm{d}*}\big(f_{\mathrm{s}}(x)\big) \le e_2,, \quad \forall x \in \mathbb{X}^{\mathrm{g}}. \tag{46}$$

We can also use Lemma 2.5, to write

$$0 \le \phi_x^{*}\big(f_{\mathrm{s}}(x)\big) - \phi_x^{\mathrm{d}*}\big(f_{\mathrm{s}}(x)\big) \le \widetilde{e}_1(f_{\mathrm{s}}(x), \phi_x, \mathbb{Y}^{\mathrm{g}}), \quad \forall x \in \mathbb{X}^{\mathrm{g}}.$$

This captures the error due to discretization of the dual domain $\mathbb{Y} = \mathbb{R}^n$, i.e., approximating ϕ_x^* in (23a) via $\varphi_x^{\mathrm{d}*}$ in (24a). Now, observe that

$$\begin{aligned}\widetilde{e}_1(f_{\mathrm{s}}(x), \phi_x, \mathbb{Y}^{\mathrm{g}}) &= \min_{y \in \partial\phi_x^*(f_{\mathrm{s}}(x))} \Big\{ \big[\, \|f_{\mathrm{s}}(x)\| + \mathrm{L}\left(\phi_x; \{y\} \cup \mathbb{Y}^{\mathrm{g}}\right)\big] \cdot \mathrm{d}(y, \mathbb{Y}^{\mathrm{g}}) \Big\} \\ &\le \min_{y \in \partial\mathcal{T}[J](x)} \Big\{ \big[\, \|f_{\mathrm{s}}(x)\| + \|f_{\mathrm{i}}(x)\| \cdot \Delta_{\mathbb{U}} + \Delta_{\mathbb{X}}\big] \cdot \mathrm{d}(y, \mathbb{Y}^{\mathrm{g}}) \Big\},\end{aligned}$$

where we used the fact that $\phi_x^*\big(f_{\mathrm{s}}(\cdot)\big) = \widehat{\mathcal{T}}[J](\cdot) = \mathcal{T}[J](\cdot)$, and

$$\begin{aligned}\mathrm{L}\left(\phi_x(\cdot)\right) &\le \mathrm{L}\left(C_x^*(-f_{\mathrm{i}}(x)^\top \cdot)\right) + \mathrm{L}\left(J^*(\cdot)\right) \\ &\le \|f_{\mathrm{i}}(x)\| \cdot \mathrm{L}(C_x^*) + \mathrm{L}(J^*) \\ &\le \|f_{\mathrm{i}}(x)\| \cdot \Delta_{\mathrm{dom}(C(x,\cdot))} + \Delta_{\mathrm{dom}(J)} \\ &\le \|f_{\mathrm{i}}(x)\| \cdot \Delta_{\mathbb{U}} + \Delta_{\mathbb{X}}.\end{aligned}$$

Hence, for each $x \in \mathbb{X}^{\mathrm{g}}$ we have

$$\begin{aligned}0 \le \phi_x^{*}\big(f_{\mathrm{s}}(x)\big) - \phi_x^{\mathrm{d}*}\big(f_{\mathrm{s}}(x)\big) &\le \widetilde{e}_1(f_{\mathrm{s}}(x), \phi_x, \mathbb{Y}^{\mathrm{g}}) \\ &\le \big[\, \|f_{\mathrm{s}}(x)\| + \|f_{\mathrm{i}}(x)\| \cdot \Delta_{\mathbb{U}} + \Delta_{\mathbb{X}}\big] \cdot \mathrm{d}\left(\partial\mathcal{T}[J](x), \mathbb{Y}^{\mathrm{g}}\right) = e_1(x).\end{aligned}$$

Combining the last inequality with the inequality (46) completes the proof.

B.8. **Proof of Theorem 5.2**

In what follows, we provide the time complexity of each line of Algorithm 2. The LLT of line 1 requires $\mathcal{O}(X + Y)$ operations; see Remark 2.3. By Assumption 5.1, computing φ^{d} in line 2 has a complexity of $\mathcal{O}(Y)$. The LLT of line 3 requires $\mathcal{O}(Y + Z)$ operations. The approximation of line 5 using LERP has a complexity of $\mathcal{O}(\log Z)$; see Remark 2.2. Hence, the `for loop` over $x \in \mathbb{X}^{\mathrm{g}}$ requires $\mathcal{O}(X \log Z) = \widetilde{\mathcal{O}}(X)$ operations. The time complexity of the whole algorithm can then be computed by adding all the aforementioned complexities.

B.9. **Proof of Theorem 5.3**

Let $\widehat{\mathcal{T}}^{\mathrm{d}}$ denote the output of the implementation of the d-CDP operator (30) via Algorithm 1. Note that the computation of the modified d-CDP operator $\widehat{\mathcal{T}}_{\mathrm{m}}^{\mathrm{d}}$ (31) via Algorithm 2 differs form that of the d-CDP operator $\widehat{\mathcal{T}}^{\mathrm{d}}$ (30) via Algorithm 1 only in the last step. To see this, note that $\widehat{\mathcal{T}}^{\mathrm{d}}$ *exactly* computes $\varphi^{\mathrm{d}*}\big(f_{\mathrm{s}}(x)\big)$ for $x \in \mathbb{X}^{\mathrm{d}}$ (see Algorithm 1:4). However, in $\widehat{\mathcal{T}}_{\mathrm{m}}^{\mathrm{d}}$, the *approximation* $\overline{\varphi^{\mathrm{d}*\mathrm{d}}}\big(f_{\mathrm{s}}(x)\big)$ is used (see Algorithm 2:5), where the approximation uses LERP over the data points $\varphi^{\mathrm{d}*\mathrm{d}} : \mathbb{Z}^{\mathrm{g}} \to \mathbb{R}$. By Corollary 2.7, this leads to an over-approximation of $\varphi^{\mathrm{d}*}$, with the upper bound $e_3 =$

