Generalized Dual Dynamic Programming for Infinite Horizon Problems in Continuous State and Action Spaces

Joseph Warrington, Paul N. Beuchat, and John Lygeros

Abstract

We describe a nonlinear generalization of dual dynamic programming theory and its application to value function estimation for deterministic control problems over continuous state and action (or input) spaces, in a discrete-time infinite horizon setting. We prove that the result of a one-stage policy evaluation can be used to produce nonlinear lower bounds on the optimal value function that are valid over the entire state space. These bounds reflect the functional form of the system's costs, dynamics, and constraints. We provide an iterative algorithm that produces successively better approximations of the optimal value function, prove some key properties of the algorithm, and describe means of certifying the quality of the output. We demonstrate the efficacy of the approach on systems whose dimensions are too large for conventional dynamic programming approaches to be practical.

I. Introduction

Dynamic Programming (DP) facilitates the computation of optimal control actions for dynamical system regulation, or other multi-stage decision problems, by representing future costs via a *value function*, or cost-to-go. This article is concerned with the solution of deterministic infinite-horizon control problems in discrete time, with continuous state and action (control input) spaces. These are a common variant of the broader class of Markov Decision Problems (MDPs) [11].

The DP algorithm was originally developed for problems with discrete state and action spaces [3]. With continuous state and action spaces it is typical to discretize, or "grid", these spaces, and revert to the same algorithmic approach as for discrete problems. This brings about two computational issues. Firstly, for a given grid resolution, the number of discrete points created is exponential in the dimensions of the state and action spaces. Chow and Tsitsiklis proved [11, equation 7.13] that a successive gridding approach leads to a worst-case complexity (under certain assumptions) of $\mathcal{O}([(1-\gamma)\delta]^{-(2n+m)})$, where γ is the discount factor, δ is the desired value function approximation accuracy, and n and m are the dimensions of the continuous state and action spaces respectively. Secondly, the evaluation of the optimal action requires interpolation between grid points of the successor state in order to estimate the cost-to-go implied by any candidate action.

In continuous state and action spaces, most existing approaches to value function estimation have been motivated by optimal control [7] and reinforcement learning problems [9]. Computing a value function for such problems is attractive because in real-time applications it is not generally possible to

compute an optimal infinite sequence of control inputs for the current system state. Access to a value function, whether exact or approximate, allows the control input to be computed using a one-stage optimization, in which the value function encodes all costs or rewards incurred from the successor state onward.

For some problems, the optimal value function and control policy can, at least for systems of modest size, be calculated exactly – and in the case of the unconstrained linear-quadratic regulator (LQR), are straightforwardly obtained even for large systems via the discrete algebraic Riccati equation [25]. Bemporad *et al.* showed that the value function for finite-horizon constrained LQR is piecewise quadratic, and presented an algorithm for computing the polyhedral "critical regions" of the state space, in each of which the optimal control action is given by a different affine function [4]. This approach is known as *explicit MPC*. As long as a sufficiently large number of time steps are modelled, it can also be shown to produce the exact value function for an infinite-horizon problem within a compact subset of the state space, by virtue of the fact that in a neighbourhood of the origin reachable in a finite number of steps, the optimal control actions are unconstrained [10], [18].

One limitation of explicit MPC is that as the regions are defined by the indices of active constraints in the finite-horizon optimization problem, their number in general grows combinatorially with the number of state and input constraints, and with the time horizon. Also, although this computation is carried out offline, an additional potential difficulty when implementing explicit MPC is the storage in memory, and real-time look-up, of the correct region's control law following measurement of the current state [22]. That said, constructive methods have been developed for approximating optimal control laws using polyhedral computations [23], although mature tools only exist for simplifying the full explicit control law after it has been generated [19]. Lastly, explicit MPC is difficult to extend to nonlinear systems or non-convex value functions, since the piecewise linear or quadratic characterization of the value function no longer holds in general. This has prompted methods such as input quantization to be applied [17], in effect borrowing from standard discretized DP ideas.

For systems too large for the value function and/or policy to be computed exactly using techniques such as the above, a separate group of methods known collectively as *approximate* DP (ADP) has arisen in the literature. ADP is closely related to reinforcement learning [34], in that both fields are concerned with finding approximate value functions and policies for MDPs [9]. The method we present in this article can be considered a novel ADP approach.

Recent approaches to ADP have been based on the so-called linear programming (LP) approach of de Farias and Van Roy [13], in which a maximization of the value function, cast as a sum of appropriate basis functions, is performed subject to a so-called Bellman *inequality* constraint, which uses the Bellman operator's monotonicity property to guarantee that the solution found is an underestimator of the optimal value function. Several variants of this approach have arisen. Wang *et al.* [35] used an *iterated* Bellman inequality to enlarge the feasible set of the problem, Summers *et al.* considered polynomial basis functions and used sum-of-squares programming to compute the value function [33], and Beuchat *et al.* iteratively constructed polynomial lower bounds whose pointwise maximum represents the value function approximation, with numerical emphasis on quadratic functions [8].

Another key development of relevance to the present article is *dual* DP (DDP), first used to solve

multi-stage optimization problems in the area of hydro power planning [26], [27]. This method, also referred to as multi-stage Benders decomposition [5], [16], decomposes the problem into single-stage problems connected by an estimate of the value function of the following stage. It has since found use in further applications such as finance, economics, and power system optimization [12], [15], [36]. The basic algorithm generates a trajectory forward through the planning horizon, in which control decisions are made by solving the single-stage problems, and then performs a backward pass in which a duality argument is used to generate hyperplanes underestimating each stage's value function. At convergence a locally-exact representation of the value function around the optimal trajectory will have been generated [28], [32]. Powell and others have presented adjustments and approximations that aim to improve convergence rate, particularly in a stochastic setting [29], [21], [24], [14].

This article proposes a generalization of DDP, using duality arguments and the properties of the Bellman operator to construct analytical lower bounds on the optimal value function of a deterministic, infinite-horizon control problem over continuous state and action spaces. These bounds arise from the dual solution of the one-stage control problem, and in contrast to the hyperplanes from standard DDP they are generally nonlinear and account for the functional form of the problem's dynamics, costs and constraints. Each iteration uses lower-bounding functions from previous iterations to influence the new bound generated. The algorithm works by increasing the value function estimate anywhere a so-called "Bellman error" [9, §3.6.1], or violation of the Bellman optimality condition, is present, while guaranteeing that the new estimate will still be a global lower bound on the optimal value function. We refer to this method as generalized DDP (GDDP). To the best of our knowledge, duality arguments of this kind have not been used to estimate infinite-horizon value functions before.

The lower-bounding functions generated in our GDDP formulation apply for the (single) infinite-horizon value function of the problem, whereas the hyperplanes generated by DDP are lower bounds on value functions for specific time steps of a multi-stage problem, and are obtained by backward recursion through the planning horizon. In both GDDP and conventional DDP, though, the value function is represented as the pointwise maximum of all lower-bounding functions generated. GDDP could alternatively be applied to finite-horizon problems in which more general lower-bounding functions may be beneficial, for example hydro-power scheduling in the presence of non-convexities [1, §7], but we do not explore this in the present article.

The method offers several potential attractions. Firstly, the solution of each one-stage problem instance generates a more expressive lower bound on the optimal value function than the hyperplanes used in conventional finite-horizon DDP. In contrast, discretization approaches to DP for continuous-state problems do not generate global lower bounds at all.

Secondly, each algorithm iteration has (under standard convexity assumptions) only polynomial complexity in the state and action dimension. This contrasts with the exponential complexity of the standard discretization approach. Although we do not claim to have overcome the so-called curse of dimensionality in this manner – in particular we do not bound the number of iterations required – the algorithm at least avoids the need to interpolate between points on a grid [11], or in the case of local approximation techniques, to locate the nearest sampled point for the purpose of using its associated

local value function estimate [2].[1]

Thirdly, when strong duality holds in the one-stage problem, each step of our algorithm guarantees a strict improvement in the value function estimate for some value of the state $\hat{x}$ at which the Bellman optimality condition does not already hold. This is reminiscent of the well-known improvement property of value function iteration for discrete reinforcement learning problems [34, §4.1].

Lastly, where the input dimension is relatively low, valid lower-bounding functions can be generated even for non-convex value functions of high state dimension, since the one-stage problem, which is solved with the state as a fixed parameter, still has limited complexity. It may even be convex or at least exhibit strong duality in practice. To generate a new lower-bounding function, it need only be possible to extract optimal dual variables from this smaller problem.

A. Contributions

1) We derive a globally-valid lower bounding function for the optimal value function parameterized by the solution of a one-stage control problem, in which the "current" system state appears as a parameter. This lower bound is in general nonlinear, and predicated on one or more existing valid lower bounding functions being available. This generalizes the existing concept of Dual Dynamic Programming, which typically generates lower-bounding hyperplanes, and only in a finite-horizon setting.

2) We provide an iterative algorithm for improving the value function estimate, which is represented as the pointwise maximum of all lower bounds generated so far. We prove that under a strong duality assumption, an iteration of this algorithm causes strict improvement in the value function estimate for any value of the state where the Bellman optimality condition does not yet hold.

3) We provide a generic guarantee that the value function estimate converges over the region of the state space where the optimal value function is finite. We also prove that where strong duality holds in the one-stage problem, the Bellman error, or difference between the two sides of the Bellman optimality condition, converges to zero for all points in the state space visited "infinitely often" by the GDDP algorithm. For a given desired Bellman error tolerance, the algorithm terminates in a finite number of iterations.

4) We demonstrate application of GDDP to numerous case studies, including linear and nonlinear control problems for which the conventional discretization approach to DP is computationally intractable.

B. Article structure

Section II states the infinite-horizon control problem to be studied, and reviews standard results on value functions and the Bellman equation. Section III derives the proposed algorithm for producing successive approximations to the optimal value function, and proves some of its key properties. Section

---

[1]In some circumstances with a discrete action set, random sampling of the state can lead to a DP algorithm for which the solution time remains polynomial in the state dimension, thanks to the scaling properties of Monte Carlo integration in higher dimensions [30]. However to our knowledge there are no such results for problems with continuous state *and* action spaces.

IV discusses implementation choices. Section V presents numerical simulations for linear systems of various sizes as well as a nonlinear example, and Section VI concludes.

C. Notation

The symbol $\mathbb{R}^n$ ($\mathbb{R}^n_+$) represents the field of (non-negative) real vectors in n dimensions, and $\mathbb{S}^n_+$ ($\mathbb{S}^n_{++}$) represent the cone of symmetric (strictly) positive semidefinite $n \times n$ matrices. Inequalities are as follows: $a \le b$ for n-dimensional vectors a and b means $b - a \in \mathbb{R}^n_+$, $A \preceq B$ for $n \times n$ matrices A and B means $B - A \in \mathbb{S}^n_+$, and $A \prec B$ means $B - A \in \mathbb{S}^n_{++}$. For a function $f : \mathbb{R}^n \to \mathbb{R}$, the vector $\nabla_x(y)$ is the gradient of f with respect to x, evaluated at $x = y$. The spectral radius (magnitude of the largest singular value) of a matrix A is denoted $\sigma(A)$.

