

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

Joseph Warrington^{*†}, *Member, IEEE*, Paul N. Beuchat^{*}, and John Lygeros^{*}, *Fellow, IEEE*

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 spaces, in a discrete-time infinite horizon setting. We prove, using a Benders-type argument leveraging the monotonicity of the Bellman operator, 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 contain terms reflecting 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, and prove under certain assumptions that it achieves an arbitrarily low desired Bellman optimality tolerance at pre-selected points in the state space, in a finite number of iterations. We also describe means of certifying the quality of the approximate value function generated. 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 selection of optimal actions for multi-stage decision problems, by representing future rewards or costs via a *value function*, or cost-to-go function. This allows the next action to be computed using a smaller “one-stage” optimization parameterized by the current system state, without considering the series of future decisions that will follow. This article is concerned with the estimation of value functions for deterministic infinite-horizon problems in discrete time, with continuous state and action (or control input) spaces. This common subclass of Markov decision processes (MDPs) [10] is often used to model problems in optimal control [6] and reinforcement learning [8].

The canonical DP algorithm was originally developed for problems on discrete state and action spaces [2]. For 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 causes two computational issues. Firstly, under quite general assumptions the cost is exponential in the state and action dimensions, $\mathcal{O}([(1 - \gamma)\bar{\delta}]^{-(2n+m)})$, where γ is the discount factor, $\bar{\delta}$ is the desired value function approximation accuracy, and n and m are the continuous state and action dimensions respectively [10, eq. (7.13)]. Secondly,

to compute an optimal action one must interpolate between grid points of the successor state to estimate the cost-to-go.

One therefore seeks more practical DP approaches for continuous problems. In rare cases, the most famous being the unconstrained linear-quadratic regulator (LQR) [22], the optimal value function and control policy can be calculated exactly. The value function for *constrained* LQR is also computable, and is piecewise quadratic [9], [16]. Algorithms exist for computing its polyhedral “critical regions,” in each of which the optimal control action is a different affine function of the state [3]. This computation can also be performed for piecewise linear cost functions. However, the solutions quickly become expensive to compute and store, as the number of regions grows combinatorially with the number of state and input constraints, and with the trajectory length considered. Approximations exist, but can generally be applied only after the full control law has been computed [20]. For almost any other problem, the critical regions are far harder to characterize and compute, and no mature algorithms exist.

A group of methods known as *approximate DP* (ADP) has arisen for continuous problems where the above methods fail. ADP is closely related to reinforcement learning [29], in that both fields are concerned with finding approximate value functions and policies for MDPs [8] using more or less model information. A number of recent ADP methods relevant to the present article build on the so-called linear programming (LP) approach of de Farias and Van Roy [12]. In this approach one maximizes a candidate value function (over some restricted function class) subject to a so-called *Bellman inequality* constraint, guaranteeing that the result lower-bounds the optimal value function. We critique existing methods in this category after describing our contributions below.

The second DP method of relevance to this article is *dual DP* (DDP), first used to solve finite-horizon optimization problems in the area of hydro power planning [23], and since applied in finance [11], economics [14], and power system optimization [32]. DDP splits a multi-stage problem into single-stage problems, each connected to its successor by an estimate of its value function. The algorithm generates a forward trajectory through the planning horizon, in which control decisions are made by solving the single-stage problems, and then performs a backward pass in which Benders’ decomposition [4], [15], is used to generate tighter lower-bounding hyperplanes for each stage’s value function. At convergence one obtains a locally-exact representation of the value function around the optimal trajectory [24], [27].

^{*} Automatic Control Laboratory, Swiss Federal Institute of Technology (ETH) Zurich, Physikstrasse 3, 8092 Zurich, Switzerland. Contact: {warrington, beuchatp, lygeros}@control.ee.ethz.ch

[†] Corresponding author.

A. Contributions

In this article we propose an infinite-horizon, nonlinear generalization of DDP theory, using duality arguments and the properties of the Bellman operator to construct provably-increasing analytical lower bounds on an optimal value function. We refer to this as *generalized DDP* (GDDP). To the best of our knowledge, duality arguments of this kind have not been used to estimate value functions in an infinite-horizon setting before. Specifically, we make the following contributions:

- 1) We show, using a Benders decomposition argument, that a globally-valid lower bound on the optimal value function can be parameterized by the dual solution of a single one-stage control problem, in which the “current” system state appears as a fixed parameter. The lower bound contains terms reflecting the problem’s stage cost, dynamics, and constraints. It is also reflexive, in that it can immediately be used within the same one-stage problem, with the same or a different state parameter, to obtain a new bounding function. This is possible thanks to the monotonicity property of the Bellman operator and the existence of a single value function for the infinite-horizon problem.
- 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. Each algorithm iteration has, under certain assumptions on the problem data, polynomial complexity in the state and action dimension. We prove that under a strong duality assumption, an iteration of this algorithm causes strict improvement in the value function estimate at any value of the state where a so-called Bellman error [8, §3.6.1] is present. This is reminiscent of the well-known improvement property of value function iteration for discrete reinforcement learning problems [29, §4.1].
- 3) We provide a generic guarantee that the value function estimate converges pointwise over the region of the state space where the optimal value function is finite. If strong duality holds in the one-stage problem, the Bellman error converges to zero for all points in the state space revisited with positive probability at each iteration. The GDDP algorithm we define achieves a given strictly positive Bellman error tolerance at all of a finite number of pre-selected state space points, within finite iterations.
- 4) We apply GDDP to linear and nonlinear control problems for which conventional direct DP approaches fail.

GDDP offers several potential attractions. Each lower-bounding function generalizes an estimate of the value function away from the single point in the state space where it was derived, efficiently exploiting the dual result of the one-stage problem. In contrast, “gridding” approaches to DP for continuous-state problems do not generate global lower bounds at all, and have an unattractive up-front exponential cost [10]. 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 [10], or in the case of local approximation techniques, to

identify approximate value functions around nearby sampled points [1].¹

GDDP can be compared with the LP approach to ADP, as both methods maximize a value function while respecting a Bellman inequality condition. Our pointwise maximum representation of the value function is more expressive than the single polynomial employed in [28], and as such we do not rely on high-order parameterizations to obtain a good-quality approximation. It is also clearly more expressive than the earlier quadratic bound [30]. Pointwise maximum representations exist within the LP-approach literature [7], [31]. In [31] an *iterated* Bellman inequality is derived, enlarging the set of feasible parameterizations. However in that formulation one must optimize over all value function “iterations” simultaneously, and the option to take a pointwise maximum is essentially a by-product of the approach. The authors of [7] avoid this simultaneous optimization by converting the result of each iteration into fixed input data for the next. However all of [28], [31], and [7] are difficult to extend beyond polynomial dynamics due to the way they reformulate the Bellman inequality constraint. Another example of an approach accommodating a pointwise maximum representation is [21], in which the Bellman condition is relaxed by a predetermined multiplicative factor. Inner- and outer-approximate value functions are then constructed to satisfy the relaxed condition. Although the algorithmic principle is general, the parameterization of these bounding functions requires application-specific insight. Our Benders argument is distinct from existing ADP literature, leads to implementable algorithms for a wide range of problems, and requires no *a priori* knowledge of the form of the optimal value function. For linear systems, GDDP involves solving more scalable optimization problems than the semidefinite programs of [28], [31], and [7].