$\Delta_{\mathbb{Y}^{\mathrm{g}}} \cdot \max_{x\in\mathbb{X}^{\mathrm{g}}} \mathrm{d}\left(f_{\mathrm{s}}(x), \mathbb{Z}^{\mathrm{g}}\right) = \Delta_{\mathbb{Y}^{\mathrm{g}}} \cdot \mathrm{d}_{\mathrm{H}}\left(f_{\mathrm{s}}(\mathbb{X}^{\mathrm{g}}), \mathbb{Z}^{\mathrm{g}}\right)$. Hence, compared to $\widehat{\mathcal{T}}^{\mathrm{d}}$, the operator $\widehat{\mathcal{T}}^{\mathrm{d}}_{\mathrm{m}}$ is an over-approximation with the difference bounded by e_3, i.e.,

$$0 \le \widehat{\mathcal{T}}^{\mathrm{d}}_{\mathrm{m}}[J^{\mathrm{d}}](x) - \widehat{\mathcal{T}}^{\mathrm{d}}[J^{\mathrm{d}}](x) \le e_3, \quad \forall x \in \mathbb{X}^{\mathrm{g}}. \tag{47}$$

The result then follows from Theorem 4.6. Indeed, using the definition of $\widehat{\mathcal{T}}^{\mathrm{d}}$ (30), we can define

$$\widehat{\mathcal{I}}^{\mathrm{d}}[J^{\mathrm{d}}](x) := \widehat{\mathcal{T}}^{\mathrm{d}}[J^{\mathrm{d}}](x) - C_{\mathrm{s}}(x) = \varphi^{\mathrm{d}*}\big(f_{\mathrm{s}}(x)\big), \quad x \in \mathbb{X}^{\mathrm{g}},$$
$$\varphi^{\mathrm{d}}(y) := C_{\mathrm{i}}^{*}(-B^{\top}y) + J^{\mathrm{d}*\mathrm{d}}(y), \quad y \in \mathbb{Y}^{\mathrm{g}}.$$

Similarly, using the DP operator (29), we can also define

$$\mathcal{I}[J](x) := \mathcal{T}[J](x) - C_{\mathrm{s}}(x) = \min_{u} \left\{C_{\mathrm{i}}(u) + J\big(f(x,u)\big)\right\}.$$

Then, by Theorem 4.6, for all $x \in \mathbb{X}^{\mathrm{g}}$, it holds that

$$-e_2 \le \mathcal{I}[J](x) - \widehat{\mathcal{I}}^{\mathrm{d}}[J^{\mathrm{d}}](x) = \mathcal{T}[J](x) - \widehat{\mathcal{T}}^{\mathrm{d}}[J^{\mathrm{d}}](x) \le e_1^{m}(x), \tag{48}$$

where e_2 is given in (27), and

$$\begin{aligned} e_1^{m}(x) &= \left[\, \|f_{\mathrm{s}}(x)\| + \|B\| \cdot \Delta_{\mathbb{U}} + \Delta_{\mathbb{X}}\right] \cdot \mathrm{d}\left(\partial \mathcal{I}[J](x), \mathbb{Y}^{\mathrm{g}}\right) \\ &= \left[\, \|f_{\mathrm{s}}(x)\| + \|B\| \cdot \Delta_{\mathbb{U}} + \Delta_{\mathbb{X}}\right] \cdot \mathrm{d}\left(\partial\big(\mathcal{T}[J] - C_{\mathrm{s}}\big)(x), \mathbb{Y}^{\mathrm{g}}\right). \end{aligned}$$

Combining the inequalities (47) and (48) completes the proof.

Appendix C. Extended Algorithms & Further Numerical Examples

C.1. Extended algorithms and their numerical study

In this section, we provide the multistep version of d-CDP algorithms developed in this study that also take into account the extensions discussed in Section 4.3, that is, additive disturbance in the dynamics and numerical computation of the conjugate of the (input-dependent) stage cost. The provided algorithms are

(i) Algorithm 3: multistep implementation of the extended version of Algorithm 1;
(ii) Algorithm 4: multistep implementation of the extended version of Algorithm 2.

We note that all the functions involved in these extended algorithms are now discrete. To simplify the exposition, we are considering disturbances that have a finite support $\mathbb{W}^{\mathrm{d}}$ of size W, with a given p.m.f. $p : \mathbb{W}^{\mathrm{d}} \to [0,1]$. Of course, one can modify the algorithm by incorporating other schemes for computing/approximating the expectation operation. Assuming that the extension operation $\widetilde{[\cdot]}$ in Algorithm 3:7 and Algorithm 4:6 are also handled via LERP, the time complexities are

(i) Algorithm 3: $\widetilde{\mathcal{O}}\big(X(U+V) + TX(W+Y)\big)$ – assuming all the grids $\mathbb{V}^{\mathrm{g}}(x)$ are of size V;
(ii) Algorithm 4: $\widetilde{\mathcal{O}}\big(U+V+T(XW+Y+Z)\big)$.

For the numerical implementation of the extended algorithms, we consider the setup of Section 6.1 for Algorithm 3 and the setup of Section 6.2 for Algorithm 4. However, we now consider stochastic dynamics by introducing an i.i.d., additive disturbance belonging to the finite set $\mathbb{W}^{\mathrm{d}} = \{-0.1, 0, 0.1\}^2$ with a uniform p.m.f., that is, $p(w) = \frac{1}{9}$ for all $w \in \mathbb{W}^{\mathrm{d}}$. Moreover, the conjugate of the stage cost (although analytically available) is computed numerically. In this regard, we note that the dual grids $\mathbb{V}^{\mathrm{g}}(x)$ of the input space are constructed following the guidelines described in Remark 4.8. Through these numerical simulations, we compare the performance of the d-DP and d-CDP algorithms for solving ten instances of the optimal control problem for random initial conditions, chosen uniformly from $\mathbb{X} = [-1,1]^2$. To this end, and similar to the setup of Section 6, we report the average of the relative trajectory cost and the average of the total running time in seconds. The results of our numerical simulations are reported in Table 5.