Notation $\max\{a, b, ...\}$ represents the maximum of scalars a, b, and so on. When $\max\{...\}$ is used with a subscript parameter below the symbol $\max$, it denotes the maximum attained value of the parameterized quantity in braces.

The symbol 0 is used to represent a scalar, vector, or matrix of zeroes of appropriate dimension, and $\mathbf{1}$ represents a vector of ones of appropriate dimension. Symbol I_n denotes the $n \times n$ identity matrix. The notation $\mathrm{diag}\{a, b, \ldots\}$ signifies a diagonal matrix whose diagonal entries are a, b, and so on.

II. PROBLEM STATEMENT

A. Infinite-horizon control problem

We consider the solution of an infinite-horizon deterministic optimal control problem in which stage costs, dynamics, and constraints are time-invariant.

$$\min_{\pi \in \Pi} \quad \mathbb{E}_{x_0 \sim \mathcal{P}_0} \left[\sum_{t=0}^{\infty} \gamma^t \ell(x_t, u_t) \right] \tag{1a}$$

$$\text{s. t.} \quad x_{t+1} = f(x_t, u_t), \quad t = 0, \ldots, \infty, \tag{1b}$$

$$u_t = \pi(x_t), \quad t = 0, \ldots, \infty, \tag{1c}$$

$$E u_t \le h(x_t), \quad t = 0, \ldots, \infty. \tag{1d}$$

The state at time t is denoted $x_t \in \mathbb{R}^n$, and the input $u_t \in \mathbb{R}^m$. Constant $\gamma \in (0, 1]$ is a discount factor for costs encountered later in the horizon, and $\ell : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}_+$ is the stage cost function, which we assume to be non-negative. The expectation is defined over possible initial states x_0, which are weighted according to some probability measure $\mathcal{P}_0$ on $\mathbb{R}^n$.

Although the initial state x_0 is random, variables x_t for $t \ge 1$ and u_t for $t \ge 0$ are uniquely determined by the dynamics $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ and the control policy $\pi : \mathbb{R}^n \to \mathbb{R}^m$, which are time-invariant and common to every possible x_0.

Set Π contains all policies π that are generated by minimizing the sum of the one-stage cost plus a γ-discounted cost-to-go function, $V : \mathbb{R}^n \to \mathbb{R}$:

$$u_t = \pi(x_t) = \arg\min_{u \in \mathcal{U}(x_t)} \{\ell(x_t, u) + \gamma V(f(x_t, u))\} , \tag{2}$$

where

$$\mathcal{U}(x) := \{u : Eu \leq h(x)\} \,, \tag{3}$$

with $E \in \mathbb{R}^{n_c \times m}$ and $h : \mathbb{R}^n \to \mathbb{R}^{n_c}$ mapping the state x to a vector of right-hand-side quantities, where n_c is the number of state-input constraints present. In accordance with other DP literature [9, §2.2], we refer to $\pi(x)$ as the *greedy policy* evaluated at x, since only the stage cost for the immediate time step is modelled explicitly in the calculation. We also refer to function V as an *approximate value function*, as its purpose is to represent all future (discounted) stage costs incurred from controlling the system according to (2).

B. Optimal value function

We define the Bellman operator on function V in terms of the objective value attained in the one-stage problem:

$$\mathcal{T}V(x) := \min_{u \in \mathcal{U}(x)} \{\ell(x, u) + \gamma V(f(x, u))\} \tag{4}$$

We use $V^\star$ to denote the *optimal* value function for problem (1). This function satisfies the Bellman optimality condition,

$$V^\star(x) = \mathcal{T}V^\star(x), \quad \forall x \in \mathcal{X}, \tag{5}$$

where $\mathcal{X}$ is a set containing all trajectories generated by following policy (2) under $V^\star$ from any initial state x_0 in the support of measure $\mathcal{P}_0$.

Assumption 1. *An unique, time-invariant, optimal value function satisfying* (5) *for problem* (1) *exists.*

Assumption 1 is satisfied in various settings. For example, a straightforward guarantee of a unique (and moreover finite) $V^\star(x)$ for all $x \in \mathcal{X}$ is that the stage cost is bounded, and the discount factor γ is strictly less than 1 [6, Prop. 1.2.3]. Alternatively, unbounded functions such as quadratic stage costs over a non-compact $\mathcal{X}$ may be accommodated under different assumptions, such as those of Hernández-Lerma and Lasserre [20, §4.2].

Our later results are generic enough to rely only on Assumption 1 holding, and not on any particular setting. The following property of $V^\star$ follows immediately:

Lemma II.1 (Non-negative optimal value function). *For all $x \in \mathcal{X}$, $V^\star(x) \geq 0$.*

Proof. For any positive value function $V(x)$ we have $\mathcal{T}V(x) = \min_{u \in \mathcal{U}(x)}\{\ell(x, u) + \gamma V(f(x, u))\} \geq 0 \, \forall x \in \mathbb{R}^n$, since $\ell(x, u)$ is non-negative. Therefore $\mathcal{T}^2 V(x)$ is also non-negative, and by induction (under the assumed existence of $V^\star(x)$), repeated application of the Bellman operator on V either converges to a finite fixed point $V^\star$ (see for example Proposition 1.2.1 in [6]), in which case $\lim_{t \to \infty} \mathcal{T}^t V(x) = V^\star(x) \geq 0$, or it diverges to $+\infty$ for states where $V^\star(x) = +\infty$. $\quad\square$

C. Discounted LQR

Problem (1) is difficult to solve in general, and even an approximate solution is difficult to obtain when the state vector has more than just a few dimensions [11]. However, a special case is the

discounted unconstrained linear-quadratic regulator problem, in which $f(x, u) = Ax + Bu$ with (A, B) stabilizable, and $\ell(x, u) = \frac{1}{2}x^\top Q x + \frac{1}{2}u^\top R u$ with $Q \succeq 0$ and $R \succ 0$. For this problem the optimal value function is $V^\star(x) = \frac{1}{2}x^\top P x$, where P is the solution of the (discounted) discrete algebraic Riccati equation. This is obtained in the same manner as for the well-known undiscounted case [25], via the substitutions $\tilde{A} := \sqrt{\gamma}A$ and $\tilde{B} := \sqrt{\gamma}B$.

D. Restriction of problem class

We impose the following additional restrictions before addressing problem (1):

Restriction 1. *The stage cost $\ell(x, u)$ consists of $K \geq 1$ terms, each of which is the pointwise maximum of general functions of x plus linear and quadratic terms in u:*

$$\ell(x, u) = \sum_{k=1}^{K} \ell_k(x, u) \tag{6}$$

where

$$\ell_k(x, u) := \max_l \left\{ \phi_{kl}(x) + r_{kl}^\top u + \frac{1}{2}u^\top R_{kl} u \right\} \tag{7}$$

and the maximum is over a finite number of indices l for each k.

This functional form encompasses, but is not limited to, affinely-scaled p-norms for $p \in \{1, 2, \infty\}$, and convex piecewise affine functions of x and u. It also allows for constant terms within the functions ϕ_{kl}.

Restriction 2. *The dynamics $f(x, u)$ take the following form:*

(a) *If $R_{kl} \succ 0$ for each of the l functions in braces in (7), then $f(x, u) = f_x(x) + F_u(x)u$.*

(b) *Otherwise, the dynamics take the form $f(x, u) = f_x(x) + Bu$, where $B \in \mathbb{R}^{n \times m}$.*

The reasons for Restriction 2 will become clear in the derivation of value function under-approximators in Section III-B.

III. LOWER BOUNDING ALGORITHM

We now describe an algorithm for producing successively better approximations of the optimal value function $V^\star(x)$ for problem (1).

A. Epigraph form of one-stage problem (4)

Let an approximation of the optimal value function be denoted $\hat{V}_I$, taking the form

$$\hat{V}_I(x) = \max_{i=1,\dots,I} g_i(x) \tag{8}$$

for a finite number I of functions $g_i : \mathbb{R}^n \to \mathbb{R}$. We define $g_0(x) = 0$, which by Lemma II.1 is always a valid lower bound on $V^\star(x)$.

Under Restrictions 1 and 2, the *one-stage problem* (4) can be written in the form (9), in which epigraph variables $\beta \in \mathbb{R}^K$ and $\alpha \in \mathbb{R}$ are used to model the stage cost and cost-to-go respectively.

We use form (a) of the dynamics from Restriction 2, noting that the derivation is the same under form (b) but with B replacing $F_u(\hat{x})$.

$$\min_{u, x_+, \beta, \alpha} \quad \mathbf{1}^\top \beta + \gamma \alpha \tag{9a}$$

$$\text{s. t.} \quad x_+ = f_x(\hat{x}) + F_u(\hat{x})u \,, \tag{9b}$$

$$Eu \leq h(\hat{x}) \,, \tag{9c}$$

$$l_j^\top \beta \geq \phi_j(\hat{x}) + r_j^\top u + \tfrac{1}{2} u^\top R_j u \,, \ j = 1, \ldots, n_j \,, \tag{9d}$$

$$\alpha \geq g_i(x_+) \,, \quad i = 1, \ldots, I \,. \tag{9e}$$

This can be considered a parametric optimization problem in which the current state of the system $\hat{x}$ is the (fixed) parameter.

Vector $x_+ \in \mathbb{R}^n$ is the successor state to $\hat{x}$ when input u is applied. In constraint (9d), vector $l_j \in \mathbb{R}^K$ is the kth unit vector (selecting the kth element of vector β) if constraint j is present in the kth summand in (6). This allows all terms contributing to $\ell(x, u)$ to be indexed with a single subscript, $j = 1, \ldots, n_j$, where n_j is the total number of functions needed to model the stage cost in (7).