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 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 space of (non-negative) n -dimensional vectors, and \mathbb{S}_{++}^n (\mathbb{S}_+^n) represents the cone of symmetric positive (semi-)definite $n \times n$ matrices. Inequality $a \leq b$ for n -dimensional vectors means $b - a \in \mathbb{R}_+^n$, $A \preceq B$ for $n \times n$ matrices means $B - A \in \mathbb{S}_+^n$, and $A \prec B$ means $B - A \in \mathbb{S}_{++}^n$. The spectral radius (magnitude of the largest-magnitude eigenvalue) of a square matrix A is denoted $\sigma(A)$. Notation

¹In 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 [25]. However to our knowledge there are no such results for problems with continuous state *and* action spaces.

$\max\{a, b, \dots\}$ represents the maximum of scalars a , b , and so on. When $\max\{\dots\}$ is used with a subscript parameter below the symbol \max , it denotes the maximum attained value of the parameterized quantity in the braces. The symbol $\mathbf{0}$ represents a vector or matrix of zeroes of appropriate size, and $\mathbf{1}$ represents a vector of ones. Symbol I_n denotes the $n \times n$ identity matrix. The notation $\text{diag}\{a, b, \dots\}$ signifies a diagonal matrix whose 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 the stage cost function, dynamics, and constraints are time-invariant:

$$V^*(x) := \inf_{u_0, u_1, \dots} \sum_{t=0}^{\infty} \gamma^t \ell(x_t, u_t) \quad (1a)$$

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

$$Eu_t \leq h(x_t), \quad t = 0, 1, \dots, \quad (1c)$$

$$x_0 = x. \quad (1d)$$

The state at time t is denoted $x_t \in \mathcal{X} \subseteq \mathbb{R}^n$, and the action, or input, is denoted $u_t \in \mathcal{U} \subseteq \mathbb{R}^m$. Constant $\gamma \in (0, 1]$ is a discount factor for costs encountered later in the horizon, $\ell : \mathcal{X} \times \mathcal{U} \rightarrow \mathbb{R}$ is the stage cost function, and the dynamics are characterized by $f : \mathcal{X} \times \mathcal{U} \rightarrow \mathcal{X}$. The sets \mathcal{X} and \mathcal{U} are the continuous state and action spaces respectively. The state-input constraints (1c) are parameterized by $E \in \mathbb{R}^{n_c \times m}$, where n_c is the number of such constraints present, and a mapping $h : \mathcal{X} \rightarrow \mathbb{R}^{n_c}$ from the state to a vector of right-hand-side quantities. The function $V^*(x)$ is the parametric infimum of problem (1) as x is varied.

B. Optimal value function

We define the Bellman operator \mathcal{T} on a proper function $V : \mathcal{X} \rightarrow \mathbb{R} \cup \{+\infty\}$ in terms of the optimal objective value of a *one-stage problem* related to (1):

$$\mathcal{T}V(x) := \inf_{u \in \mathcal{U}(x)} \{\ell(x, u) + \gamma V(f(x, u))\}, \quad (2)$$

where

$$\mathcal{U}(x) := \{u \in \mathcal{U} : Eu \leq h(x)\}. \quad (3)$$

We refer to function V as an *approximate value function*, as its purpose in (2) is to represent all costs incurred from time $t+1$ onwards along the trajectory. Where the infimum in (2) is attained, one can define a control policy satisfying

$$u = \pi(x) \in \arg \min_{u \in \mathcal{U}(x)} \{\ell(x, u) + \gamma V(f(x, u))\}. \quad (4)$$

In accordance with other DP literature [8, §2.2], we refer to $\pi(x)$ as the *greedy policy* for V , since only the stage cost for the immediate time step is modelled explicitly in the calculation.

An *optimal value function* V^* for problem (1) satisfies the well-known Bellman optimality condition,

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

We make the following assumption:

Assumption 1. *The stage cost function $\ell(\cdot, \cdot)$ is non-negative on its domain, and a unique, time-invariant, optimal value function satisfying (5) exists.*

This existence condition is satisfied in various settings. For example, a straightforward guarantee of a unique V^* is that the stage cost be bounded on \mathcal{X} , and the discount factor γ be strictly less than 1. If in addition $\mathcal{U}(x)$ is non-empty for all $x \in \mathcal{X}$, then V^* is also finite on \mathcal{X} [5, Prop. 1.2.3]. Alternatively, unbounded stage costs such as quadratic functions over a non-compact domain may be accommodated under different assumptions, such as those of Hernández-Lerma and Lasserre [18, §4.2]. Our later results are generic enough to rely only on Assumption 1 itself holding, and not on specific settings such as these.

An additional, obvious property of V^* follows:

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

Proof. The stage costs summed in (1) are non-negative, thus their infimum over control decisions is also non-negative. \square

If the limit under repeated value iteration $\lim_{N \rightarrow \infty} \mathcal{T}^N V(x)$ exists at x , then $V^*(x)$ is finite and equal to this limiting value [5, Prop. 1.2.1]. Otherwise, we say $V^*(x) = +\infty$ and that $V^*(x) = \mathcal{T}V^*(x)$ there by convention.

C. Restriction of problem class

In addition to assuming that the stage cost is non-negative, we restrict it to the following form, which ensures it will be compatible with our use of epigraph variables in Section III.

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 :*

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

where

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

and the maximum is over a finite number of indices l for each k . Each function $\phi_{kl}(x)$ is finite-valued for any given $x \in \mathcal{X}$.

Restriction 1 accommodates, but is not limited to, affinely-scaled p -norms for $p \in \{1, 2, \infty\}$, and convex piecewise affine functions of x and u .

In fact, to aid developments from Section III onward, we will model the stage cost as the sum of K epigraph variables, $\ell(x, u) = \mathbf{1}^\top \beta$ for $\beta \in \mathbb{R}^K$, and replace the K terms of the form (7) with a single list of J constraints on β ,

$$e_j^\top \beta \geq \phi_j(x) + r_j^\top u + \frac{1}{2} u^\top R_j u, \quad j = 1, \dots, J.$$

All indices kl have been rolled into a single index j , and e_j is a unit vector selecting the element of β to which the j^{th} constraint applies.

We also restrict the dynamics:

Restriction 2. The dynamics $f(x, u)$ take one of the following input-affine forms:

- (a) If in (7) we have $R_{kl} \succ 0$ for each index kl , then $f(x, u) = f_x(x) + F_u(x)u$, where $F_u(x)$ has finite entries for any given $x \in \mathcal{X}$.
- (b) Otherwise, the dynamics take the form $f(x, u) = f_x(x) + Bu$, where $B \in \mathbb{R}^{n \times m}$.

In both (a) and (b), $f_x(x)$ has finite entries for any given $x \in \mathcal{X}$.

The reasons for Restriction 2 will become clear in the derivation of lower bounds on $V^*(x)$ in Section III-B.