TABLE 5. Comparison of the performance of the d-DP algorithm and the extended d-CDP Algorithms 3 and 4 for different grid sizes $(X, Y, U, Z, V(x) = N)$: The reported numbers are the average of the relative trajectory cost (w.r.t. the trajectory cost of d-DP (μ) with $N = 41^2$) (left – blue), and the average of the total running time (right – red). See the setup described in Section 6.1 for more details.

Relative trajectory cost / Running time (seconds)			
Alg. \ N	11^2	21^2	41^2
d-CDP Alg. 3	1.45 / 6.4e + 0	0.96 / 7.3e + 1	1.00 / 1.0e + 3
d-DP (J)	1.50 / 2.5e + 1	0.96 / 3.5e + 2	1.00 / 5.1e + 3
d-DP (μ)	1.01 / 2.5e + 1	1.02 / 3.4e + 2	1 / 5.1e + 3
d-CDP Alg. 4	1.47 / 5.5e − 1	0.94 / 1.8e + 0	0.98 / 7.9e + 0
d-DP (J)	1.50 / 2.2e + 1	0.96 / 3.1e + 2	0.99 / 4.9e + 3
d-DP (μ)	1.01 / 2.2e + 1	1.02 / 3.1e + 2	1 / 4.9e + 3

Algorithm 3 Multistep implementation of the extended d-CDP Algorithm 1

Input: dynamics $f_{\mathrm{s}} : \mathbb{R}^n \to \mathbb{R}^n$, $f_{\mathrm{i}} : \mathbb{R}^n \to \mathbb{R}^{n\times m}$;
discrete stage cost $C^{\mathrm{d}}(x,\cdot) : \mathbb{U}^{\mathrm{g}} \to \overline{\mathbb{R}}$ for $x \in \mathbb{X}^{\mathrm{g}}$;
discrete terminal cost $C^{\mathrm{d}}_T : \mathbb{X}^{\mathrm{g}} \to \mathbb{R}$;
discrete disturbance $\mathbb{W}^{\mathrm{d}}$ and its p.m.f. $p : \mathbb{W}^{\mathrm{d}} \to [0,1]$.
Output: discrete costs-to-go $J^{\mathrm{d}}_t : \mathbb{X}^{\mathrm{g}} \to \mathbb{R}$, $t = 0,1,\dots,T$.

for each $x \in \mathbb{X}^{\mathrm{d}}$ **do**
  construct the grid $\mathbb{V}^{\mathrm{g}}(x)$;
  use LLT to compute $C^{\mathrm{d}*\mathrm{d}}_x : \mathbb{V}^{\mathrm{g}}(x) \to \mathbb{R}$ from $C^{\mathrm{d}}(x,\cdot) : \mathbb{U}^{\mathrm{g}} \to \overline{\mathbb{R}}$;
end for
$J^{\mathrm{d}}_T(x) \leftarrow C^{\mathrm{d}}_T(x)$ for $x \in \mathbb{X}^{\mathrm{g}}$;
for $t = T, \dots, 1$ **do**
  $J^{\mathrm{d}}_{\mathrm{w},t}(x) \leftarrow \sum_{w\in\mathbb{W}^{\mathrm{d}}} p(w) \cdot \widetilde{J^{\mathrm{d}}}(x+w)$ for $x \in \mathbb{X}^{\mathrm{g}}$;
  construct the grid $\mathbb{Y}^{\mathrm{g}}$;
  use LLT to compute $J^{\mathrm{d}*\mathrm{d}}_{\mathrm{w},t} : \mathbb{Y}^{\mathrm{g}} \to \mathbb{R}$ from $J^{\mathrm{d}}_{\mathrm{w},t} : \mathbb{X}^{\mathrm{g}} \to \mathbb{R}$;
  for each $x \in \mathbb{X}^{\mathrm{g}}$ **do**
    for each $y \in \mathbb{Y}^{\mathrm{g}}$ **do**
      use LERP to compute $\overline{C^{\mathrm{d}*\mathrm{d}}_x}(-f_{\mathrm{i}}(x)^\top y)$ from $C^{\mathrm{d}*\mathrm{d}}_x : \mathbb{V}^{\mathrm{g}}(x) \to \mathbb{R}$;
      $\varphi^{\mathrm{d}}_x(y) \leftarrow \overline{C^{\mathrm{d}*\mathrm{d}}_x}(-f_{\mathrm{i}}(x)^\top y) + J^{\mathrm{d}*\mathrm{d}}_{\mathrm{w},t}(y)$;
    end for
    $J^{\mathrm{d}}_{t-1}(x) \leftarrow \max_{y\in\mathbb{Y}^{\mathrm{g}}} \{\langle f_{\mathrm{s}}(x), y\rangle - \varphi^{\mathrm{d}}_x(y)\}$.
  end for
end for