Lemma III.1. *The optimal value of problem* (9) *for parameter $\hat{x}$ is equal to $\mathcal{T}\hat{V}_I(\hat{x})$. That is, if it is feasible and $(\hat{u}^\star, \hat{x}_+^\star, \hat{\beta}^\star, \hat{\alpha}^\star)$ is an optimal solution,*

$$\mathbf{1}^\top \hat{\beta}^\star + \gamma \hat{\alpha}^\star = \min_{u \in \mathcal{U}(\hat{x})} \left\{ \ell(\hat{x}, u) + \gamma \hat{V}_I(f(\hat{x}, u)) \right\} \,.$$

Proof. The condition $u \in \mathcal{U}(\hat{x})$ is reflected in constraint (9c), and constraint (9b) can be eliminated by substituting the definition of x_+ into constraint (9e). It is immediate that $\mathbf{1}^\top \hat{\beta}^\star = \sum_{k=1}^K \hat{\beta}_k^\star$, where to be feasible and optimal each $\hat{\beta}_k^\star = \max_j \{ \phi_j(\hat{x}) + r_j^\top \hat{u}^\star + \tfrac{1}{2} \hat{u}^{\star \top} R_j \hat{u}^\star \}$ for those entries j corresponding to the kth summand in definition (6). Therefore it follows that $\mathbf{1}^\top \hat{\beta}^\star = \ell(\hat{x}, \hat{u}^\star)$. If $\hat{\alpha}^\star > \max_i g_i(\hat{x}_+^\star) = \max_i g_i(f(\hat{x}, \hat{u}^\star))$ there must be an improving direction for α, and if $\exists i : \hat{\alpha}^\star < g_i(\hat{x}_+^\star)$ then $\hat{\alpha}^\star$ is infeasible. Therefore $\gamma \hat{\alpha}^\star = \gamma \hat{V}_I(f(\hat{x}, \hat{u}^\star))$. $\qquad \square$

B. Lower-bounding lemma

We now derive the dual of problem (9) in order to support the lemma that follows. Assign the multipliers $\pi \in \mathbb{R}^n$ to constraint (9b), $\lambda_c \in \mathbb{R}_+^{n_c}$ to constraint (9c) (where n_c is the number of elements in h), $\lambda_\beta \in \mathbb{R}_+^{n_j}$ to constraint (9d), and $\lambda_\alpha \in \mathbb{R}_+^I$ to constraint (9e). We use $\lambda_{\beta,j}$ to denote the jth element of λ_β, and $\lambda_{\alpha,i}$ to denote the ith element of λ_α.

The Lagrangian of (9) is

$$\mathcal{L}(u, x_+, \beta, \alpha, \pi, \lambda_c, \lambda_\beta, \lambda_\alpha) := \mathbf{1}^\top \beta + \gamma \alpha$$
$$+ \pi^\top (f_x(\hat{x}) + F_u(\hat{x})u - x_+) + \lambda_c^\top (Eu - h(\hat{x}))$$
$$+ \sum_{j=1}^{n_j} \lambda_{\beta,j} \left(\phi_j(\hat{x}) + r_j^\top u + \tfrac{1}{2} u^\top R_j u - l_j^\top \beta \right)$$
$$+ \sum_{i=0}^{I} \lambda_{\alpha,i} \left(g_i(x_+) - \alpha \right)$$
$$= (\mathbf{1} - \overline{L}^\top \lambda_\beta)^\top \beta + (\gamma - \mathbf{1}^\top \lambda_\alpha)\alpha - \pi^\top x_+ + \sum_{i=0}^{I} \lambda_{\alpha,i} g_i(x_+)$$
$$+ (\pi^\top F_u(\hat{x}) + \lambda_c^\top E + \lambda_\beta^\top \overline{R})u + \frac{1}{2} u^\top \left(\sum_{j=1}^{n_j} \lambda_{\beta,j} R_j \right) u$$
$$+ \pi^\top f_x(\hat{x}) - \lambda_c^\top h(\hat{x}) + \lambda_\beta^\top \phi(\hat{x}) \tag{10}$$

where for compactness we have introduced the additional symbols $\overline{L} \in \mathbb{R}^{n_j \times K}$, $\overline{R} \in \mathbb{R}^{n_j \times m}$ and $\phi(\hat{x}) \in \mathbb{R}^{n_j}$, whose j^{th} rows contain $l_j^\top$, $r_j^\top$, and $\phi_j(\hat{x})$ respectively. The dual of problem (9) can then be written:

$$\max_{\pi, \lambda_c, \lambda_\beta, \lambda_\alpha} \quad \phi(\hat{x})^\top \lambda_\beta - h(\hat{x})^\top \lambda_c + f_x(\hat{x})^\top \pi$$
$$+ \zeta_1(\pi, \lambda_\alpha) + \zeta_2(\hat{x}, \pi, \lambda_c, \lambda_\beta) \tag{11a}$$
$$\text{s. t.} \quad \overline{L}^\top \lambda_\beta = \mathbf{1} \tag{11b}$$
$$\mathbf{1}^\top \lambda_\alpha = \gamma \tag{11c}$$
$$\lambda_c \geq 0, \ \lambda_\beta \geq 0, \ \lambda_\alpha \geq 0. \tag{11d}$$

where

$$\zeta_1(\pi, \lambda_\alpha) := \min_{x_+ \in \mathbb{R}^n} \left\{ -\pi^\top x_+ + \sum_{i=0}^{I} \lambda_{\alpha,i} g_i(x_+) \right\} \tag{12}$$

and

$$\zeta_2(\hat{x}, \pi, \lambda_c, \lambda_\beta) := \min_{u \in \mathbb{R}^m} \left\{ (\pi^\top F_u(\hat{x}) + \lambda_c^\top E + \lambda_\beta^\top \overline{R})u + \frac{1}{2} u^\top \left(\sum_{j=1}^{n_j} \lambda_{\beta,j} R_j \right) u \right\} \tag{13}$$

The requirement for the minimizations in (12) and (13) to be bounded from below may place additional implicit constraints on the dual problem. We do not write these constraints out explicitly, because they do not always apply and their form is not in fact relevant. However it is critical for the proof of Lemma III.2 below that they do not depend on parameter $\hat{x}$, since we will require that the constraints of problem (11) are invariant to $\hat{x}$.

This is satisfied for (12), which has no dependence on $\hat{x}$, but consideration of (13) makes the reason for Restriction 2 clear. If all matrices R_j are strictly positive definite (form (a) of the restriction), then

$\zeta_2(\hat{x}, \pi, \lambda_c, \lambda_\beta)$ has a finite analytic value for any $\hat{x}$. If not, then it is generally unbounded from below unless $F_u(\hat{x})^\top \pi + E^\top \lambda_c + \overline{R}^\top \lambda_\beta$ lies in the range space of $\sum_{j=1}^{n_j} \lambda_{\beta,j} R_j$. This range space constraint must then be invariant to $\hat{x}$, which is satisfied if $F_u(\hat{x})$ is a constant (form (b) of the restriction).

Let $J_P(\hat{x})$ denote the optimal objective value of problem (9) as a function of the parameter $\hat{x}$, and similarly define $J_D(\hat{x})$ as the optimal objective value of problem (11). We now state the lemma on which the GDDP algorithm is based.

Lemma III.2 (Lower-bounding lemma). *Suppose $g_i(x) \leq V^\star(x)$, $\forall x \in \mathbb{R}^n$, for $i = 1, \ldots, I$. Then if optimal dual variables exist for problem (11) with parameter $\hat{x}$, denote these $(\hat{\pi}^\star, \hat{\lambda}_c^\star, \hat{\lambda}_\beta^\star, \hat{\lambda}_\alpha^\star)$. The following relationship then holds:*

$$g_{I+1}(x) := \hat{\lambda}_\beta^{\star\top} \phi(x) - \hat{\lambda}_c^{\star\top} h(x) + \hat{\pi}^{\star\top} f_x(x) + \zeta_1(\hat{\pi}^\star, \hat{\lambda}_\alpha^\star) + \zeta_2(x, \hat{\pi}^\star, \hat{\lambda}_c^\star, \hat{\lambda}_\beta^\star) \leq V^\star(x), \; \forall x \in \mathbb{R}^n$$

(14)

Proof. Consider solving problem (11) for some other $\overline{x} \neq \hat{x}$. If the optimal objective value is finite, denote the optimal solution $(\overline{\pi}^\star, \overline{\lambda}_c^\star, \overline{\lambda}_\beta^\star, \overline{\lambda}_\alpha^\star)$. Then

$$J_D(\overline{x}) = \phi(\overline{x})^\top \overline{\lambda}_\beta^\star - h(\overline{x})^\top \overline{\lambda}_c^\star + f_x(\overline{x})^\top \overline{\pi}^\star + \zeta_1(\overline{\pi}^\star, \overline{\lambda}_\alpha^\star) + \zeta_2(\overline{x}, \overline{\pi}^\star, \overline{\lambda}_c^\star, \overline{\lambda}_\beta^\star) \qquad (15a)$$

$$\geq \phi(\overline{x})^\top \hat{\lambda}_\beta^\star - h(\overline{x})^\top \hat{\lambda}_c^\star + f_x(\overline{x})^\top \hat{\pi}^\star + \zeta_1(\hat{\pi}^\star, \hat{\lambda}_\alpha^\star) + \zeta_2(\overline{x}, \hat{\pi}^\star, \hat{\lambda}_c^\star, \hat{\lambda}_\beta^\star). \qquad (15b)$$

where line (15a) is simply the definition of the optimal dual solution, and (15b) follows trivially by virtue of (15a) being optimal. Note that the feasible sets of problem (11) are the same for parameters $\hat{x}$ and $\overline{x}$, and therefore this inequality is always valid.

If the optimal objective value is infinite, i.e. $J_D(\overline{x}) = +\infty$, then the inequality between $J_D(\hat{x})$ and (15b) still holds.

But for any $\overline{x}$, the following relationships hold:

$$J_P(\overline{x}) = \mathcal{T}\hat{V}_I(\overline{x}) \leq \mathcal{T}V^\star(\overline{x}) = V^\star(\overline{x}), \qquad (16)$$

where the left-hand equality arises from Lemma III.1, the central inequality comes from monotonicity of the Bellman operator [6, Lemma 1.1.1], and the right-hand equality comes from the Bellman optimality condition (5).

Furthermore, from weak duality, $J_D(\overline{x}) \leq J_P(\overline{x})$. Therefore, referring to any $\overline{x}$ simply as x, and combining lines (15b) and (16), the result follows. Note we transpose the first three terms in definition (14) as the Lagrange multipliers can be considered coefficients for functions of x. $\qquad \square$

Remark 1 (Alternative form of $g_{I+1}(x)$). *The definition of $g_{I+1}(x)$ can also be written in terms of $J_D(\hat{x})$ and terms related to the difference between x and $\hat{x}$:*

$$g_{I+1}(x) = J_D(\hat{x}) + \hat{\lambda}_\beta^{\star\top}(\phi(x) - \phi(\hat{x})) - \hat{\lambda}_c^{\star\top}(h(x) - h(\hat{x}))$$

$$+ \hat{\pi}^{\star\top}(f_x(x) - f_x(\hat{x})) + \zeta_2(x, \hat{\pi}^\star, \hat{\lambda}_c^\star, \hat{\lambda}_\beta^\star) - \zeta_2(\hat{x}, \hat{\pi}^\star, \hat{\lambda}_c^\star, \hat{\lambda}_\beta^\star) \qquad (17)$$

In case strong duality holds between problems (9) *and* (11)*, $J_P(\hat{x})$ can be used in place of $J_D(\hat{x})$, and the new function can be expressed in terms of the optimal objective value and Lagrange multipliers*

from problem (9).