D. Unconstrained LQR

A special tractable case of problem (1) is the unconstrained LQR problem, in which $f(x, u) = Ax + Bu$ with (A, B) stabilizable, and $\ell(x, u) = \frac{1}{2}x^\top Qx + \frac{1}{2}u^\top Ru$ with $Q \succeq 0$ and $R \succ 0$. For this problem the optimal value function is $V^*(x) = \frac{1}{2}x^\top Px$, where P is the solution of the (discounted) discrete algebraic Riccati equation [22]. The discount factor γ is handled via the substitutions $\hat{A} := \sqrt{\gamma}A$ and $\hat{B} := \sqrt{\gamma}B$.

III. LOWER BOUNDING ALGORITHM

We now describe an algorithm for producing successively better approximations of $V^*(x)$ from below.

A. Epigraph form of one-stage problem (2)

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

$$\hat{V}_I(x) = \max_{i=0, \dots, I} g_i(x) \quad (8)$$

where $g_i : \mathcal{X} \rightarrow \mathbb{R} \cup \{+\infty\}$ are proper functions and I is finite. We define $g_0(x) = 0$, which from Lemma II.1 is always a valid lower bound on $V^*(x)$.

Under Restrictions 1 and 2 the one-stage problem in (2) can be rewritten using epigraph variables $\beta \in \mathbb{R}^K$ and $\alpha \in \mathbb{R}$ to represent the stage cost and cost-to-go respectively. Using dynamics of form (a) from Restriction 2 – the derivation is the same for form (b) – one obtains the following:

$$\inf_{u, x_+, \beta, \alpha} \mathbf{1}^\top \beta + \gamma \alpha \quad (9a)$$

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

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

$$e_j^\top \beta \geq \phi_j(\hat{x}) + r_j^\top u + \frac{1}{2}u^\top R_j u, \quad j = 1, \dots, J, \quad (9d)$$

$$\alpha \geq g_i(x_+), \quad i = 0, \dots, I. \quad (9e)$$

This is a parametric optimization problem in which the current state of the system \hat{x} is the (fixed) parameter. Vector x_+ has been introduced to model explicitly the successor state to \hat{x} when input u is applied.

The following lemma establishes equivalence between (9) and (2).

Lemma III.1. If problem (9) is feasible and an optimal solution $(\hat{u}^*, \hat{x}_+, \hat{\beta}^*, \hat{\alpha}^*)$ is attained for parameter \hat{x} ,

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

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 must hold that $\mathbf{1}^\top \hat{\beta}^* = \sum_{k=1}^K \hat{\beta}_k^* = \ell(\hat{x}, \hat{u}^*)$, and that $\gamma \hat{\alpha}^* = \gamma \hat{V}_I(f(\hat{x}, \hat{u}^*))$, otherwise the epigraph constraints (9d) and (9e) are either violated, or are not binding. \square

B. Lower-bounding lemma

We now derive the dual of problem (9) in order to support the lemma that follows. Assign the multipliers $\nu \in \mathbb{R}^n$ to constraint (9b), $\lambda_c \in \mathbb{R}_+^{n_c}$ to constraint (9c), $\lambda_\beta \in \mathbb{R}_+^J$ to constraint (9d), and $\lambda_\alpha \in \mathbb{R}_+^{I+1}$ to constraint (9e). We use $\lambda_{\beta, j}$ to denote the j^{th} element of λ_β , and $\lambda_{\alpha, i}$ to denote the i^{th} element of λ_α . The Lagrangian of (9) is, after grouping terms,

$$\begin{aligned} \mathcal{L}(u, x_+, \beta, \alpha, \nu, \lambda_c, \lambda_\beta, \lambda_\alpha) := & \\ & (\mathbf{1} - L^\top \lambda_\beta)^\top \beta + (\gamma - \mathbf{1}^\top \lambda_\alpha) \alpha - \nu^\top x_+ + \sum_{i=0}^I \lambda_{\alpha, i} g_i(x_+) \\ & + (\nu^\top F_u(\hat{x}) + \lambda_c^\top E + \lambda_\beta^\top \bar{R})u + \frac{1}{2}u^\top \left(\sum_{j=1}^J \lambda_{\beta, j} R_j \right) u \\ & + \nu^\top f_x(\hat{x}) - \lambda_c^\top h(\hat{x}) + \lambda_\beta^\top \phi(\hat{x}), \end{aligned} \quad (10)$$

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

$$\begin{aligned} \sup_{\nu, \lambda_c, \lambda_\beta, \lambda_\alpha} & \phi(\hat{x})^\top \lambda_\beta - h(\hat{x})^\top \lambda_c + f_x(\hat{x})^\top \nu \\ & + \zeta_1(\nu, \lambda_\alpha) + \zeta_2(\hat{x}, \nu, \lambda_c, \lambda_\beta) \end{aligned} \quad (11a)$$

$$\text{s. t. } L^\top \lambda_\beta = \mathbf{1}, \quad (11b)$$

$$\mathbf{1}^\top \lambda_\alpha = \gamma, \quad (11c)$$

$$\lambda_c \geq 0, \lambda_\beta \geq 0, \lambda_\alpha \geq 0, \quad (11d)$$

where

$$\zeta_1(\nu, \lambda_\alpha) := \inf_{x_+ \in \mathcal{X}} \left\{ -\nu^\top x_+ + \sum_{i=0}^I \lambda_{\alpha, i} g_i(x_+) \right\} \quad (12)$$

and

$$\begin{aligned} \zeta_2(\hat{x}, \nu, \lambda_c, \lambda_\beta) := & \\ & \inf_{u \in \mathcal{U}} \left\{ (\nu^\top F_u(\hat{x}) + \lambda_c^\top E + \lambda_\beta^\top \bar{R})u + \frac{1}{2}u^\top \left(\sum_{j=1}^J \lambda_{\beta, j} R_j \right) u \right\}. \end{aligned} \quad (13)$$

Terms ζ_1 and ζ_2 appear in the dual objective as a standard consequence of minimizing the Lagrangian over primal variables. The requirement for the two minimizations to be bounded from below (which ensures the dual function is meaningful) may place additional implicit constraints on the dual variables in

(11). We do not write these constraints out explicitly, because they do not always apply and their form is not in fact relevant.

It will, however, be crucial for the Benders-type argument of Lemma III.2 below that any extra implicit constraints in (11) are invariant to \hat{x} . Thus (13) could be problematic if the minimization is unbounded for some \hat{x} . If all matrices R_j are strictly positive definite then $\zeta_2(\hat{x}, \nu, \lambda_c, \lambda_\beta)$ has a finite analytic value for any \hat{x} , and no extra constraint coupling the variables \hat{x} , ν , λ_c , and λ_β is needed. If there is an R_j which is *not* strictly positive definite, then one needs $F_u(\hat{x})^\top \nu + E^\top \lambda_c + \bar{R}^\top \lambda_\beta$ to lie in the span of the eigenvectors of $\sum_{j=1}^J \lambda_{\beta,j} R_j$ with strictly positive eigenvalues. This span constraint must then be invariant to \hat{x} , which is satisfied if $F_u(\hat{x})$ is a constant. These issues justify Restriction 2, which is sufficient to ensure that constraint invariance holds.