Algorithm 4 Multistep implementation of the extended d-CDP Algorithm 2

Input: dynamics $f_{\mathrm{s}} : \mathbb{R}^n \to \mathbb{R}^n,\ B \in \mathbb{R}^{n \times m}$;
discrete state-dependent stage cost $C_{\mathrm{s}}^{\mathrm{d}} : \mathbb{X}^{\mathrm{g}} \to \mathbb{R}$;
discrete input-dependent stage cost $C_{\mathrm{i}}^{\mathrm{d}} : \mathbb{U}^{\mathrm{g}} \to \mathbb{R}$;
discrete terminal cost $C_T^{\mathrm{d}} : \mathbb{X}^{\mathrm{g}} \to \mathbb{R}$;
discrete disturbance $\mathbb{W}^{\mathrm{d}}$ and its p.m.f. $p : \mathbb{W}^{\mathrm{d}} \to [0,1]$.
Output: discrete costs-to-go $J_t^{\mathrm{d}} : \mathbb{X}^{\mathrm{g}} \to \mathbb{R},\ t = 0, 1, \dots, T$.
construct the grid $\mathbb{V}^{\mathrm{g}}$;
use LLT to compute $C_{\mathrm{i}}^{\mathrm{d*d}} : \mathbb{V}^{\mathrm{g}} \to \mathbb{R}$ from $C_{\mathrm{i}}^{\mathrm{d}} : \mathbb{U}^{\mathrm{g}} \to \mathbb{R}$;
construct the grid $\mathbb{Z}^{\mathrm{g}}$;
$J_T^{\mathrm{d}}(x) \leftarrow C_T^{\mathrm{d}}(x)$ for $x \in \mathbb{X}^{\mathrm{g}}$;
for $t = T, \dots, 1$ **do**
  $J_{\mathrm{w},t}^{\mathrm{d}}(x) \leftarrow \sum_{w \in \mathbb{W}^{\mathrm{d}}} p(w) \cdot \widetilde{J^{\mathrm{d}}}(x+w)$ for $x \in \mathbb{X}^{\mathrm{g}}$;
  construct the grid $\mathbb{Y}^{\mathrm{g}}$;
  use LLT to compute $J_{\mathrm{w},t}^{\mathrm{d*d}} : \mathbb{Y}^{\mathrm{g}} \to \mathbb{R}$ from $J_{\mathrm{w},t}^{\mathrm{d}} : \mathbb{X}^{\mathrm{g}} \to \mathbb{R}$;
  for each $y \in \mathbb{Y}^{\mathrm{g}}$ **do**
    use LERP to compute $\overline{C_{\mathrm{i}}^{\mathrm{d*d}}}(-B^\top y)$ from $C_{\mathrm{i}}^{\mathrm{d*d}} : \mathbb{V}^{\mathrm{g}} \to \mathbb{R}$;
    $\varphi^{\mathrm{d}}(y) \leftarrow \overline{C_{\mathrm{i}}^{\mathrm{d*d}}}(-B^\top y) + J_{\mathrm{w},t}^{\mathrm{d*d}}(y)$;
  end for
  use LLT to compute $\varphi^{\mathrm{d*d}} : \mathbb{Z}^{\mathrm{g}} \to \mathbb{R}$ from $\varphi^{\mathrm{d}} : \mathbb{Y}^{\mathrm{g}} \to \mathbb{R}$;
  for each $x \in \mathbb{X}^{\mathrm{d}}$ **do**
    use LERP to compute $\overline{\varphi^{\mathrm{d*d}}}\big(f_{\mathrm{s}}(x)\big)$ from $\varphi^{\mathrm{d*d}} : \mathbb{Z}^{\mathrm{g}} \to \mathbb{R}$;
    $J_{t-1}^{\mathrm{d}}(x) \leftarrow C_{\mathrm{s}}^{\mathrm{d}}(x) + \overline{\varphi^{\mathrm{d*d}}}\big(f_{\mathrm{s}}(x)\big)$;
  end for
end for

C.2. **Echt examples**

In this section, we showcase the application of the proposed d-CDP algorithms in solving the optimal control problem for two typical systems. In particular, we use the extended versions of these algorithms for the optimal control of an SIR (Susceptible–Infected–Recovered) model for epidemics and a noisy inverted pendulum. Again, to show the effectiveness of the proposed algorithms, we compare their performance with the benchmark d-DP algorithm. Moreover, through these examples, we highlight some issues that can arise in the real world application of the proposed algorithms.

C.2.1. *SIR model.* We consider the application of the extended version of the d-CDP Algorithm 1 for computing the optimal vaccination plan in a simple epidemic model. To this end, we consider an SIR system described by [15, Sec. 4]

$$\begin{cases} s_{t+1} = s_t(1-u_t) - \alpha i_t s_t (1-u_t) \\ i_{t+1} = i_t + \alpha i_t s_t (1-u_t) - \beta i_t \\ r_{t+1} = r_t + u_t s_t, \end{cases}$$

where $s_t, i_t, r_t \geq 0$ are, respectively, the normalized number of susceptible, infected, and immune individuals in the population, and $u_t \in [0, u_{\max}]$ is the control input which can be interpreted as the proportion of the susceptibles to be vaccinated ($u_{\max} \leq 1$). We are interested in computing the optimal vaccination policy with linear cost $\sum_{t=0}^{T-1}(\gamma i_t + u_t) + \gamma i_T$, over $T = 3$ steps ($\gamma > 0$). The model parameters are the transmission rate $\alpha = 2$, the death rate $\beta = 0.1$, the maximum vaccination capacity $u_{\max} = 0.8$, and the cost coefficient $\gamma = 100$ (corresponding to the values in [15, Sec. 4.2]).

We now provide the formulation of this problem w.r.t. Setting 1. In this regard, note that the variable r_t (number of immune individuals) can be safely ignored as it affects neither the evolution of the other two variables, nor the cost to be minimized. Hence, we can take $x_t = (s_t, i_t) \in \mathbb{R}^2$ and $u_t \in \mathbb{R}$ as the state and input variables, respectively. The dynamics of the system is then described by $x_{t+1} = f_{\mathrm{s}}(x_t) + f_{\mathrm{i}}(x_t) \cdot u_t$, where

$$f_{\mathrm{s}}(s,i) = \begin{bmatrix} s - \alpha s i \\ (1-\beta) i + \alpha s i \end{bmatrix}, \quad f_{\mathrm{i}}(s,i) = \begin{bmatrix} -s + \alpha s i \\ -\alpha s i \end{bmatrix}.$$