C. Generalized DDP algorithm

Lemma III.2 states that if functions $g_i(x)$ for $i = 0, \ldots, I$ are all known to be lower bounds on the optimal value function $V^\star(x)$ for all x, then a new lower bound $g_{I+1}(x)$ can be generated from the optimal solution of problem (11) for some (indeed any) choice of parameter $\hat{x}$ if it attains a finite dual optimum. Algorithm 1 constructs a series of globally-valid lower-bounding functions for the optimal value function based on this result, starting with $g_0(x) = 0$, which from Lemma II.1 is a valid lower bound. We refer to this algorithm as Generalized DDP (GDDP), where the generalization in question is the use of nonlinear lower bounding functions directly from the duality argument above, in contrast to the hyperplanes used in conventional DDP.

The purpose of the algorithm is to create an approximate value function which satisfies the Bellman optimality condition at multiple points of interest, the set of which we denote $\mathcal{X}_{\mathrm{Alg}} := \{x_1, \ldots, x_M\}$. Defining the *Bellman error* $\varepsilon(x; V)$ for any value function estimator V and state x as

$$\varepsilon(x; V) := \mathcal{T}V(x) - V(x) \tag{18}$$

the aim is to approach the condition $\varepsilon(x_m, \hat{V}_I) = 0$ for all $x_m \in \mathcal{X}_{\mathrm{Alg}}$. Denoting $\varepsilon_I \in \mathbb{R}^M$ the vector of such Bellman errors at iteration I, the algorithm terminates if $||\epsilon_I||_\infty \leq \delta$, and outputs a final approximation $\hat{V}(x)$.[2]

In the algorithm listing, the function $\mathrm{XPicker}(\mathcal{X}_{\mathrm{Alg}}, \varepsilon_I)$ chooses a member of $\mathcal{X}_{\mathrm{Alg}}$ to use as the parameter $\hat{x}$ when solving the one-stage problem (9) (or its explicit dual (11)). Strategies for selecting $\hat{x}$ are described in Section IV-A. The function $\mathrm{OneStage}(\hat{x}; \hat{V}_I)$ performs this optimization and returns, if optimal dual variables are available, the new lower-bounding function $g_{I+1}(x)$ as described in Lemma III.2.

We now prove some key properties of Algorithm 1.

Lemma III.3 (Positive Bellman error). *For the value function under-approximator $\hat{V}_I(x)$ generated by Algorithm 1 at each iteration I,*

$$\varepsilon(x; \hat{V}_I) \geq 0, \quad \forall x \in \mathbb{R}^n. \tag{19}$$

Proof. For $I = 0$, the result holds trivially since $\hat{V}_0(x) = g_0(x) = 0$, and $\mathcal{T}\hat{V}_0(x)$ is non-negative for any x by virtue of non-negative stage costs $\ell(x, u)$.

For any $I > 0$ we have, from definition (8), $\hat{V}_I(x) \geq \hat{V}_{I-1}(x)$ for all $x \in \mathbb{R}^n$, and therefore, by monotonicity of the Bellman operator, $\mathcal{T}\hat{V}_I(x) \geq \mathcal{T}\hat{V}_{I-1}(x)$ for all $x \in \mathbb{R}^n$. Applying the proof of Lemma III.2 to iteration $I - k$, for any $k = 0, \ldots, I - 1$, we have

$$g_{I-k}(x) \leq J_D(x) \leq J_P(x) = \mathcal{T}\hat{V}_{I-k-1}(x), \quad \forall x \in \mathbb{R}^n.$$

---

[2]For the purpose of assessing convergence, let $\varepsilon(x_m, \hat{V}_I) = 0$ by convention for any x_m where the one-stage problem is infeasible.

Algorithm 1 Generalized Dual Dynamic Programming for M fixed samples

---

1: Set $g_0(x) = 0$
2: Generate samples $\mathcal{X}_{\text{Alg}} := \{x_1, \ldots, x_M\}$
3: Set $I = 0$
4: **while** TRUE **do**
5:      $\hat{V}_I(x) \leftarrow \max_{i=0,\ldots,I} g_i(x)$
6:      **for** $m = 1$ **to** M **do**
7:          $\varepsilon(x_m; \hat{V}_I) \leftarrow \mathcal{T}\hat{V}_I(x_m) - \hat{V}_I(x_m)$
8:      **end for**
9:      **if** $\|\varepsilon_I\|_\infty \le \delta$ **then**
10:          **break**
11:      **end if**
12:      $\hat{x} \leftarrow \text{XPicker}(\mathcal{X}_{\text{Alg}}, \varepsilon_I)$
13:      **if** $\text{OneStage}(\hat{x}; \hat{V}_I)$ is feasible **then**
14:          $g_{I+1}(x) \leftarrow \text{OneStage}(\hat{x}; \hat{V}_I)$ according to (14)
15:      **else**
16:          $V^\star(\hat{x}) = +\infty$; do not revisit $\hat{x}$
17:      **end if**
18:      $I \leftarrow I + 1$
19: **end while**
20: Output $\hat{V}(x) := \max_{i=0,\ldots,I} g_i(x)$

Since $\mathcal{T}\hat{V}_{I-k-1}(x) \le \mathcal{T}\hat{V}_{I-k}(x) \le \ldots \le \mathcal{T}\hat{V}_I(x)$ we have

$$\mathcal{T}\hat{V}_I(x) \ge g_i(x), \quad \forall x \in \mathbb{R}^n, \ \forall i = 0, \ldots, I,$$

from which the result is immediate by the definitions of $\hat{V}_I(x)$ and $\varepsilon(x; \hat{V}_I)$. $\qquad\square$

Lemma III.4 (Strict increase in value function approximator). *Suppose $\hat{V}_I(\hat{x}) < \mathcal{T}\hat{V}_I(\hat{x})$ for some $\hat{x} \in \mathbb{R}^n$, and that $\hat{x}$ is chosen as the evaluation point in iteration $I + 1$ of Algorithm 1. If strong duality holds between problems (9) and (11), then this iteration brings about a strict increase in the value function approximation at $\hat{x}$, i.e., $\hat{V}_{I+1}(\hat{x}) > \hat{V}_I(\hat{x})$. The increase is equal to $\mathcal{T}\hat{V}_I(\hat{x}) - \hat{V}_I(\hat{x})$, i.e.,*

$$\hat{V}_{I+1}(\hat{x}) - \hat{V}_I(\hat{x}) = \mathcal{T}\hat{V}_I(\hat{x}) - \hat{V}_I(\hat{x}) \tag{20}$$

Proof. From definition (8) we have

$$\hat{V}_{I+1}(x) = \max\left\{\hat{V}_I(x), g_{I+1}(x)\right\}.$$

If strong duality holds between problems (9) and (11), i.e., $J_P(\hat{x}) = J_D(\hat{x})$, it follows from (17) that $J_P(\hat{x}) = \mathcal{T}\hat{V}_I(\hat{x}) = g_{I+1}(\hat{x})$. Since we suppose $\hat{V}_I(\hat{x}) < \mathcal{T}\hat{V}_I(\hat{x})$, we will have $g_{I+1}(\hat{x}) > \hat{V}_I(\hat{x})$, and hence $\hat{V}_{I+1}(\hat{x}) > \hat{V}_I(\hat{x})$. Equation (20) follows immediately. $\qquad\square$

We use these lemmas to provide two convergence results for Algorithm 1.

Theorem III.5 (Convergence of $\hat{V}_I(x)$ as $I \to \infty$). *For each $x \in \mathbb{R}^n$ for which $V^\star(x)$ is finite, there exists a limiting value $\hat{V}_{\lim}(x) \le V^\star(x)$ such that $\lim_{I \to \infty} \hat{V}_I(x_m) = \hat{V}_{\lim}(x)$.*

Proof. We show that for any x with finite optimal value, the sequence $(\hat{V}_0(x), \hat{V}_1(x), \ldots)$ is a Cauchy sequence on the real line, and therefore converges to a real number. If $(\hat{V}_0(x), \hat{V}_1(x), \ldots)$ is a Cauchy sequence, there exists an iteration number m_0 such that for any choice of scalar $\delta > 0$, $|\hat{V}_{m_1}(x) - \hat{V}_{n_1}(x)| < \delta$, for all $m_1, n_1 > m_0$.

Suppose for the sake of contradiction that $(\hat{V}_0(x), \hat{V}_1(x), \ldots)$ is not a Cauchy sequence. Then for some $\delta > 0$ there does not exist a sufficiently high m_0. In other words, for any choice of m_0, there exists $m_1, n_1 > m_0$ such that $|\hat{V}_{m_1}(x) - \hat{V}_{n_1}(x)| \geq \delta$. Moreover, after element m_1 of the sequence there must exist another $m_2, n_2 > m_1$ such that $|\hat{V}_{m_2}(x) - \hat{V}_{n_2}(x)| \geq \delta$, otherwise the value of m_1 could have been used in place of m_0.

By induction it is therefore possible to find an arbitrarily high number P of index pairs (m_i, n_i) in this manner, where (without loss of generality) $m_i > n_i$, so that

$$\sum_{i=1}^{P} |\hat{V}_{m_i}(x) - \hat{V}_{n_i}(x)| \geq P\delta.$$

From definition (8), $\hat{V}_I(x)$ increases monotonically with iteration number I. Therefore $|\hat{V}_{m_i}(x) - \hat{V}_{n_i}(x)| = \hat{V}_{m_i}(x) - \hat{V}_{n_i}(x)$ and since each $n_i > m_{i-1}$, we also have $\hat{V}_{n_i}(x) \geq \hat{V}_{m_{i-1}}(x)$. It therefore follows that

$$\hat{V}_{m_P}(x) - \hat{V}_{m_0}(x) \geq P\delta.$$

Since $\hat{V}_{m_P}$ is the pointwise maximum of $m_P + 1$ functions each satisfying inequality (14), $\hat{V}_{m_P}(x) \leq V^\star(x)$. The choice $P > (V^\star(x) - \hat{V}_{m_0}(x))/\delta$ therefore leads to a contradiction. Thus $(\hat{V}_0(x), \hat{V}_1(x), \ldots)$ is a Cauchy sequence, and the limiting value $\hat{V}_{\lim}(x)$ exists. $\square$

Theorem III.6 (Convergence of $\varepsilon(x_m, \hat{V}_I)$ to zero). *Suppose strong duality holds for each one-stage problem* (9). *Then as long as there is no upper bound on the number of times each $x_m \in \mathcal{X}_{\text{Alg}}$ is picked by* $\text{XPicker}(\mathcal{X}_{\text{Alg}}, \varepsilon_I)$ *as I tends to infinity, and each $V^\star(x_m)$ is finite, Algorithm 1 converges in a finite number of iterations for any tolerance $\delta > 0$.*

Proof. Application of Theorem III.5 to point x_m shows that $\lim_{I \to \infty} \hat{V}_I(x_m)$ exists. Each time x_m is picked by $\text{XPicker}(\mathcal{X}_{\text{Alg}}, \varepsilon_I)$, from Lemma III.4 the value function estimate at x_m increases by an amount equal to $\varepsilon(x_m; \hat{V}_I)$, as long as strong duality holds in problem (9) each time it is solved.