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 our proposed algorithm is based.

Lemma III.2 (Lower-bounding lemma). *Suppose $g_i(x) \leq V^*(x)$, $\forall x \in \mathcal{X}$, for $i = 0, \dots, I$. Assume that optimal dual variables exist for problem (11) with parameter \hat{x} , and denote these $(\hat{\nu}^*, \hat{\lambda}_c^*, \hat{\lambda}_\beta^*, \hat{\lambda}_\alpha^*)$. Then the following relationship holds:*

$$\begin{aligned} g_{I+1}(x) &:= \hat{\lambda}_\beta^{*\top} \phi(x) - \hat{\lambda}_c^{*\top} h(x) + \hat{\nu}^{*\top} f_x(x) \\ &\quad + \zeta_1(\hat{\nu}^*, \hat{\lambda}_\alpha^*) + \zeta_2(x, \hat{\nu}^*, \hat{\lambda}_c^*, \hat{\lambda}_\beta^*) \\ &\leq V^*(x), \quad \forall x \in \mathcal{X}. \end{aligned} \quad (14)$$

Proof. Consider solving problem (11) for some other $\bar{x} \neq \hat{x}$. If the supremum is attained there, let $(\bar{\nu}^*, \bar{\lambda}_c^*, \bar{\lambda}_\beta^*, \bar{\lambda}_\alpha^*)$ denote an optimal solution. Then

$$\begin{aligned} J_D(\bar{x}) &= \phi(\bar{x})^\top \bar{\lambda}_\beta^* - h(\bar{x})^\top \bar{\lambda}_c^* + f_x(\bar{x})^\top \bar{\nu}^* \\ &\quad + \zeta_1(\bar{\nu}^*, \bar{\lambda}_\alpha^*) + \zeta_2(\bar{x}, \bar{\nu}^*, \bar{\lambda}_c^*, \bar{\lambda}_\beta^*) \end{aligned} \quad (15a)$$

$$\begin{aligned} &\geq \phi(\bar{x})^\top \hat{\lambda}_\beta^* - h(\bar{x})^\top \hat{\lambda}_c^* + f_x(\bar{x})^\top \hat{\nu}^* \\ &\quad + \zeta_1(\hat{\nu}^*, \hat{\lambda}_\alpha^*) + \zeta_2(\bar{x}, \hat{\nu}^*, \hat{\lambda}_c^*, \hat{\lambda}_\beta^*), \end{aligned} \quad (15b)$$

where line (15a) is simply the definition of an optimal dual solution, and (15b) follows trivially as $(\hat{\nu}^*, \hat{\lambda}_c^*, \hat{\lambda}_\beta^*, \hat{\lambda}_\alpha^*)$ cannot be better than any optimal solution. Under Restriction 2 the feasible sets of problem (11) are the same for parameters \hat{x} and \bar{x} , and therefore this inequality always holds.

Even if the supremum in (11) for parameter \bar{x} is not attained, or the optimal objective value is infinite, i.e. $J_D(\bar{x}) = +\infty$, then the inequality between $J_D(\bar{x})$ and (15b) still holds.

Now for any \bar{x} , the following relationships also hold:

$$J_P(\bar{x}) = \mathcal{T}\hat{V}_I(\bar{x}) \leq \mathcal{T}V^*(\bar{x}) = V^*(\bar{x}), \quad (16)$$

where the left-hand equality arises from Lemma III.1, the central inequality comes from monotonicity of the Bellman operator [5, Lemma 1.1.1] and the fact that $\hat{V}_I(\bar{x}) \leq V^*(\bar{x})$, and the right-hand equality comes from the Bellman optimality condition (5).

Furthermore, from weak duality, $J_D(\bar{x}) \leq J_P(\bar{x})$. Therefore, referring to any \bar{x} simply as x , and combining (15b), (15a), weak duality, and (16), the result follows. Note that definition (14) is the same as (15b); we have transposed the

first three terms as the fixed Lagrange multipliers appear as coefficients for functions of x in the algorithm. \square

Remark 1 (Alternative form of $g_{I+1}(x)$). *An equivalent definition of $g_{I+1}(x)$, using $J_D(\hat{x})$ and difference terms between x and \hat{x} , is*

$$\begin{aligned} g_{I+1}(x) &= J_D(\hat{x}) + \hat{\lambda}_\beta^{*\top} (\phi(x) - \phi(\hat{x})) - \hat{\lambda}_c^{*\top} (h(x) - h(\hat{x})) \\ &\quad + \hat{\nu}^{*\top} (f_x(x) - f_x(\hat{x})) \\ &\quad + \zeta_2(x, \hat{\nu}^*, \hat{\lambda}_c^*, \hat{\lambda}_\beta^*) - \zeta_2(\hat{x}, \hat{\nu}^*, \hat{\lambda}_c^*, \hat{\lambda}_\beta^*). \end{aligned} \quad (17)$$

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 $i = 0, \dots, I$ are all known to be global lower bounds on $V^*(x)$, then a new lower bound $g_{I+1}(x)$ can be generated from the solution of dual problem (11) with parameter \hat{x} , as long as that problem attains a finite optimum. Algorithm 1 constructs a series of valid lower-bounding functions based on this result, starting with $g_0(x) = 0$, by solving the one-stage problem at multiple pre-selected points in the state space \mathcal{X} , the set of which we denote $\mathcal{X}_{\text{Alg}} := \{x_1, \dots, x_M\}$. Defining the *Bellman error* $\varepsilon(x; V)$ for any approximate value function V and state x as

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

the algorithm systematically approaches the condition $\varepsilon(x_m, \hat{V}_I) = 0$ for all $x_m \in \mathcal{X}_{\text{Alg}}$. Denoting $\varepsilon_I \in \mathbb{R}^M$ the vector of such Bellman errors at iteration I , the algorithm terminates when $\|\varepsilon_I\|_\infty \leq \delta$, and outputs a final approximation $\hat{V}(x)$.²

In the algorithm listing, the function $\text{XPicker}(\mathcal{X}_{\text{Alg}}, \varepsilon_I)$ chooses an element of \mathcal{X}_{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 $\text{OneStage}(\hat{x}; \hat{V}_I)$ 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 ,*

$$\varepsilon(x; \hat{V}_I) \geq 0, \quad \forall x \in \mathcal{X}. \quad (19)$$

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 $x \in \mathcal{X}$, and therefore, by monotonicity of the Bellman operator, $\mathcal{T}\hat{V}_I(x) \geq \mathcal{T}\hat{V}_{I-1}(x)$ for all $x \in \mathcal{X}$.