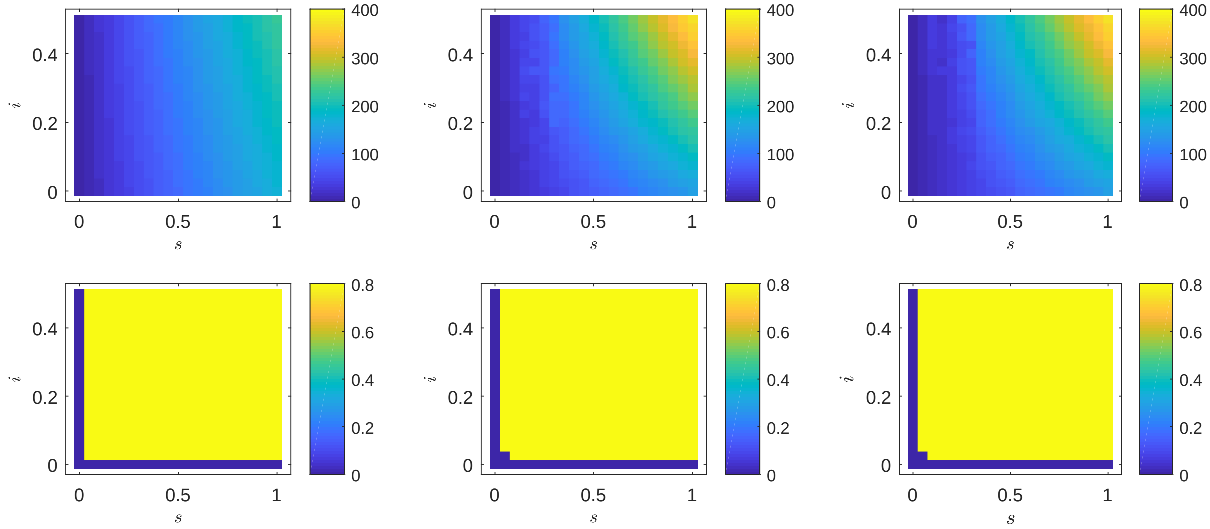
We consider the state constraint $x_t \in \mathbb{X} = [0,1] \times [0,0.5]$, and the input constraint $u_t \in \mathbb{U} = [0,0.8]$. In particular, the constraint $i_t \in [0,0.5]$ is chosen so that the feasibility condition of Setting 1-(ii) is satisfied. Also, the corresponding stage and terminal costs read as $C(s,i,u) = \gamma i + u$, and $C_T(s,i) = i$, respectively. We note that, although the conjugate of the stage cost (C_x^*) is analytically available, we use the scheme provided in Section 4.3.2 to compute C_x^* numerically; ; see also Algorithm 3 in Appendix C.1. In order to deploy the d-DP algorithm and the extended d-CDP algorithm, we use uniform grid-like discretizations of the state and input spaces and the their dual spaces ($\mathbb{X}^{\mathrm{g}}$, $\mathbb{U}^{\mathrm{g}}$, $\mathbb{Y}^{\mathrm{g}}$, and $\mathbb{V}^{\mathrm{g}}(x)$ for $x \in \mathbb{X}^{\mathrm{g}}$). The dual grids $\mathbb{Y}^{\mathrm{g}}$ and $\mathbb{V}^{\mathrm{g}}(x)$ are constructed following the guidelines provided in Remarks 4.7 (with $\alpha = 0.5$) and 4.8, respectively.

Figure 6 depicts the computed cost $J_0^{\mathrm{d}} : \mathbb{X}^{\mathrm{g}} \to \mathbb{R}$ and control law $\mu_0^{\mathrm{d}} : \mathbb{X}^{\mathrm{g}} \to \mathbb{U}^{\mathrm{g}}$, using the d-DP and d-CDP algorithms. In particular, for the d-CDP algorithm, we are reporting the simulation results for two configurations of the dual grids. Table 6 reports the corresponding grid sizes and the running times for solving the value iteration Problem 3.1. In particular, notice how d-DP algorithm significantly outperforms the d-CDP algorithm with the discretization scheme of configuration 1, where $X = Y$ and $U = V$. In this regard, recall that the time complexity of d-DP algorithm is of $\mathcal{O}(TXU)$, while that of d-CDP algorithm is of $\mathcal{O}\left(X(U+V) + TXY\right) = \mathcal{O}\left(XU + TX^2\right)$, when $X = Y$ and $U = V$. Hence, what we observe is indeed expected since the number of input channels is less than the dimension of the state space. For such problems, we should be cautious when using the d-CDP algorithm, particularly, in choosing the sizes Y and V of the dual grids. For instance, for the problem at hand, as reported in Table 6, we can reduce the size of the dual grids as in configuration 2 and hence reduce the running time of the d-CDP algorithm. However, as shown in Figure 6, this reduction in the size of the dual grids does not affect the quality of the computed costs and hence the corresponding control laws.

Table 6. Optimal control of SIR model: Grid sizes and running times.

Alg.	Grid size	Running time
d-DP	$X = 21^2,\ U = 21$	2.00 sec
d-CDP (config. 1)*	$Y = 21^2,\ V = 21$	18.26 sec
d-CDP (config. 2)*	$Y = 11^2,\ V = 11$	6.19 sec

*X and U are the same as in d-DP.



(a) d-DP (b) d-CDP (config. 1) (c) d-CDP (config. 2)

FIGURE 6. Optimal control of SIR model: Cost J_0^{d} (top) and control law μ_0^{d} (bottom).

C.2.2. *Inverted pendulum.* We now consider an application of the extension of the d-CDP Algorithm 2, which handles additive disturbance in the dynamics; see Algorithm 4 in Appendix C.1. To this end, we consider the optimal control of a noisy inverted pendulum with quadratic stage and terminal costs, over a finite horizon. The deterministic continuous-time dynamics of the system is described by [11, Sec. 4.5.3]

$$\ddot{\theta} = \alpha \sin\theta + \beta\dot{\theta} + \gamma u,$$

where θ is the angle (with $\theta = 0$ corresponding to upward position), and u is the control input. The values of the parameters are $\alpha = 118.6445$, $\beta = -1.599$, and $\gamma = 29.5398$ (corresponding to the values of the physical parameters in [11, Sec. 4.5.3]). Here, we consider the corresponding discrete-time dynamics, by using forward Euler method with sampling time $\tau = 0.05$. We also introduce stochasticity by considering an additive disturbance in the dynamics. The discrete-time dynamics then reads as $x_{t+1} = f_{\mathrm{s}}(x_t) + Bu_t + w_t$, where $x_t = (\theta_t, \dot{\theta}_t) \in \mathbb{R}^2$ is the state variable (angle and angular velocity), $w_t \in \mathbb{R}^2$ is the disturbance, and

$$f_{\mathrm{s}}(\theta, \dot{\theta}) = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} + \tau \cdot \begin{bmatrix} \dot{\theta} \\ \alpha \sin\theta + \beta\dot{\theta} \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ \gamma \end{bmatrix}.$$