Since $(\hat{V}_0(x_m), \hat{V}_1(x_m), \ldots)$ is a Cauchy sequence (even though $\hat{x}$ is not chosen by $\text{XPicker}(\mathcal{X}_{\text{Alg}}, \varepsilon_I)$ every iteration of the algorithm), let $N(x_m, \delta)$ denote the finite iteration number beyond which all differences between later elements of the sequence have magnitude less than δ. By assumption, x_m will be picked at some iteration $I > N(x_m, \delta)$, at which stage we will have $|\hat{V}_{I+1}(x_m) - \hat{V}_I(x_m)| < \delta$. Applying this to the maximum $N(x_m, \delta)$ over all points x_m in the lemma implies that the termination criterion of Algorithm 1 will be satisfied. $\square$

D. Suboptimality of the GDDP algorithm output

Convergence to a final value function approximation $\hat{V}$ does not imply $\hat{V}(\hat{x}) = V^\star(\hat{x})$ for any given $\hat{x}$, even if $\varepsilon(\hat{x}; \hat{V}) = 0$. We now relate $\hat{V}$, the output of Algorithm 1, to the optimal value function $V^\star$, and first provide a lemma in support of this analysis.

Lemma III.7. *For a vector $y \in \mathbb{R}^n$ reachable from x (i.e., for which problem (9) with parameter x is feasible when augmented with the constraint $x_+ = y$), let $\theta(x, y; \hat{V})$ be the increase in optimal cost relative to the original problem (9) when the constraint $x_+ = y$ is added. In other words, $\theta(x, y; \hat{V})$ is the (non-negative) cost of artificially constraining the successor state to be y rather than letting it be chosen freely.*

Then the following relationship between $V^\star$ and $\hat{V}$ holds for x and y:

$$V^\star(x) - \gamma V^\star(y) \leq \hat{V}(x) - \gamma \hat{V}(y) + \theta(x, y; \hat{V}) + \varepsilon(x; \hat{V}) \tag{21}$$

Proof. Let $u^\star(x; \hat{V})$ be a minimizer in (2) for value function $\hat{V}$, and let $u_{x \to y}$ be an input that brings about successor state y at minimum stage cost. From definition (18) we have

$$\hat{V}(x) = \mathcal{T}\hat{V}(x) - \varepsilon(x; \hat{V}) \tag{22a}$$

$$= \ell(x, u^\star(x; \hat{V})) + \gamma \hat{V}(f(x, u^\star(x; \hat{V}))) - \varepsilon(x; \hat{V}) \tag{22b}$$

$$= \ell(x, u_{x \to y}) + \gamma \hat{V}(y) - \varepsilon(x; \hat{V})$$
$$+ \ell(x, u^\star(x; \hat{V})) + \gamma \hat{V}(f(x, u^\star(x; \hat{V}))) - \left(\ell(x, u_{x \to y}) + \gamma \hat{V}(y) \right) \tag{22c}$$

$$= \ell(x, u_{x \to y}) + \gamma \hat{V}(y) - \theta(x, y; \hat{V}) - \varepsilon(x; \hat{V}) \tag{22d}$$

For the optimal value function, similarly defining $u(x; V^\star)$ to be a minimizer in (2) for value function $V^\star$, we have

$$V^\star(x) = \mathcal{T}V^\star(x)$$

$$= \ell(x, u^\star(x; V^\star)) + \gamma V^\star(f(x, u^\star(x; V^\star)))$$

$$\leq \ell(x, u_{x \to y}) + \gamma V^\star(y),$$

where the last line follows from suboptimality of $u_{x \to y}$ and y in the one-stage problem. Hence $\ell(x, u_{x \to y}) \geq V^\star(x) - \gamma V^\star(y)$. Substitution into equation (22d) completes the proof. $\qquad\square$

Corollary III.8. *Suppose there exists a state $\overline{y}$ for which it is known that $\hat{V}(\overline{y}) = V^\star(\overline{y})$. For example, if $\overline{y}$ is an equilibrium and can be maintained while incurring zero stage cost, $\hat{V}(\overline{y}) = V^\star(\overline{y}) = 0$. Then from inequality (21),*

$$V^\star(x) \leq \hat{V}(x) + \theta(x, \overline{y}; \hat{V}) + \varepsilon(x; \hat{V}). \tag{24}$$

Now suppose there exists a feasible state trajectory $(x_0, x_1, \ldots, x_T)$ where $x_T = \overline{y}$. Then, applying Lemma III.7 recursively from any step $t < T$:

$$V^\star(x_t) \leq \gamma V^\star(x_{t+1}) + \hat{V}(x_t) - \gamma \hat{V}(x_{t+1}) + \theta(x_t, x_{t+1}; \hat{V}) + \varepsilon(x_t; \hat{V})$$

$$\leq \gamma \left(\gamma V^\star(x_{t+2}) + \hat{V}(x_{t+1}) - \gamma \hat{V}(x_{t+2}) + \theta(x_{t+1}, x_{t+2}; \hat{V}) + \varepsilon(x_{t+1}; \hat{V}) \right)$$

$$+ \hat{V}(x_t) - \gamma \hat{V}(x_{t+1}) + \theta(x_t, x_{t+1}; \hat{V}) + \varepsilon(x_t; \hat{V})$$

$$\leq \gamma^{T-t} V^\star(x_T) + \sum_{\tau=t}^{T-1} \gamma^{\tau-t} \left(\hat{V}(x_\tau) - \gamma \hat{V}(x_{\tau+1}) + \theta(x_\tau, x_{\tau+1}; \hat{V}) + \varepsilon(x_\tau; \hat{V}) \right)$$

$$= \gamma^{T-t} \left(V^\star(x_T) - \hat{V}(x_T) \right) + \hat{V}(x_t) + \sum_{\tau=t}^{T-1} \gamma^{\tau-t} \left(\theta(x_\tau, x_{\tau+1}; \hat{V}) + \varepsilon(x_\tau; \hat{V}) \right)$$

Since $V^\star(x) \geq \hat{V}(x)$ for any $x \in \mathbb{R}^n$, and $V^\star(x_T) - \hat{V}(x_T) = V^\star(\overline{y}) - \hat{V}(\overline{y}) = 0$, the optimal value function can be bounded from below and above:

$$\hat{V}(x_t) \leq V^\star(x_t) \leq \hat{V}(x_t) + \sum_{\tau=t}^{T-1} \gamma^{\tau-t} \left(\theta(x_\tau, x_{\tau+1}; \hat{V}) + \varepsilon(x_\tau; \hat{V}) \right) \tag{25}$$

Note that the state trajectory in Corollary III.8 need not consist of elements of $\mathcal{X}_{\text{Alg}}$ used in Algorithm 1.

This leads to the following observation.

Remark 2 (Certifying $\hat{V}(x) = V^\star(x)$)**.** *For any x, the value function estimate is exactly correct, i.e. $\hat{V}(x_t) = V^\star(x_t)$, for each step t along any state trajectory starting at $x_0 = x$, satisfying the following conditions:*

1) *The Bellman error $\varepsilon(x_t; \hat{V})$ equals zero at each step t;*
2) *Each state x_{t+1} is an optimal successor state of x_t (i.e., an optimal x_+ in problem (9)), and hence $\theta(x_t, x_{t+1}; \hat{V}) = 0$;*
3) *The final state is $\overline{y}$.*

Unfortunately it is in general difficult to achieve such a certification of exactness using Algorithm 1 for an arbitrary point x_0. Even if Algorithm 1 terminates with $||\varepsilon_I||_\infty = 0$ for the M elements of $\mathcal{X}_{\text{Alg}}$, these are generated when the algorithm is initialized and will not in general contain a sequence arising from the greedy policy (i.e., a sequence for which $\theta(x_\tau, x_{\tau+1}; \hat{V}) = 0$). If the states in the sequence are not elements of $\mathcal{X}_{\text{Alg}}$, then in general we will have $\varepsilon(x_\tau; \hat{V}) > 0$ unless the Bellman optimality condition holds everywhere for $\hat{V}$.

Moreover, the optimal state trajectory for many infinite horizon problems only tends asymptotically to the equilibrium point. Therefore in this case there does not exist a finite sequence satisfying all the conditions in Remark 2.

Remark 3 (Bounding suboptimality at a point)**.** *The following two methods bound $|V^\star(x) - \hat{V}(x)|$ for an arbitary point $x \in \mathbb{R}^n$, assuming $V^\star(x)$ and $\hat{V}(x)$ are both finite and that there exists a state $\overline{y} \in \mathcal{X}_{\text{Alg}}$ used in Algorithm 1 for which it is known that $\hat{V}(\overline{y}) = V^\star(\overline{y})$.*

M1. *Generate a sequence $(x_0, x_1, \ldots)$ by iteratively applying the greedy policy (2) from an initial state $x_0 = x$. Suppose that at some time $T - 1$ the state x_{T-1} is in a neighbourhood of $\overline{y}$, whence there exists a feasible input that brings about $x_T = \overline{y}$. The input at time $T - 1$ will generally be suboptimal compared to the greedy policy.*

Then, for each step τ except $\tau = T - 1$, we have $\theta(x_\tau, x_{\tau+1}; \hat{V}) = 0$, since $x_{\tau+1}$ is the optimal

successor state of x_τ under the greedy policy. From relationship (25) it then holds that

$$\hat{V}(x) \leq V^\star(x) \leq \hat{V}(x) + \gamma^{T-1}\theta(x_{T-1}, \overline{y}; \hat{V}) + \sum_{\tau=0}^{T-1} \gamma^\tau \varepsilon(x_\tau; \hat{V}). \tag{26}$$

M2. *Suppose a feasible trajectory $(x_0, x_1, \ldots, x_T)$ can be constructed where $x_0 = x$, $x_T = \overline{y}$, and $x_1, \ldots, x_{T-1}$ are elements of $\mathcal{X}_{\mathrm{Alg}}$. Then in the limit where Algorithm 1 terminates with $\delta = 0$, we have $\varepsilon(x_\tau; \hat{V}) = 0$ for each $x_\tau \in \mathcal{X}_{\mathrm{Alg}}$, and only x_0, which in general is not an element of $\mathcal{X}_{\mathrm{Alg}}$, will have $\varepsilon(x_0; \hat{V}) > 0$. From relationship (25) it then holds that*

$$\hat{V}(x) \leq V^\star(x) \leq \hat{V}(x) + \varepsilon(x; \hat{V}) + \sum_{\tau=0}^{T-1} \gamma^\tau \theta(x_\tau, x_{\tau+1}; \hat{V}). \tag{27}$$

Methods M1 and M2 are two extreme varieties of bound suggested by (25). In M1, contributions to suboptimality arise primarily from the Bellman errors $\varepsilon(x_\tau; \hat{V})$ along the "greedy policy trajectory". This is in effect the common practice of obtaining an upper bound by simulating the policy and measuring the incurred costs [27], but with more precise treatment of the tail end of the trajectory. In M2, contributions arise primarily from the terms $\theta(x_\tau, x_{\tau+1}; \hat{V})$. These terms, reflecting the added cost of forcing a state transition from x_τ to $x_{\tau+1}$, are generally greater than zero because it is not possible to choose the elements of $\mathcal{X}_{\mathrm{Alg}}$ in advance to coincide with the closed-loop trajectory of the system under $\hat{V}$.