²For the purpose of assessing convergence, we 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 algorithm for M fixed state space points

```

1: Generate samples  $\mathcal{X}_{\text{Alg}} := \{x_1, \dots, x_M\}$ 
2: Set  $I = 0$ 
3: Set  $g_0(x) = 0$ 
4: while TRUE do
5:    $\hat{V}_I(x) \leftarrow \max_{i=0, \dots, I} g_i(x)$ 
6:   for  $x_m \in \mathcal{X}_{\text{Alg}}$  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  $\|\varepsilon_I\|_\infty \leq \delta$  then
10:    break
11:  end if
12:   $\hat{x} \leftarrow \text{XPicker}(\mathcal{X}_{\text{Alg}}, \varepsilon_I)$ 
13:  if 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:     $V^*(\hat{x}) = +\infty$ ; do not revisit  $\hat{x}$ 
17:  end if
18:   $I \leftarrow I + 1$ 
19: end while
20: Output  $\hat{V}(x) := \max_{i=0, \dots, I} g_i(x)$ 

```

Applying the proof of Lemma III.2 to iteration $I - k$, for any $k = 0, \dots, I - 1$, we have

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

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

$$\mathcal{T}\hat{V}_I(x) \geq g_i(x), \quad \forall x \in \mathcal{X}, \forall i = 0, \dots, I,$$

from which the result is immediate by the definitions of $\hat{V}_I(x)$ and $\varepsilon(x; \hat{V}_I)$. \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 $\hat{x} \in \mathcal{X}$, 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.,*

$$\hat{V}_{I+1}(\hat{x}) - \hat{V}_I(\hat{x}) = \mathcal{T}\hat{V}_I(\hat{x}) - \hat{V}_I(\hat{x}) > 0. \quad (20)$$

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. \square

We use Lemmas III.2 to III.4 to prove two convergence results for Algorithm 1.

Theorem III.5 (Pointwise convergence of $\hat{V}_I(x)$ as $I \rightarrow \infty$). *For each $x \in \mathcal{X}$ for which $V^*(x)$ is finite, there exists a limiting value $\hat{V}_{\text{lim}}(x) \leq V^*(x)$ such that $\lim_{I \rightarrow \infty} \hat{V}_I(x) = \hat{V}_{\text{lim}}(x)$.*

Proof. From Lemma III.2 and (8), we have that for any x with finite optimal value, the sequence $\{\hat{V}_I(x)\}_{I=0}^\infty$ is bounded from above by $V^*(x)$. Its value is non-decreasing each time a new lower-bounding function $g_{I+1}(x)$ is generated, and therefore the limit $\hat{V}_{\text{lim}}(x)$ exists by the Monotone Convergence Theorem. \square

Theorem III.6 (Finite termination of Algorithm 1). *Suppose the following conditions are met:*

- (i) *Strong duality holds for the one-stage problem (9) with parameter x_m each time it is solved, for all $x_m \in \mathcal{X}_{\text{Alg}}$.*
- (ii) *In the limit as $I \rightarrow \infty$ each $x_m \in \mathcal{X}_{\text{Alg}}$ is picked by $\text{XPicker}(\mathcal{X}_{\text{Alg}}, \varepsilon_I)$ with strictly positive probability at each iteration.*
- (iii) *Each $V^*(x_m)$ is finite.*

Then Algorithm 1 converges in a finite number of iterations with probability 1 for any tolerance $\delta > 0$.

Proof. Application of Theorem III.5 to any $x_m \in \mathcal{X}_{\text{Alg}}$ shows that $\lim_{I \rightarrow \infty} \hat{V}_I(x_m)$ 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).

The convergent sequence $(\hat{V}_0(x_m), \hat{V}_1(x_m), \dots)$ satisfies necessary conditions for a Cauchy sequence, even though x_m is not chosen by $\text{XPicker}(\mathcal{X}_{\text{Alg}}, \varepsilon_I)$ at every iteration of the algorithm. Thus, 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, with probability 1, be picked at some iteration $I > N(x_m, \delta)$, at which stage we will have

$$\varepsilon(x_m; \hat{V}_I) = \hat{V}_{I+1}(x_m) - \hat{V}_I(x_m) = |\hat{V}_{I+1}(x_m) - \hat{V}_I(x_m)| < \delta$$

where the first equality comes from Lemma III.4, the second comes from Lemma III.3, and the inequality comes from the definition of a Cauchy sequence. The constant $N(x_m, \delta)$ is different but finite for each $x_m \in \mathcal{X}_{\text{Alg}}$. Applying this argument to the largest such value over all points $x_m \in \mathcal{X}_{\text{Alg}}$, one deduces that the termination criterion of Algorithm 1 will be satisfied in finite iterations. \square

D. Suboptimality of GDDP output

Convergence to a final value function approximation \hat{V} does not imply $\hat{V}(\hat{x}) = V^*(\hat{x})$ for any given \hat{x} , even if one manages to achieve $\varepsilon(\hat{x}; \hat{V}) = 0$. It is therefore desirable to be able to relate \hat{V} , the output of Algorithm 1, to the optimal value function V^* . We now state a lemma which allows us to derive bounds on the difference, supported by the following definition.

For a vector y reachable from x (i.e., for which problem (9) with parameter x remains feasible when augmented with the constraint $x_+ = y$), let $\theta(x, y; \hat{V})$ be the increase in optimal cost when the constraint $x_+ = y$ is added, relative to the original problem (9). 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.

Lemma III.7. *The optimal value function V^* and approximate value function \hat{V} are related by the following inequality, for any vector pair (x, y) where y is reachable from x :*

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

Proof. Let $u^*(x; \hat{V})$ be a minimizer in (4) for value function \hat{V} , and let $u_{x \rightarrow y}$ be an input that brings about successor state y at minimum stage cost. From definition (18) we have

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

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

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

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

For the optimal value function, similarly defining $u(x; V^*)$ to be a minimizer in (4) for value function V^* , we have

$$V^*(x) = \mathcal{T}V^*(x) \\ = \ell(x, u^*(x; V^*)) + \gamma V^*(f(x, u^*(x; V^*))) \\ \leq \ell(x, u_{x \rightarrow y}) + \gamma V^*(y),$$

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

Corollary III.8. *Suppose there exists a state \bar{y} for which it is known that $\hat{V}(\bar{y}) = V^*(\bar{y})$. Then from inequality (21),*

$$V^*(x) \leq \hat{V}(x) + \theta(x, \bar{y}; \hat{V}) + \varepsilon(x; \hat{V}). \quad (24)$$

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

$$V^*(x_t) \leq \gamma V^*(x_{t+1}) + \hat{V}(x_t) - \gamma \hat{V}(x_{t+1}) \\ + \theta(x_t, x_{t+1}; \hat{V}) + \varepsilon(x_t; \hat{V}) \\ \leq \gamma \left(\gamma V^*(x_{t+2}) + \hat{V}(x_{t+1}) - \gamma \hat{V}(x_{t+2}) \right. \\ \left. + \theta(x_{t+1}, x_{t+2}; \hat{V}) + \varepsilon(x_{t+1}; \hat{V}) \right) \\ + \hat{V}(x_t) - \gamma \hat{V}(x_{t+1}) + \theta(x_t, x_{t+1}; \hat{V}) + \varepsilon(x_t; \hat{V}) \\ \leq \gamma^{T-t} V^*(x_T) + \sum_{\tau=t}^{T-1} \gamma^{\tau-t} \left(\hat{V}(x_\tau) - \gamma \hat{V}(x_{\tau+1}) \right. \\ \left. + \theta(x_\tau, x_{\tau+1}; \hat{V}) + \varepsilon(x_\tau; \hat{V}) \right) \\ = \gamma^{T-t} \left(V^*(x_T) - \hat{V}(x_T) \right) + \hat{V}(x_t) \\ + \sum_{\tau=t}^{T-1} \gamma^{\tau-t} \left(\theta(x_\tau, x_{\tau+1}; \hat{V}) + \varepsilon(x_\tau; \hat{V}) \right)$$