We consider the state constraints $x_t \in \mathbb{X} = [-\frac{\pi}{4}, \frac{\pi}{4}] \times [\pi, \pi]$, and the input constraints $u_t \in \mathbb{U} = [-3, 3]$. Moreover, we assume that the disturbances w_t are i.i.d., with a uniform distribution over the compact support $\mathbb{W} = \frac{\pi}{4} \cdot [-0.05, 0.05] \times \pi \cdot [-0.05, 0.05]$. The problem of interest is then to compute the costs-to-go $J_t^{\mathrm{d}} : \mathbb{X}^{\mathrm{g}} \to \overline{\mathbb{R}}$ for $t = T-1, \ldots, 0$, over the horizon $T = 50$, with quadratic costs $C_\nu(\cdot) = \|\cdot\|^2, \ \nu \in \{\mathrm{s}, \mathrm{i}, T\}$. We recall that $\mathbb{X}^{\mathrm{g}}$ is a grid-like discretization of the state space $\mathbb{X}$. Also, we note that the conjugate of the input-dependent stage cost C_{i}^* is analytically available, and given by $C_{\mathrm{i}}^*(v) = \hat{u}v - \hat{u}^2, \ v \in \mathbb{R}$, where $\hat{u} = \max\left\{-3, \ \min\left\{\frac{v}{2}, \ 3\right\}\right\}$.

The extension of the d-CDP algorithm for handling additive disturbance involves applying the d-CDP operation to $J^{\mathrm{d}}_{\mathrm{w}}(\cdot) := \mathbb{E}_w \widetilde{J^{\mathrm{d}}}(\cdot + w)$, where $\mathbb{E}$ is the expectation operator, and $\widetilde{[\cdot]}$ is an extension operator. For the extension operation, we use LERP. For the expectation operation, we consider the approximation scheme described in Section 4.3.1, involving discretization of the disturbance set. Precisely, we assume that $w_t \in \mathbb{W}^{\mathrm{d}} = \mathbb{W}^{\mathrm{d}}_1 \times \mathbb{W}^{\mathrm{d}}_2 \subset \mathbb{W}$ with a uniform p.m.f., where

$$\mathbb{W}^{\mathrm{d}}_1 = \frac{\pi}{4} \cdot \{-0.05, -0.025, 0, 0.025, 0.05\},$$
$$\mathbb{W}^{\mathrm{d}}_2 = \pi \cdot \{-0.05, -0.025, 0, 0.025, 0.05\}.$$

Under such assumption, we have

$$J^{\mathrm{d}}_{\mathrm{w}}(x) = \frac{1}{W} \sum_{w \in \mathbb{W}^{\mathrm{d}}} \overline{J^{\mathrm{d}}}(x + w), \qquad x \in \mathbb{X}^{\mathrm{g}}.$$

In order to deploy the stochastic versions of the d-DP and d-CDP algorithms for the optimal control problem described above, we use uniform discretizations of the state, input, and state dual spaces, with $N_i = 21$ discrete points in each dimension, i.e., $X = Y = 21^2$ and $U = 21$. The grid $\mathbb{Z}^{\mathrm{g}}$ is also constructed with the same size ($Z = 21^2$). For the construction of the grids $\mathbb{Y}^{\mathrm{g}}$ and $\mathbb{Z}^{\mathrm{g}}$, we follow the guidelines provided in Remarks 4.7 (with $\alpha = 1$) and 5.4, respectively.

Figure 7 shows the computed cost-to-go at $t = 0$, using the d-DP and d-CDP algorithms. We note that the optimal control problem at hand does not satisfy the feasibility condition assumed in this study. That is, there exist $x \in \mathbb{X}^{\mathrm{g}}$ for which there is no $u \in \mathbb{U}^{\mathrm{g}}$ such that $x^+ = f_{\mathrm{s}}(x) + Bu \in \mathbb{X}$. This explains the black areas in the left panel of Figure 7 with $J_0 = \infty$, computed using d-DP algorithm. Notice, however, that in the right panel of Figure 7, the d-CDP algorithm assigns finite values for these states. This does not contradict our error analysis, as the assumption on the optimal control problem to be feasible for all $x \in \mathbb{X}$ is violated. Indeed, for feasible initial states in the state space, our theoretical results still hold true. We also note that the running times of the two algorithms for solving the value iteration Problem 3.1 were 269.8 *seconds for the d-DP algorithm*, and 10.8 *seconds for the d-CDP algorithm*. As a further illustration, Figure 8 depicts a sample state trajectory of the system, where the control input sequence is derived via minimization of the costs-to-go computed using the d-DP and d-CDP algorithms.

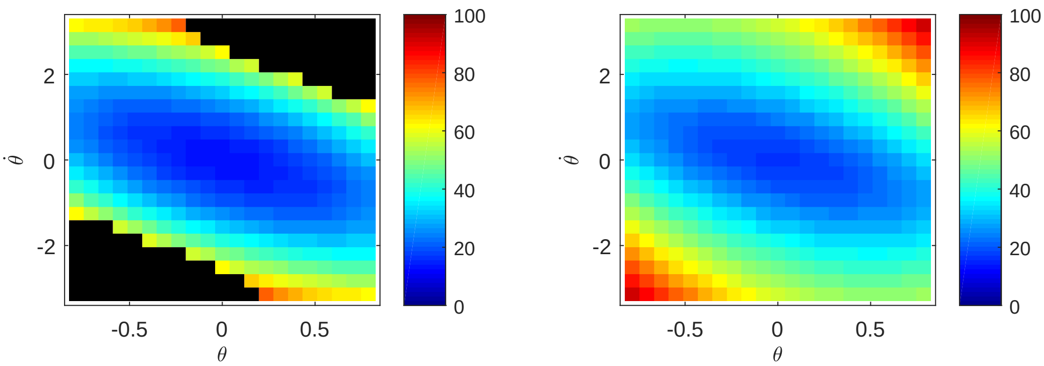


FIGURE 7. Optimal control of noisy inverted pendulum: cost-to-go at $t = 0$ using d-DP (left) and d-CDP (right). The black areas correspond to $J_0 = \infty$.