In practice a mixture of methods can be used to construct a bound. Method M1 is used in the closed-loop simulation bounds of Section V-A, with a method resembling M2 employed for the "tail" of each trajectory to force regulation to $\overline{y} = 0$ (see footnote 4). A variant such as this is required in general, because a 1-step transition from an arbitrary state to $\overline{y}$ is only possible for some systems.

Remark 4 (Alternative convergence criterion). *Although the Bellman error is used as the convergence criterion in Algorithm 1, another potential convergence criterion could be to use the upper bounding procedures described in Remark 3. In particular, the upper bound in M1, obtained by using $\hat{V}_I$ to simulate the evolution of the system forward from each of the states $x \in \mathcal{X}_{\mathrm{Alg}}$ is conventional in other DP and DDP literature [27], [8]. However this is more costly to compute than the one-stage Bellman error.*

E. Expressiveness of lower bounds $g_i(x)$

The question arises to what degree a finite number of lower-bounding functions (14) can be expected to match the optimal value function closely (in the sense of a norm defined on the difference), and in particular how many are needed to obtain a desired accuracy.

It is instructive to consider the simple case of the unconstrained LQR problem described in Section II-C. In this case each lower-bounding function (14) generated takes the form $g_i(x) = \frac{1}{2}x^\top Q x + p_i^\top x + s_i$ for some vector p_i and scalar s_i. This is because $\lambda_\beta = 1$, and all other terms are affine in x.

The optimal value function we wish to approximate, $V^\star(x) = \frac{1}{2}x^\top P x$, depends on A, B, Q, and R, and it is easily shown that if $R \succ 0$ then $P \succ Q$, or equivalently $P - Q \succ 0$. Now

suppose that for each element $x_m \in \mathcal{X}_{\text{Alg}}$, the algorithm has terminated with $g_{i_m}(x_m) = V^\star(x_m)$ and $\nabla_x g_{i_m}(x_m) = \nabla_x V^\star(x_m)$ where i_m is the index of the last constraint added at point x_m. In other words suppose a best-case scenario in which the GDDP algorithm matches both the value and gradient of the optimal value function at all M points in $\mathcal{X}_{\text{Alg}}$. It is straightforward to show that in this case,

$$V^\star(x) - \hat{V}(x) = \min_{x_m \in \mathcal{X}_{\text{Alg}}} \left\{ \frac{1}{2}(x - x_m)^\top (P - Q)(x - x_m) \right\}.$$

Defining the norm $||S||_\infty := \max_{x \in \mathcal{X}} |S(x)|$ for a function $S : \mathcal{X} \to \mathbb{R}$ and a compact set $\mathcal{X} \subset \mathbb{R}^n$, the number of points $|\mathcal{X}_{\text{Alg}}|$ required to obtain $||V^\star - \hat{V}||_\infty \le \overline{\delta}$ is equal to the number of ellipsoids $\{x : \frac{1}{2}(x - x_m)^\top (P - Q)(x - x_m) \le \overline{\delta}\}$ required to cover set $\mathcal{X}$. Since this number is in general exponential in the state dimension, we can conclude that to obtain a desired value function approximation using GDDP, at least by this metric, M must also be exponential in the state dimension. Algorithm 1 requires at least one iteration per element of $\mathcal{X}_{\text{Alg}}$ to attain this solution, and therefore this particular aspect of the DP curse of dimensionality [11] persists under GDDP.

IV. Implementation

We now discuss how the several degrees of freedom offered by Algorithm 1 might be treated.

A. Choice of set $\mathcal{X}_{\text{Alg}}$

Algorithm 1 is agnostic to how the set $\mathcal{X}_{\text{Alg}}$ is generated, however it is natural to sample the points independently from the performance weighting (probability) measure $\mathcal{P}_0$ defined in the problem statement (1). This agrees with the typical continuous integral formulation of the LP approach [13], [35], in that as M increases, the objective of (1) is approximated with greater accuracy by the sample average.

However, it may be beneficial to generate $\mathcal{X}_{\text{Alg}}$ in other ways. For example, if the priority is to ensure that the value function can always be upper bounded as in method M2 above, then states could be generated systematically according to a reachability criterion.

B. Sampling from $\mathcal{X}_{\text{Alg}}$

The function XPicker$(\mathcal{X}_{\text{Alg}}, \varepsilon_I)$ chooses which element of $\mathcal{X}_{\text{Alg}}$ to use to derive the next bound $g_{I+1}(x)$. Some possible choices are as follows. In each case, the key requirement for Theorem III.6 to hold is that there be no upper bound on the number of times a given element is visited. Choices 1 and 3 are used in the numerical results of Section V.

1) Return a random choice (equiprobable or weighted) of element of $\mathcal{X}_{\text{Alg}}$ at each iteration.
2) Loop sequentially through all elements of $\mathcal{X}_{\text{Alg}}$ in order, returning to x_1 after finishing an iteration with $\hat{x} = x_M$.
3) Select the element with the largest Bellman error. In this case it will first be necessary to measure the Bellman error $\mathcal{T}\hat{V}_I(\hat{x}) - \hat{V}_I(\hat{x})$ for all elements $\hat{x} \in \mathcal{X}_{\text{Alg}}$.
4) Iterate on the same $\hat{x}$ until its Bellman error $\varepsilon(\hat{x}; \hat{V}_I)$ has reduced to tolerance δ. Then cycle through all points in this manner.

For obvious reasons, choices 3 and 4 are inappropriate if a strictly positive Bellman error is expected to persist (see the strong duality condition of Theorem III.6).

C. Convergence check

Measurement of the Bellman error in lines 6-8 need not be performed at every iteration I. Checking the Bellman error for all elements of $\mathcal{X}_{\mathrm{Alg}}$ requires M solutions of problem (9) or (11), each of which could just as well be used to generate a new lower-bounding function. Therefore it may well be computationally more attractive to spend a greater share of computation time creating lower-bounding functions in line 14, and skip the convergence check during most iterations.

D. Convexity and solver compatibility

Because Algorithm 1 relies on repeated solution of problem (9) or its dual (11), it is attractive for this to be solvable in reasonable time. We make the following observations on achieving this.

Remark 5 (Convexity of problem (9)). *On initialization of the algorithm with $g_0(x) = 0$, the one-stage problem has a convex objective and constraints for any parameter $\hat{x}$ if $R_j \succeq 0$ for all constraints j in (9d). Therefore the only potential source of non-convexity as the algorithm progresses is the introduction of a non-convex lower bounding function $g_i(x)$. In many cases it will be attractive to try and preserve convexity by ensuring that the lower-bounding functions are convex. For this it is sufficient (but not necessary) for all of the following to hold:*

(a) *Function $\phi(x)^\top \hat{\lambda}_\beta^\star$ is convex, e.g. because each element of $\phi_j(x)$ for which the corresponding element of $\hat{\lambda}_\beta^\star$ is non-zero is convex;*

(b) *Function $f_x(x)^\top \hat{\pi}^\star$ is convex, e.g. because $f_x(x)$ is affine, or convex (resp. concave) in x for each element for which the corresponding element of $\hat{\pi}^\star$ is negative (resp. positive);*

(c) *Function $h(x)^\top \hat{\lambda}_c^\star$ is convex, e.g. because $h(x)$ is affine, or convex in x for each element for which the corresponding element of $\hat{\lambda}_c^\star$ is non-zero.*

Even if $g_{I+1}(x)$ is not globally convex, it may still be convex in the region where it is "active", i.e. $g_{I+1}(x) = \max_i g_i(x) = \hat{V}_{I+1}(x)$. In such cases it may be possible to add the new lower bound without dramatically increasing the cost of solving the one-stage problem. Lastly, note that a lower-bounding function that fails to satisfy any convexity test need not be added to the constraint set at all.

Remark 6 (Simplified lower-bounding functions). *If in line 14 of Algorithm 1 the new lower-bounding function (14) to be added is convex, it can be approximated from below by an affine function, whose equation is the first-order Taylor expansion of $g_i(x)$ around $\hat{x}$. Alternatively, a quadratic under-approximator may be available using a second-order expansion. Although these simpler functions will be less tight a lower bound on $V^\star(x)$ than the original one (14), they may change the class of optimization problem (9) or its dual (11) to one for which more efficient solvers exist, for example a linear program or a second-order cone program.*

TABLE I
GDDP ITERATIONS, $\delta = 10^{-3}$, SINGLE SYSTEM INSTANCE

States n	Inputs m	M (number of elements in $\mathcal{X}_{\text{Alg}}$)						
		1	2	5	10	20	50	100
1	1	2	2	6	20	40	110	180
2	1	2	3	7	13	45	123	269
3	1	3	6	13	30	65	160	354
4	2	2	3	6	13	26	66	130
5	2	3	4	14	32	56	167	326
8	3	2	3	6	16	41	115	263
10	4	2	3	8	14	36	100	213

Remark 7 (Redundant lower-bounding functions). *The GDDP algorithm adds a constraint* (9e) *to the one-stage problem at each iteration. It may also be that an existing lower-bounding function $g_i(x)$ is made redundant by the addition of others, i.e. $g_i(x) \leq \max_{j \neq i} g_j(x) \ \forall x$, and can be removed without affecting the solution. As in conventional DDP, then, it may be desirable to modify Algorithm 1 to "prune" redundant lower-bounding functions [14], to the extent that these can be identified efficiently.*

V. NUMERICAL RESULTS

We now present simulated results for constrained linear systems of various sizes, as well as a nonlinear example.