Since $V^(x) \geq \hat{V}(x)$ for any $x \in \mathcal{X}$, and $V^*(x_T) - \hat{V}(x_T) = V^*(\bar{y}) - \hat{V}(\bar{y}) = 0$, the optimal value function can be bounded from below and above:*

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

An example of a suitable \bar{y} is an equilibrium that can be maintained while incurring zero stage cost, so that $\hat{V}(\bar{y}) = V^*(\bar{y}) = 0$. Note that the state trajectory used in Corollary III.8 can be unrelated to \mathcal{X}_{Alg} used in Algorithm 1. This leads to the following observation.

Remark 2 (Certifying $\hat{V}(x) = V^*(x)$). *For any x , the value function estimate is exactly correct, i.e. $\hat{V}(x_t) = V^*(x_t)$, 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 \bar{y} .*

Unfortunately it is in general difficult to achieve this certification for an arbitrary point x . Even if Algorithm 1 terminates with $\|\varepsilon_I\|_\infty = 0$ for the M elements of \mathcal{X}_{Alg} , these will have been generated when the algorithm was initialized, and will not in general contain the required greedy policy sequence satisfying $\theta(x_t, x_{t+1}; \hat{V}) = 0$ for each t . If the states in the sequence are not elements of \mathcal{X}_{Alg} , then in general we will have $\varepsilon(x_t; \hat{V}) > 0$ as the algorithm was not tailored to those points. Moreover, optimal state trajectories for many infinite horizon problems are themselves infinitely long, in which case there does not even exist a finite-time check for the conditions in Remark 2.

We therefore seek more practical interpretations of Corollary III.8. The following observation suggests that a bound can be derived either by playing forward the greedy policy, or by following a pre-constructed feasible trajectory.

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

M1. *Generate a sequence (x_0, x_1, \dots) by iteratively applying the greedy policy (4) from an initial state $x_0 = x$. Suppose that at some time $T - 1$ the state x_{T-1} is in a neighbourhood of \bar{y} , whence there exists a feasible input that brings about $x_T = \bar{y}$. 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. Then from relationship (25),*

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

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

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

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 [23], but with more precise treatment of the tail end of the trajectory. In M2, which could be appropriate in a motion planning or reachability setting, contributions arise primarily from the terms $\theta(x_\tau, x_{\tau+1}; \hat{V})$. These terms reflect the added cost of forcing a state transition from x_τ to $x_{\tau+1}$ in preference to the greedy policy.

In practice a mixture of these methods can be used to construct a bound (see footnote 4 in §V).

Remark 4 (Alternative convergence criterion). *The Bellman error convergence criterion in Algorithm 1 could be replaced by one based on 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 $x \in \mathcal{X}_{\text{Alg}}$ is conventional in other DP and DDP literature [7], [23]. However, it is more costly to compute than the one-stage Bellman error, and does not directly inherit the termination guarantee provided by Theorem III.6.*

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

Algorithm 1 produces lower bounding functions only of the specific form (14), and it may not be possible to represent $V^*(x)$ as their pointwise maximum. One may therefore ask,

- (i) What is the lowest number of points $M = |\mathcal{X}_{\text{Alg}}|$ required to obtain $|V^*(x) - \hat{V}(x)| \leq \bar{\delta}$ for all x in a compact subset of \mathcal{X} , for some $\bar{\delta} > 0$?
- (ii) What is the minimum number of iterations required to achieve this, when these M points are used in the algorithm?

Consider the case of LQR, where $V^*(x) = \frac{1}{2}x^\top Px$ (see §II-D). Inspection of (14) shows that the lower bounds are of the form $g_i(x) = \frac{1}{2}x^\top Qx + p_i^\top x + s_i$ for some $p_i \in \mathbb{R}^n$ and $s_i \in \mathbb{R}$, where Q is the quadratic state cost parameter. It is also straightforward to show that $P - Q \succ 0$, i.e., that $V^*(x)$ has strictly higher curvature in all directions than the functions $g_i(x)$.

To meet the condition in (i), one must generate enough lower-bounding functions for $\max_i g_i(x)$ to stay within $\bar{\delta}$ of $V^*(x)$ everywhere, regardless of the performance of the algorithm. It becomes clear that this is equivalent to covering the state space with ellipsoids characterized by matrix $P - Q$.

The number required is generally exponential in the state dimension. Answering (ii), since each of the lower bounds arises from a GDDP iteration, the number of iterations required is also exponential. Hence, GDDP does not overcome this aspect of the well-known curse of dimensionality [10].

IV. IMPLEMENTATION

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

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

Algorithm 1 is agnostic to how the set \mathcal{X}_{Alg} is generated. One natural choice is to sample the points independently from a performance (or state relevance) weighting \mathcal{P}_0 , which defines states x where one wishes to minimize the parametric cost of problem (1). The rationale for this is that by running Algorithm 1 on these points, one obtains a small Bellman error there; one might then assume from the Bellman optimality condition (5) that this leads to a good approximation of V^* at the same points.

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

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

The function $\text{XPicker}(\mathcal{X}_{\text{Alg}}, \varepsilon_I)$ chooses which element of \mathcal{X}_{Alg} to use to derive the next bound $g_{I+1}(x)$. For example:

- 1) Return a random (equiprobable or with strictly positive weights) element of \mathcal{X}_{Alg} at each iteration.
- 2) Loop sequentially through all elements of \mathcal{X}_{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.
- 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.

Note that strictly speaking Choices 2 and 4 additionally require $\text{XPicker}(\mathcal{X}_{\text{Alg}}, \varepsilon_I)$ to be supplied with knowledge of previous iterations.

Choice 1 fulfils the conditions of Theorem III.6, and Choice 2 also converges under a trivial adaptation of the same theorem. Choices 3 and 4 are clearly inappropriate if the strong duality condition of Theorem III.6 is not satisfied, as this will cause a persistent Bellman error. Results for Choices 1 and 3 are reported in Section V.

C. Convergence check

Measurement of the Bellman error in lines 6-8 need not be performed at every iteration I . A measurement for all elements of \mathcal{X}_{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. It may be more attractive to spend a greater share of computation time creating lower-bounding functions, and check convergence less frequently.

D. Convexity and solver compatibility

Because Algorithm 1 relies on repeated solution of problem (9) or its dual (11), one may wish to minimize solution time by considering convexity, problem class, or the accumulation of lower-bounding functions.