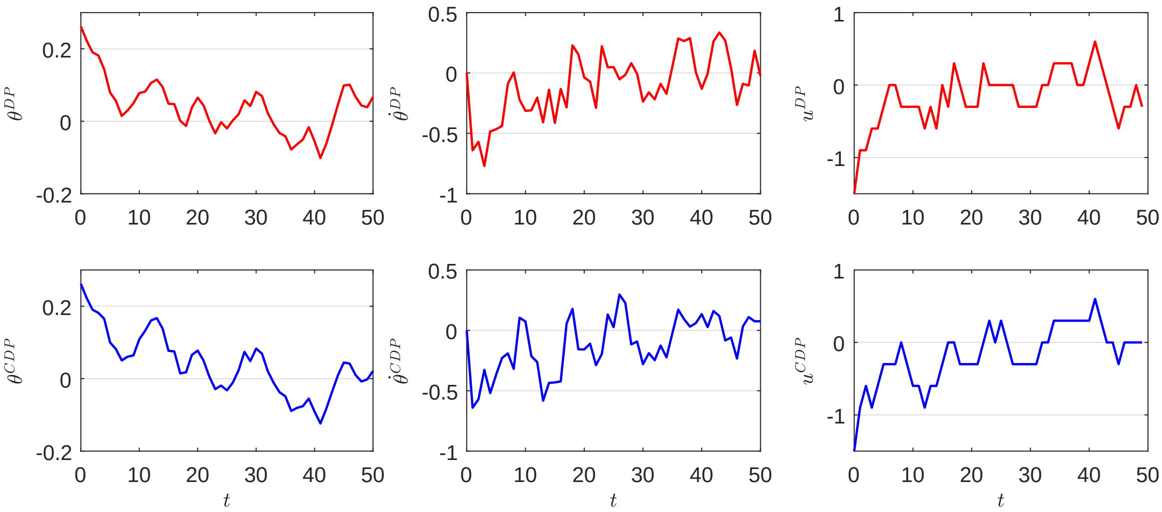


FIGURE 8. Optimal control of noisy inverted pendulum: the state trajectory and input sequence for $x_0 = (\frac{\pi}{12}, 0)^\top$ using d-DP (top) and d-CDP (bottom).

Appendix D. The d-CDP MATLAB package

The MATLAB package [21] concerns the implementation of the two d-CDP algorithms (and their extensions) developed in this study. The provided codes include detailed instructions/comments on how to use them. Also provided is the the numerical examples of Section 6 and Appendix C.1. In what follows we highlight the most important aspects of the developed package with a list of available routines.

Recall that, in this study, we exclusively considered *grid-like* discretizations of both primal and dual domains. This allows us to use the MATALB function `griddedInterpolant` for all the extension operations. We also note that the interpolation and extrapolation methods of this fucntion are all set to `linear`, hence leading to multilinear interpolation & extrapolation (LERP). However, this need not be the case in general, and the user can choose other options available in the `griddedInterpolant` routine, by modifying the corresponding parts of the provided codes; see the comments in the codes for more details. We also note that for the discrete conjugation (LLT), we used the MATLAB package (the `LLTd` routine and two other subroutines, specifically) provided in [23] to develop an n-dimensional LLT routine via factorization (the function `LLT` in the package). Table 7 lists other routines that are available in the developed package. In particular, there are four high level functions (functions (1-4) in Table 7) that are developed separately for the two settings considered in this article. We also note that the provided implementations do not require the discretization of the state and input spaces to satisfy the state and input constraints (particularly, the feasibility condition of Setting 1-(ii)). Nevertheless, the function `feasibility_check_`$*$ ($* = 1, 2$) is developed to provide the user with a warning if that is the case. Finally, we note that the conjugate of four extended real-valued convex functions are also provided in the package; see Table 8.

Table 7. List of routines available in the d-CDP MATLAB package.

MATLAB Function	Description
(1) `d_CDP_Alg_*`	Backward value iteration for finding costs using d-CDP
(2) `d_DP_Alg_*`	Backward value iteration for finding costs and control laws using d-DP
(3) `forward_iter_J_*`	Forward iteration for finding the control sequence for a given initial condition using costs (derived via d-DP or d-CDP)
(4) `forward_iter_Pi_*`	Forward iteration for finding the control sequence for a given initial condition using control laws (derived via d-DP)
(5) `feasibility_check_*`	For checking if the discrete state-input space satisfies the constraints
(6) `eval_func`	For discretization of an analytically available function over a given grid
(7) `eval_func_constr`	An extension of `eval_func` that also checks given constraints
(8) `ext_constr`	For extension of a discrete function while checking a given set of constraints
(9) `ext_constr_expect`	For computing expectation of a discrete function subjected to additive noise
(10) `slope_range`	For computing the range of slopes of a convex-extensible discrete function with a grid-like domain

$* = 1, 2$, corresponding to Settings 1 and 2, respectively.

Table 8. List of analytically available conjugate functions in the d-CDP MATLAB package.

Function	Effective Domain	MATLAB Func. for Conj.
$g : \mathbb{R}^n \to \overline{\mathbb{R}} : u \mapsto u^\top Ru \ (R \succ 0)$	Ball centered at the origin	`conj_Quad_ball`
$g : \mathbb{R}^n \to \overline{\mathbb{R}} : u \mapsto u^\top Ru \ (R \succ 0)$	Box containing the origin	`conj_Quad_box`
$g : \mathbb{R}^n \to \overline{\mathbb{R}} : u \mapsto \sum_{i=1}^n \lvert u_i \rvert$	Box containing the origin	`conj_L1_box`
$g : \mathbb{R}^n \to \overline{\mathbb{R}} : u \mapsto \sum_{i=1}^n e^{\lvert u_i \rvert} - n$	Box containing the origin	`conj_ExpL1_box`

References

[1] Achdou, Y., Camilli, F., and Corrias, L. (2014). On numerical approximation of the Hamilton-Jacobi-transport system arising in high frequency approximations. *Discrete & Continuous Dynamical Systems-Series B*, 19(3).