A. Random linear systems

Random asymptotically stable, controllable linear systems (controllable matrix pairs (A, B) with $\sigma(A) \leq 0.99$) were created for different state and input dimensions n and m. The input was bounded with $||u||_\infty \leq 1$, the stage cost was $\ell(x, u) = \frac{1}{2}x^\top x + \frac{1}{2}u^\top u$, and the discount factor was $\gamma = 1$. The elements of $\mathcal{X}_{\text{Alg}}$ were drawn from the same normal distribution as the performance weighting $\mathcal{P}_0 = \mathcal{N}(0, 5^2 \cdot I_n)$, i.e., with standard deviation 5 in all coordinates.

In this setting, each lower-bounding function $g_i(x)$ generated by GDDP is a convex quadratic function, and the one-stage problems take the form of a quadratically-constrained linear program. These were solved in Gurobi 7.0.2, on a computer with a 2.6 GHz Intel Core i7 processor and 16 GB of RAM.

1) Iterations to termination: For a given dimension (n, m), the algorithm was run for an illustrative random system to a tolerance of $\delta = 10^{-3}$. The XPicker$(\mathcal{X}_{\text{Alg}}, \varepsilon_I)$ function used choice 3 described in Section IV-A.[3] Table I shows the iterations required as a function of M in each instance. Although the computational cost of solving the one-stage problem increases with system size, no relationship between system size and the number of iterations required is apparent.

---

[3]Although this requires a relatively costly measurement of all Bellman errors at each iteration, it arguably gives a fairer illustration of the number of iterations required than a random choice of x_m, since it avoids wasting large numbers of iterations revisiting points where the Bellman error is known already to be within the convergence tolerance.

TABLE II
GDDP SOLUTION QUALITY AFTER 200 ITERATIONS; AVERAGES ACROSS 20 RANDOM LINEAR SYSTEMS PER ROW

States n	Inputs m	$M = 200$ elements of $\mathcal{X}_{\text{Alg}}$		1,000 independent samples from $\mathcal{P}_0$		$\mathcal{P}_0/\mathcal{X}_{\text{Alg}}$ subopt. ratio	GDDP time (s)[3]	Val. it. time (s)	MPT time (s)
		RBE (%)[1]	Subopt. bd. (%)[2]	RBE (%)[1]	Subopt. bd. (%)[2]				
1	1	0.04038	0.3197	0.05931	0.3584	1.12	17.06	0.8652	0.8943
2	1	0.3958	2.104	0.586	2.666	1.27	19.10	113.3	3.185
3	1	1.160	6.102	1.807	7.324	1.20	19.34	12,890	19.20
4	2	1.080	4.357	2.068	5.934	1.36	23.68	—[4]	1989
5	2	2.095	7.894	4.192	10.61	1.34	24.70	—[4]	—[4]
8	3	4.695	12.20	10.45	17.49	1.43	34.83	—[4]	—[4]
10	4	5.386	11.96	12.76	17.53	1.47	47.14	—[4]	—[4]

[1]Relative Bellman error, $(\mathcal{T}\hat{V}(x) - \hat{V}(x))/\hat{V}(x)$, averaged across all samples and systems. [2]Suboptimality bound obtained via closed-loop simulation, weighted by $\hat{V}(x)$.
[3]Mean total time taken to generate all lower bounds $g_i(x)$ for systems in each row. [4]Mean time above 6 hours or out of memory.

2) Approximation quality: Although Algorithm 1 converges in finite iterations for a given Bellman error requirement, it is generally more practical to run a fixed number $I_{\max}$ of iterations due to the high computational cost of the convergence check itself (see Section IV-C). Table II shows solution quality statistics for 20 random systems of each dimension with $I_{\max} = 200$ and $M = 200$ samples. The XPicker($\mathcal{X}_{\text{Alg}}, \varepsilon_I$) function chose a random member of $\mathcal{X}_{\text{Alg}}$ as $\hat{x}$ at each iteration (choice 1 of Section IV-A). The table provides average relative Bellman errors, along with a "closed-loop" upper bound on suboptimality generated via the method described in Corollary III.8, simulating the system for 30 time steps from each point $x_m \in \mathcal{X}_{\text{Alg}}$ (50 time steps for the last two rows).[4] If a bound $g_{I+1}(x)$ generated by a given iteration did not increase the value function approximation at x_m by at least 10^{-4}, it was not added to the constraint set.

It is to be expected that since the algorithm fits a value function at the points in $\mathcal{X}_{\text{Alg}}$, the quality of the approximation (in terms of both Bellman error and closed-loop costs) might be poorer on average at other, independently chosen locations in the state space. An unbiased estimate of the solution quality was obtained by measuring the same statistics at 1000 independently sampled points from $\mathcal{P}_0$. The degree of suboptimality was on average within 50% for these independently-generated points, with an increasing trend as system dimension increased. Value function suboptimality was less than 20% on average for the independent samples, across all system sizes tested, although systems with even larger dimension would likely require additional iterations or larger M to achieve comparable performance.

Fig. 1 shows a representative plot of convergence behaviour with regard to the suboptimality bound over 200 iterations in the case of a 5-state, 2-input system.

Two comparisons to existing DP methods were made.

Value iteration: Standard repeated application of the Bellman operator [6, §1.2], was used to derive gridded value function approximations over the compact region $-10\cdot\mathbf{1} \leq x \leq 10\cdot\mathbf{1}$, i.e., two standard deviations from the origin in each coordinate. The state was discretized to 50 equal intervals (51 grid

[4]More precisely, to avoid any kind of truncation effect at the end of the horizon, method M1 in Remark 3 was used until k steps before the end of the horizon, where $k \leq n$ is the number of steps needed to form a full-rank matrix $[B, AB, \ldots, A^{k-1}B]$. An input sequence, which (due to proximity to the origin) was in all cases feasible with respect to the box constraint on the input, was then generated for the last k steps, in the sense of the reachability argument made for method M2.
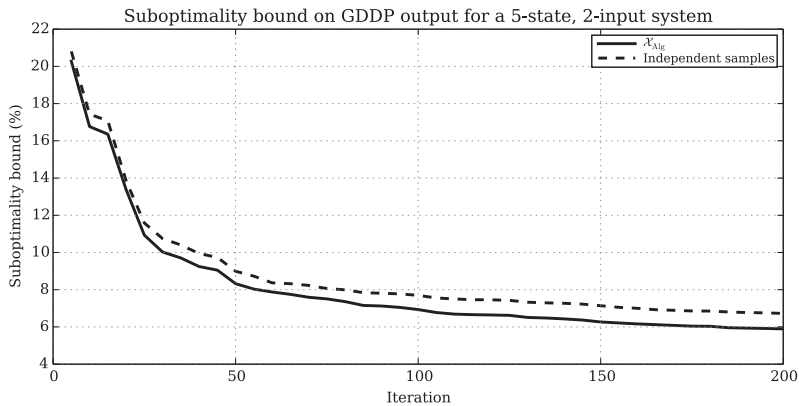
Fig. 1. Representative convergence behaviour of GDDP for a randomly-generated 5-state, 2-input system, in terms of the suboptimality implied by inequality (25) measured from closed-loop simulations. The two lines are $\hat{V}(x)$-weighted averages for the $M = 200$ elements of $\mathcal{X}_{\mathrm{Alg}}$ and the 1000 samples generated independently from $\mathcal{P}_0$. Convergence was evaluated every 5 iterations.

points) in each coordinate, and the input was discretized to 10 equal intervals (11 grid points) in each coordinate. The iterations were terminated when the maximum absolute difference between any grid point's value and its previous value reduced to 10^{-3}. Average total times are reported in the penultimate column of Table II. The computation time grew dramatically with system size, and while more efficient implementations would reduce the times shown, the poor scaling behaviour of this approach is well known and would remain essentially unchanged.

Explicit MPC: For systems of this class, an explicit representation of the optimal value function can be obtained using the Multi-Parametric Toolbox [19] for a given finite optimization horizon. As an illustrative comparison, we provide the mean computation time required to generate this optimal value function for a horizon of 10 steps.[5] Average times are given in the last column of Table II. As with value iteration, the computation time increased dramatically with system dimension, although we note that for systems small enough for the explicit controller to be computed, the online effort to evaluate the resulting policy [22] would be substantially lower than solving (2) with the value function returned by GDDP.

B. Nonlinear system

We demonstrate GDDP for a nonlinear system, namely a discrete-time implementation of the 4-state simplified ball-and-beam example from [31, §10.2]. The state vector is $x = [r, \dot{r}, \theta, \dot{\theta}]^{\top}$ where r is the position in metres along the beam from the pivot, and θ is the angle in radians from horizontal measured counter-clockwise.[6] The single input is a torque τ in Newtons applied at the

---

[5]We note that the region in which the explicit value function is calculated in this manner overlaps only partially with the compact region used for value iteration, and that calculating the number of steps required to cover the same region entirely is itself a computationally expensive exercise.

[6]The simplification from [31] is that the rolling of the ball is modelled as frictionless sliding, to avoid having to include rolling and contact interactions between the ball and beam.

pivot, constrained to an interval $-\tau_{\max} \leq u \leq \tau_{\max}$. The Euler-discretized dynamics are $x_+ = f_x(x) + F_u(x)u$, with the functions defined as follows:

$$f_x(x) = x + \begin{bmatrix} \dot{r} \\ r\dot{\theta}^2 - g\sin\theta \\ \dot{\theta} \\ -\frac{2mr\dot{r} + mgr\cos\theta}{mr^2 + J} \end{bmatrix} \Delta t \,, \quad F_u(x) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{mr^2 + J} \end{bmatrix} \Delta t \,, \qquad (28)$$

where m is the mass of the ball, J is the moment of inertia of the beam, g is gravitational acceleration, and Δt is the time discretization interval.

The parameters used were $m = 0.1$ kg, $J = 0.5$ kg m^2, $g = 9.81$ m s^{-2}, and $\Delta t = 0.1$ s. The cost function used was $\ell(x, u) = \frac{1}{2}x^\top Qx + \frac{1}{2}u^\top Ru$, with $Q = \mathrm{diag}\{10, 1, 1, 1\}$ and $R = 0.01$, and the input constraint was $\tau_{\max} = 3$. The set $\mathcal{X}_{\mathrm{Alg}}$ contained $M = 500$ samples, half of which were drawn from a normal distribution $\mathcal{N}(0, \Xi_1)$ with covariance matrix $\Xi_1 = \mathrm{diag}\{0.5^2, 0.5^2, 0.5^2, 0.5^2\}$, and the other half of which were samples from $\mathcal{N}(0, \Xi_2)$ with $\Xi_2 = \mathrm{diag}\{0.1^2, 0.1^2, 0.1^2, 0.1^2\}$. This choice attached relatively high weight to behaviour around the equilibrium position. The one-stage problems were solved by brute force, meaning no special adpatation to account for the problem structure was made.