1) *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 bound $g_i(x)$. Inspection of (14) indicates that to preserve convexity it is sufficient, but not necessary, for all of $\phi(x)^\top \hat{\lambda}_\beta^*$, $f_x(x)^\top \hat{\nu}^*$, and $h(x)^\top \hat{\lambda}_c^*$ to be convex. This holds in many cases, e.g. $h(x)$ and $f_x(x)$ affine and $\phi(x)$ convex. Even if $g_{I+1}(x)$ is not globally convex, it may still be convex in the region where it is “active”, i.e. where $g_{I+1}(x) = \max_i g_i(x) = \hat{V}_{I+1}(x)$. Lastly, one may choose not to add a lower-bounding function that fails a convexity test.

2) *Simplified lower-bounding functions*: If in line 14 the new lower bound (14) can be approximated from below by its first- or second-order Taylor expansion around \hat{x} , the simpler functions may usefully restrict problem (9) or its dual (11) to a class for which more efficient solvers exist, for example a linear or quadratically-constrained program. These simpler functions will however be less tight a lower bound on $V^*(x)$.

3) *Redundant lower-bounding functions*: The GDDP algorithm adds a constraint (9e) to the one-stage problem at each iteration, which may make an existing one redundant, i.e. $g_i(x) \leq \max_{j \neq i} g_j(x) \forall x \in \mathcal{X}$. As in conventional DD, it may be desirable to “prune” these redundant lower-bounding functions [13] where they 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 pairs (A, B) with $\sigma(A) \leq 0.99$) were created for different state and input dimensions n and m . The state and input spaces were $\mathcal{X} = \mathbb{R}^n$ and $\mathcal{U} = \mathbb{R}^m$ respectively. Constraint (1c) was used to bound the input 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}_{Alg} were drawn from a normal distribution $\mathcal{P}_0 = \mathcal{N}(0, 5^2 \cdot I_n)$. This distribution can be viewed as a weighting of the states for which we wish to solve problem (1); see Section IV-A.

For these systems, each lower-bounding function $g_i(x)$ generated by GDDP is a convex quadratic, and the one-stage problems (9) are quadratically-constrained linear programs. 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.

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

States n	Inputs m	M (number of elements in \mathcal{X}_{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

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-B.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, the number of iterations appears to be roughly linear in M and unrelated to n and m .

2) *Approximation quality*: The quality of the value function \hat{V} output by the GDDP algorithm can be measured by taking an expectation of the infinite-horizon cost over initial states $x_0 \sim \mathcal{P}_0$. For any sample x_0 , we used inequality (25) to bound the suboptimality of this infinite-horizon cost with respect to the solution of (1).⁴

Table II compares Bellman errors and the suboptimality bounds in percentage terms for

- Set \mathcal{X}_{Alg} , consisting of $M = 200$ points drawn from \mathcal{P}_0 .
- An “out-of-sample” evaluation set of 1,000 points drawn independently from the same distribution.

Choice 1 was used for XPicker($\mathcal{X}_{\text{Alg}}, \varepsilon_I$), i.e. an equiprobable random choice of x_m at each iteration. The quantities reported were measured after 200 iterations, such that each element of \mathcal{X}_{Alg} was visited once on average. Each row of the table reports mean values over 20 random systems.

The algorithm fits a lower-bounding function at the points in \mathcal{X}_{Alg} , and as a result the suboptimality bounds computed are a little worse at the out-of-sample points (17.5% vs. 12.0% for the largest system studied). Fig. 1 shows a representative plot of convergence of the suboptimality bound in the case of a 5-state, 2-input system.

In addition, Table II includes comparisons with two other DP methods.

Value iteration: Standard repeated application of the Bellman operator [5, §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 of \mathcal{P}_0 from the

³Although this requires a relatively costly measurement of all Bellman errors at each iteration, it arguably gives a fairer illustration of the best-case number of iterations required than a random choice of x_m .

⁴In these tests, we simulated the system for 30 time steps from each sampled starting state (50 time steps for the last two rows). 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 $[B, AB, \dots, A^{k-1}B]$. 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.

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}_{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	0.5860	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

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

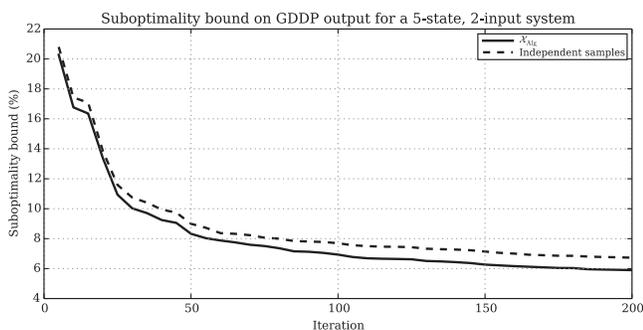


Fig. 1. Representative convergence behaviour of GDDP for a randomly-generated 5-state, 2-input system, in terms of the suboptimality implied by inequality (25) measured from closed-loop simulations. The two lines are $\hat{V}(x)$ -weighted averages for the $M = 200$ elements of \mathcal{X}_{Alg} and the 1000 samples generated independently from \mathcal{P}_0 . Convergence was evaluated every 5 iterations.

origin in each coordinate. The state was discretized to 50 equal intervals (51 grid 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 change in value fell below 10^{-3} . Average total times are reported in the penultimate column of Table II. The computation time grew dramatically with system size, and while a more efficient implementation would likely reduce the times shown, the poor scaling behaviour of this approach is well known and would remain essentially unchanged.

Parametric solution: For systems of this class, an explicit representation of the optimal value function can be obtained using the Multi-Parametric Toolbox [17] 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. 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. 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 [19] would be substantially lower than

solving (4) with the approximate value function returned by GDDP.

B. Nonlinear system

We demonstrate GDDP for the 4-state simplified ball-and-beam example from [26, §10.2]. The state vector is $x = [r, \dot{r}, \theta, \dot{\theta}]^T$ 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. The simplification we adopt from [26] is to model the rolling of the ball as frictionless sliding, to avoid having to include rolling and contact interactions between the ball and beam.

The single input is a torque τ in Newtons applied at the 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

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

where m is the mass of the ball, J_b 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, $J_b = 0.5$ kg m², $g = 9.81$ m s⁻², and $\Delta t = 0.1$ s. The cost function is $\ell(x, u) = \frac{1}{2}x^T Qx + \frac{1}{2}u^T Ru$, with $Q = \text{diag}\{10, 1, 1, 1\}$ and $R = 0.01$, and the input constraint was $\tau_{\max} = 3$. The set \mathcal{X}_{Alg} contained $M = 500$ samples, half of which were drawn from a normal distribution $\mathcal{N}(0, \Xi_1)$ with covariance matrix $\Xi_1 = \text{diag}\{0.5^2, 0.5^2, 0.5^2, 0.5^2\}$, and the other half of which were samples from $\mathcal{N}(0, \Xi_2)$ with $\Xi_2 = \text{diag}\{0.1^2, 0.1^2, 0.1^2, 0.1^2\}$. This choice attached relatively high weight to behaviour around the equilibrium position. The one-stage problems were solved by brute force, with no special adaptations to account for the problem structure.