[2] Akian, M., Gaubert, S., and Lakhoua, A. (2008). The max-plus finite element method for solving deterministic optimal control problems: Basic properties and convergence analysis. *SIAM Journal on Control and Optimization*, 47(2):817–848.

[3] Bach, F. (2019). Max-plus matching pursuit for deterministic Markov decision processes. *arXiv preprint arXiv:1906.08524*.

[4] Balaji, N., Kiefer, S., Novotnỳ, P., Pérez, G. A., and Shirmohammadi, M. (2018). On the complexity of value iteration. *preprint arXiv:1807.04920*.

[5] Bellman, R. and Karush, W. (1962). Mathematical programming and the maximum transform. *Journal of the Society for Industrial and Applied Mathematics*, 10(3):550–567.

[6] Berthier, E. and Bach, F. (2020). Max-plus linear approximations for deterministic continuous-state markov decision processes. *IEEE Control Systems Letters*, pages 1–1.

[7] Bertsekas, D. P. (2005). *Dynamic Programming and Optimal Control, Vol. I.* Athena Scientific, Belmont, MA, 3rd edition.

[8] Bertsekas, D. P. (2009). *Convex Optimization Theory.* Athena Scientific, Belmont, MA.

[9] Bertsekas, D. P. (2019). *Reinforcement Learning and Optimal Control.* Athena Scientific, Belmont, MA.

[10] Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization.* Cambridge University Press.

[11] Busoniu, L., Babuska, R., De Schutter, B., and Ernst, D. (2017). *Reinforcement learning and dynamic programming using function approximators.* CRC press.

[12] Carpio, R. and Kamihigashi, T. (2020). Fast value iteration: an application of Legendre-Fenchel duality to a class of deterministic dynamic programming problems in discrete time. *Journal of Difference Equations and Applications*, 26(2):209–222.

[13] Corrias, L. (1996). Fast Legendre-Fenchel transform and applications to Hamilton-Jacobi equations and conservation laws. *SIAM Journal on Numerical Analysis*, 33(4):1534–1558.

[14] Costeseque, G. and Lebacque, J.-P. (2014). A variational formulation for higher order macroscopic traffic flow models: Numerical investigation. *Transportation Research Part B: Methodological*, 70:112 – 133.

[15] Ding, W. and Lenhart, S. (2010). Introduction to optimal control for discrete time models with an application to disease modeling. In Gumel, A. B. and Lenhart, S., editors, *Modeling Paradigms and Analysis of Disease Transmission Models*, pages 109–120. American Mathematical Society.

[16] Esogbue, A. O. and Ahn, C. W. (1990). Computational experiments with a class of dynamic programming algorithms of higher dimensions. *Computers & Mathematics with Applications*, 19(11):3 – 23.

[17] Felzenszwalb, P. F. and Huttenlocher, D. P. (2012). Distance transforms of sampled functions. *Theory of computing*, 8(1):415–428.

[18] Jacobs, M. and Léger, F. (2019). A fast approach to optimal transport: the back-and-forth method. *arXiv preprint arXiv:1905.12154.*

[19] Joó, I. and Stachó, L. L. (1982). A note on Ky Fan's minimax theorem. *Acta Mathematica Academiae Scientiarum Hungarica*, 39(4):401–407.

[20] Klein, C. M. and Morin, T. L. (1991). Conjugate duality and the curse of dimensionality. *European Journal of Operational Research*, 50(2):220 – 228.

[21] Kolarijani, M. A. S. and Mohajerin Esfahani, P. (2020). Discrete conjugate dynamic programming (d-CDP) MATLAB package. Available online at https://github.com/AminKolarijani/d-CDP.

[22] Lucet, Y. (1996). A fast computational algorithm for the Legendre-Fenchel transform. *Computational Optimization and Applications*, 6(1):27–57.

[23] Lucet, Y. (1997). Faster than the fast Legendre transform, the linear-time Legendre transform. *Numerical Algorithms*, 16(2):171–185.

[24] Lucet, Y. (2009). New sequential exact Euclidean distance transform algorithms based on convex analysis. *Image and Vision Computing*, 27(1):37 – 44.

[25] Lucet, Y. (2010). What shape is your conjugate? A survey of computational convex analysis and its applications. *SIAM Review*, 52(3):505–542.

[26] McEneaney, W. M. (2003). Max-plus eigenvector representations for solution of nonlinear H_∞ problems: basic concepts. *IEEE Transactions on Automatic Control*, 48(7):1150–1163.

[27] McEneaney, W. M. (2006). *Max-plus methods for nonlinear control and estimation.* Springer Science & Business Media.

[28] Murota, K. (2003). *Discrete Convex Analysis.* Society for Industrial and Applied Mathematics.

[29] Powell, W. B. (2011). *Approximate Dynamic Programming: Solving the Curses of Dimensionality.* John Wiley & Sons, Hoboken, NJ, 2nd edition.

[30] Rockafellar, R. (1974). *Conjugate Duality and Optimization.* Philadelphia: Society for Industrial and Applied Mathematics.

[31] Ronagh, P. (2019). Quantum algorithms for solving dynamic programming problems. *preprint arXiv:1906.02229.*

[32] Sidford, A., Wang, M., Wu, X., and Ye, Y. (2018). Variance reduced value iteration and faster algorithms for solving Markov decision processes. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 770–787. SIAM.

[33] Simons, S. (1995). Minimax theorems and their proofs. In Du, D.-Z. and Pardalos, P. M., editors, *Minimax and Applications*, pages 1–23. Springer US, Boston, MA.

[34] Sutter, D., Nannicini, G., Sutter, T., and Woerner, S. (2020). Quantum Legendre-Fenchel transform. *preprint arXiv:2006.04823*.