Fig. 2 shows regulation of the system to the origin from $x_0 = [1, 0, -0.1745, 0]^\top$ (the ball 1 m to the right of the pivot in a position 10 degrees below horizontal) under the derived control policy. Results are plotted for 100, 200, 300, and 400 GDDP iterations, which took 299, 723, 1206, and 1705 seconds respectively. The control policy after 100 iterations caused the state trajectory to diverge, whereas the control was stabilizing with improving transient performance after higher numbers of iterations.

A linear feedback controller based on the Riccati solution for the system linearized about the origin (see §II-C) was not able to stabilize the system from this initial condition.

VI. CONCLUSION

This article proposed a means of constructing a series of lower bounding functions whose pointwise maximum achieves progressively tighter approximations of the optimal value function for an infinite-horizon control problem. In the linear examples tested, good closed-loop bounds on performance were achieved in only a few dozen iterations.

A number of potential extensions present themselves. Firstly, a stronger connection to the finite-horizon DDP algorithm could be made by considering sequences of states, for example by solving a multi-stage problem in place of the one-stage problem at each iteration of the algorithm. In a finite horizon setting, the conventional DDP algorithm [27] works by refining such a sequence from a known initial state until an upper and lower bound on the optimal trajectory cost have converged, whereas in the GDDP algorithm presented here, the successor states x_+ are not used as sample states for the construction of subsequent lower-bounding functions. The notion of refining policies along state trajectories is well-known in the reinforcement learning literature, where on- and off-policy learning are often contrasted [9]. It may be that a mixture of the two can achieve improvements in our setting.
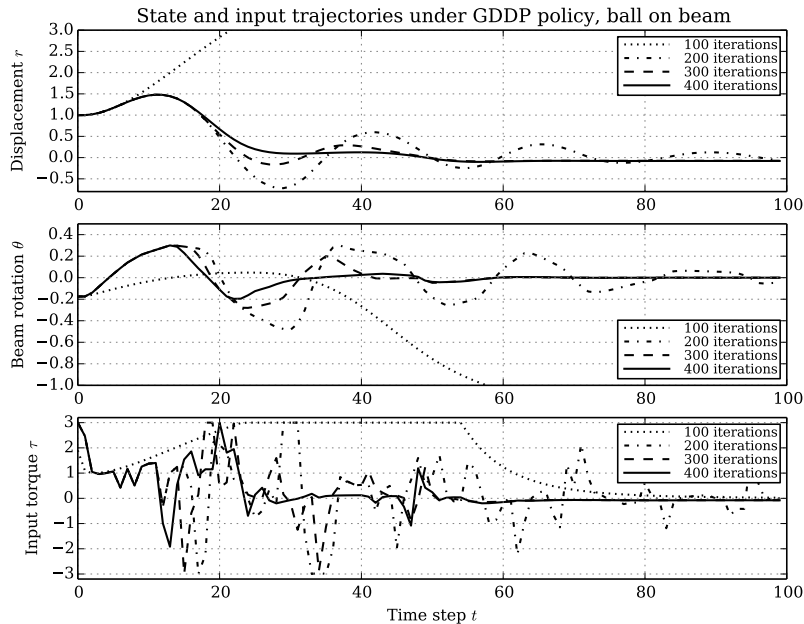
Fig. 2. State and input trajectory from $x_0 = [1, 0, -0.1745, 0]^\top$ under the greedy policy obtained after 100, 200, 300, and 400 iterations. Small residual state offsets after the transient are due to the inexact value function approximation around the origin.

Secondly, it would be attractive to be able to extend the class of lower-bounding functions present in definition (14) to be effective for a wider range of problems with highly non-convex value functions, for example more complex reinforcement learning problems. Presently, a strict increase in the value function lower bound is only guaranteed at a state $\hat{x}$ if strong duality holds in the one-stage problem (9). However, this problem (and its dual) can be defined in numerous ways starting from the original statement (1), and it is not clear whether an alternative formulation might have attractions over the one we have presented.

Thirdly, although we have proven finite convergence of the algorithm (to a desired Bellman tolerance) in the case of strong duality in each one-stage problem instance, we make no claims with regard to the number of iterations required, nor do we derive *a priori* bounds on the number of iterations, or the cardinality of $\mathcal{X}_{\mathrm{Alg}}$, required to achieve a desired value function approximation accuracy in the sense of the discussion in Section III-E. These appear to be difficult problems whose solutions rely on characterizing the accumulating lower-bounding constraints as a function of the original problem data.

REFERENCES

[1] H. Abgottspon. *Hydro power planning: Multi-horizon modeling and its applications.* PhD Thesis, ETH Zurich, Switzerland, 2015.

[2] C. G. Atkeson and B. J. Stephens. Random Sampling of States in Dynamic Programming. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 38(4):924–929, August 2008.

[3] R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[4] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38(1):3–20, January 2002.

[5] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252, December 1962.

[6] D. P. Bertsekas. *Dynamic Programming and Optimal Control, Vol 2*. Athena Scientific, 4 edition, 2012.

[7] D. P. Bertsekas and Steven Shreve. *Stochastic optimal control: the discrete-time case*. Athena Scientific, 2004.

[8] P. N. Beuchat, J. C. Warrington, and J. Lygeros. Point-wise Maximum Approach to Approximate Dynamic Programming. In *IEEE Conference on Decision and Control*, Melbourne, Australia, 2017.

[9] L. Busoniu, R. Babuska, B. De Schutter, and D. Ernst. *Reinforcement learning and dynamic programming using function approximators*, volume 39. CRC press, 2010.

[10] D. Chmielewski and V. Manousiouthakis. On constrained infinite-time linear quadratic optimal control. *Systems & Control Letters*, 29(3):121–129, November 1996.

[11] C. S. Chow and J. N. Tsitsiklis. An optimal one-way multigrid algorithm for discrete-time stochastic control. *IEEE Transactions on Automatic Control*, 36(8):898–914, August 1991.

[12] G. B. Dantzig and G. Infanger. Multi-stage stochastic linear programs for portfolio optimization. *Annals of Operations Research*, 45(1):59–76, December 1993.

[13] D. P. de Farias and B. Van Roy. The Linear Programming Approach to Approximate Dynamic Programming. *Operations Research*, 51(6):850–865, December 2003.

[14] V. L. de Matos, A. B. Philpott, and E. C. Finardi. Improving the performance of Stochastic Dual Dynamic Programming. *Journal of Computational and Applied Mathematics*, 290(Supplement C):196–208, December 2015.

[15] R. Egging. Benders Decomposition for multi-stage stochastic mixed complementarity problems Applied to a global natural gas market model. *European Journal of Operational Research*, 226(2):341–353, April 2013.

[16] A. M. Geoffrion. Generalized Benders decomposition. *Journal of Optimization Theory and Applications*, 10(4):237–260, October 1972.

[17] A. Grancharova and T. A. Johansen. Explicit Approximate Model Predictive Control of Constrained Nonlinear Systems with Quantized Input. In *Nonlinear Model Predictive Control*, Lecture Notes in Control and Information Sciences, pages 371–380. Springer, Berlin, Heidelberg, 2009. DOI: 10.1007/978-3-642-01094-1_30.

[18] P. Grieder, F. Borrelli, F. Torrisi, and M. Morari. Computation of the constrained infinite time linear quadratic regulator. *Automatica*, 40(4):701–708, April 2004.

[19] M. Herceg, M. Kvasnica, C. N. Jones, and M. Morari. Multi-Parametric Toolbox 3.0. In *2013 European Control Conference (ECC)*, pages 502–510, July 2013.

[20] O. Hernández-Lerma and J. B. Lasserre. *Discrete-Time Markov Control Processes: Basic Optimality Criteria*. Applications of Mathematics. Springer, 1996.

[21] T. Homem-de Mello, V. L. de Matos, and E. C. Finardi. Sampling strategies and stopping criteria for stochastic dual dynamic programming: a case study in long-term hydrothermal scheduling. *Energy Systems*, 2(1):1–31, March 2011.

[22] C. N. Jones, P. Grieder, and S. V. Raković. A logarithmic-time solution to the point location problem for parametric linear programming. *Automatica*, 42(12):2215–2218, December 2006.

[23] C. N. Jones and M. Morari. Polytopic Approximation of Explicit Model Predictive Controllers. *IEEE Transactions on Automatic Control*, 55(11):2542–2553, November 2010.

[24] N. Löhndorf, D. Wozabal, and S. Minner. Optimizing Trading Decisions for Hydro Storage Systems Using Approximate Dual Dynamic Programming. *Operations Research*, 61(4):810–823, August 2013.

[25] T. Pappas, A. Laub, and N. Sandell. On the numerical solution of the discrete-time algebraic Riccati equation. *IEEE Transactions on Automatic Control*, 25(4):631–641, August 1980.

[26] M. V. F. Pereira. Optimal stochastic operations scheduling of large hydroelectric systems. *International Journal of Electrical Power & Energy Systems*, 11(3):161–169, July 1989.

[27] M. V. F. Pereira and L. M. V. G. Pinto. Multi-stage stochastic optimization applied to energy planning. *Mathematical Programming*, 52(1-3):359–375, May 1991.

[28] A. B. Philpott and Z. Guan. On the convergence of stochastic dual dynamic programming and related methods. *Operations Research Letters*, 36(4):450–455, July 2008.

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

[30] J. Rust. Using Randomization to Break the Curse of Dimensionality. *Econometrica*, 65(3):487–516, 1997.

[31] S. Sastry. *Nonlinear Systems: Analysis, Stability, and Control*. Springer New York, 1999.

[32] A. Shapiro. Analysis of stochastic dual dynamic programming method. *European Journal of Operational Research*, 209(1):63–72, February 2011.

[33] T. H. Summers, K. Kunz, N. Kariotoglou, M. Kamgarpour, S. Summers, and J. Lygeros. Approximate dynamic programming via sum of squares programming. In *2013 European Control Conference (ECC)*, pages 191–197, July 2013.

[34] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 2 edition, 2017.

[35] Y. Wang, B. O'Donoghue, and S. Boyd. Approximate dynamic programming via iterated Bellman inequalities. *International Journal of Robust and Nonlinear Control*, 25(10):1472–1496, July 2015.

[36] C. Zhao, J. Wang, J. P. Watson, and Y. Guan. Multi-Stage Robust Unit Commitment Considering Wind and Demand Response Uncertainties. *IEEE Transactions on Power Systems*, 28(3):2708–2717, August 2013.