Fig. 2 shows regulation of the system to the origin from $x_0 = [1, 0, -0.1745, 0]^T$ (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

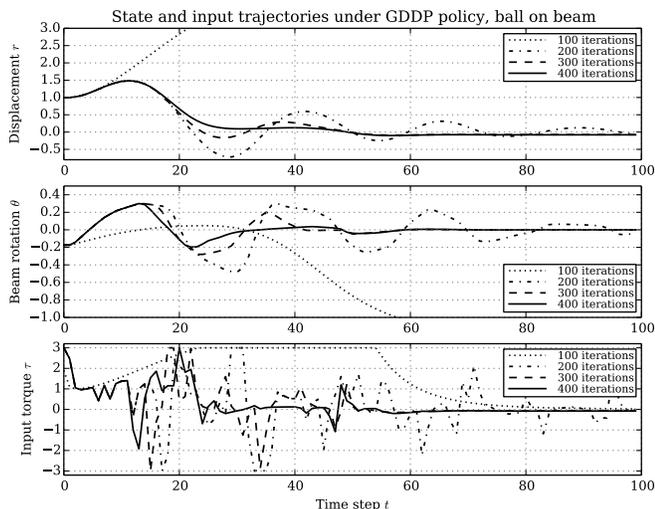


Fig. 2. State and input trajectory from $x_0 = [1, 0, -0.1745, 0]^T$ under the greedy policy obtained after 100, 200, 300, and 400 iterations. Small residual state offsets after the transient are due to the inexact value function approximation around the origin.

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-D) 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 [23] 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 [8]. It may be that a mixture of the two can achieve improvements in our setting.

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 those encountered in 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 do not derive *a priori* bounds on the number of iterations, or the cardinality of \mathcal{X}_{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] 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.
- [2] R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.
- [3] 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.
- [4] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252, December 1962.
- [5] D. P. Bertsekas. *Dynamic Programming and Optimal Control, Vol 2*. Athena Scientific, 4 edition, 2012.
- [6] D. P. Bertsekas and S. Shreve. *Stochastic optimal control: the discrete-time case*. Athena Scientific, 2004.
- [7] 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.
- [8] L. Busoniu, R. Babuska, B. De Schutter, and D. Ernst. *Reinforcement learning and dynamic programming using function approximators*, volume 39. CRC press, 2010.
- [9] D. Chmielewski and V. Manousiouthakis. On constrained infinite-time linear quadratic optimal control. *Systems & Control Letters*, 29(3):121–129, November 1996.
- [10] 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.
- [11] G. B. Dantzig and G. Infanger. Multi-stage stochastic linear programs for portfolio optimization. *Annals of Operations Research*, 45(1):59–76, December 1993.
- [12] D. P. de Farias and B. Van Roy. The Linear Programming Approach to Approximate Dynamic Programming. *Operations Research*, 51(6):850–865, December 2003.
- [13] 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.
- [14] 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.
- [15] A. M. Geoffrion. Generalized Benders decomposition. *Journal of Optimization Theory and Applications*, 10(4):237–260, October 1972.
- [16] 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.
- [17] 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.
- [18] O. Hernandez-Lerma and J. B. Lasserre. *Discrete-Time Markov Control Processes: Basic Optimality Criteria*. Applications of Mathematics. Springer, 1996.
- [19] C. N. Jones, P. Grieder, and S. V. Rakovi. A logarithmic-time solution to the point location problem for parametric linear programming. *Automatica*, 42(12):2215–2218, December 2006.
- [20] C. N. Jones and M. Morari. Polytopic Approximation of Explicit Model Predictive Controllers. *IEEE Transactions on Automatic Control*, 55(11):2542–2553, November 2010.

- [21] B. Lincoln and A. Rantzer. Relaxing dynamic programming. *IEEE Transactions on Automatic Control*, 51(8):1249–1260, 2006.
- [22] 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.
- [23] 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.
- [24] 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.
- [25] J. Rust. Using Randomization to Break the Curse of Dimensionality. *Econometrica*, 65(3):487–516, 1997.
- [26] S. Sastry. *Nonlinear Systems: Analysis, Stability, and Control*. Springer New York, 1999.
- [27] A. Shapiro. Analysis of stochastic dual dynamic programming method. *European Journal of Operational Research*, 209(1):63–72, February 2011.
- [28] 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.
- [29] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 2 edition, 2017.
- [30] Y. Wang and S. Boyd. Performance bounds for linear stochastic control. *Systems & Control Letters*, 58(3):178–182, March 2009.
- [31] 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.
- [32] 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.



Joseph Warrington is a Senior Scientist in the Automatic Control Lab (IfA) at ETH Zurich, Switzerland. His Ph.D. is from ETH Zurich (2013), and his B.A. and M.Eng. degrees in Mechanical Engineering are from the University of Cambridge (2008). From 2014-2016 he worked as an energy consultant at Baringa Partners LLP, London, UK, and he has also worked as a control systems engineer at Wind Technologies Ltd., Cambridge, UK, and privately as an operations research consultant. He is the recipient of the 2015 ABB Research Prize for an outstanding

PhD thesis in automation and control, and a Simons-Berkeley Fellowship for the period January-May 2018. His research interests include dynamic programming, large-scale optimization, and predictive control, with applications including power systems and transportation networks.



Paul N. Beuchat received the B.Eng. degree in mechanical engineering and B.Sc. in physics from the University of Melbourne, Australia, in 2008, and the M.Sc. degree in robotics, systems and control from ETH Zurich, Switzerland, in 2014, where he is currently working towards the Ph.D. degree at the Automatic Control Laboratory. From 2009-2012 he worked as a subsurface engineer for ExxonMobil, based in Melbourne, Australia. His research interests are control and optimization of large scale systems, with a focus towards developing approximate dynamic programming techniques for applications in the areas of power systems, building control, and coordinated flight.



John Lygeros completed a B.Eng. degree in electrical engineering in 1990 and an M.Sc. degree in Systems Control in 1991, both at Imperial College of Science Technology and Medicine, London, UK. In 1996 he obtained a Ph.D. degree from the Electrical Engineering and Computer Sciences Department, University of California, Berkeley. During the period 1996–2000 he held a series of post-doctoral researcher appointments at the Laboratory for Computer Science, M.I.T., and the Electrical Engineering and Computer Sciences Department at U.C. Berkeley. Between 2000 and 2003 he was a University Lecturer at the Department of Engineering, University of Cambridge, UK, and a Fellow of Churchill College. Between 2003 and 2006 he was an Assistant Professor at the Department of Electrical and Computer Engineering, University of Patras, Greece. In July 2006 he joined the Automatic Control Laboratory at ETH Zurich, where he is currently serving as the Head of the Automatic Control Laboratory and the Head of the Department of Information Technology and Electrical Engineering. His research interests include modelling, analysis, and control of hierarchical, hybrid, and stochastic systems, with applications to biochemical networks, automated highway systems, air traffic management, power grids and camera networks. John Lygeros is a Fellow of the IEEE, and a member of the IET and the Technical Chamber of Greece; since 2013 he has served as the Treasurer of the International Federation of Automatic Control.