Accelerated Point-wise Maximum Approach to Approximate Dynamic Programming

Originally published in:
IEEE Transactions on Automatic Control 67(1), https://doi.org/10.1109/TAC.2021.3050440

Funding acknowledgement:
787845 - Optimal control at large (EC)

Accelerated Point-wise Maximum Approach to Approximate Dynamic Programming

Paul N. Beuchat[1], *Member, IEEE*, Joseph Warrington[1], *Member, IEEE*, and John Lygeros[1], *Fellow, IEEE*

***Abstract*—We describe an approximate dynamic programming approach to compute lower bounds on the optimal value function for a discrete time, continuous space, infinite horizon setting. The approach iteratively constructs a family of lower bounding approximate value functions by using the so-called Bellman inequality. The novelty of our approach is that, at each iteration, we aim to compute an approximate value function that maximizes the point-wise maximum taken with the family of approximate value functions computed thus far. This leads to a non-convex objective, and we propose a gradient ascent algorithm to find stationary points by solving a sequence of convex optimization problems. We provide convergence guarantees for our algorithm and an interpretation for how the gradient computation relates to the state-relevance weighting parameter appearing in related approximate dynamic programming approaches. We demonstrate through numerical examples that, when compared to existing approaches, the algorithm we propose computes tighter sub-optimality bounds with comparable computation time.**

I. Introduction

A. Motivation

Many important challenges in science and engineering can be cast in the problem formulation of infinite horizon stochastic optimal control (SOC), from climate control of a building [1] to control of cell populations [2]. The goal of such problems is to find a state feedback policy that minimizes an infinite-horizon discounted cost function. For a general SOC problem instance, the solution, i.e., the optimal policy, is typically characterized by the theory of dynamic programming (DP) [3]–[5]. However, in all but a few special cases, solving the SOC directly or applying the DP theory is intractable due to the so-called *curse of dimensionality*. As such, an extensive body of literature has proposed approximation techniques for computing sub-optimal solutions to infinite horizon SOC problems, ranging from model-free and simulation-based algorithms [6], to model-based approaches [7]–[9]. Many simulation-based algorithms first select a policy parametrization, for example neural networks, and then search for parameters with the best performance, [6], [10], [11]. A range of model-based approaches instead address a finite horizon SOC counterpart, many of which use optimization-based policies to approximate the finite horizon SOC [12]–[14]. Any technique based on DP theory falls in the category of Approximate DP (ADP), see [11], [15], [16] for an overview. Although the optimal policy is intractable to compute, techniques have been developed to bound the sub-optimality of an approximate policy. Such bounds provide the designer with valuable information about the potential benefit of synthesizing and evaluating alternative policies. In this paper we consider infinite horizon SOC and propose approaches that provide sub-optimality bounds based on the so-called *Linear Programming (LP) approach to ADP* [17].

This work was supported by the European Research Council under the project OCAL, grant number 787845.

[1] The authors are with the Automatic Control Laboratory at ETH Zürich, Switzerland, jlygeros@ethz.ch

B. Prior Work

The LP approach to ADP provides sub-optimality bounds by computing approximations that are lower bounds of the so-called *value function*, i.e., the solution of the Bellman equation for DP [3]. The LP approach parameterizes approximate value functions as a linear combination of fixed basis functions and uses the so-called *Bellman inequality* to restrict consideration to only those linear combinations that are point-wise lower bounds of the value function. The appropriate approximation steps for ensuring that the Bellman inequality is satisfied depends on the problem class. Previous works developed such approximation steps for a variety of problem classes [18]–[24]. To compute an approximate value function the designer specifies the regions of the state space that are of interest via the *state-relevance weighting* distribution, and then solves an optimization problem to find a linear combination of basis functions that maximizes the integral with respect to this weighting. An open question for the LP approach is how to choose the state-relevance weighting distribution so that the resulting approximate value function provides the tightest lower bound on the optimal value function, and hence the tightest sub-optimality bound for any policy.

The LP approach was first proposed for finite state and input spaces in [18], and equipped with theoretical guarantees in [19]. The authors of [19] also provide a discussion on the importance and difficulty of choosing the state-relevance weighting to give the lower bound and policy performance. An iterated version of the Bellman inequality was proposed in [20] and used to compute tighter lower bounds, however, the topic of choosing the state-relevance weighting is not addressed. The subsequent works [25], [26] avoid the need for a state-relevance weighting by focusing on the design of policies. In both of these works the authors propose using the iterated Bellman inequality as a constraint in the policy and choose the state-relevance weighting as a Dirac function located at the current state measurement. Thus the lower bound, which results from solving the policy, is expected to be the tightest in the vicinity of the state measurement and may not generalize well to other regions of the state space. Moreover, the approximate value functions computed from previous solves of the policy are discarded when solv-

ing the policy for the current state measurement. In [21] the authors use sum-of-squares programming techniques to compute high-order polynomial approximate value functions using the iterated Bellman inequality. The use of high-order polynomials would reduce the difficulty of choosing the state-relevance weighting, however, the optimization problem to solve becomes formidable.

Given a family of lower bounding approximate value functions, computed via the LP approach to ADP, taking a point-wise maximum over the family will yield the same or better approximation of the value function. The benefit of a point-wise maximum combination is empirically demonstrated in [20] for a simple example, with the set of state-relevance weighting parameters hand-picked using problem-specific insight. In our previous work [27], we proposed a problem formulation with the point-wise maximum combination used in the Bellman inequality. The formulation was used to develop an iterative algorithm for computing lower bounding approximate value functions, however, the quality of the approximation, comparable with that of [20], still relies on the designer choosing a sequence of state-relevance weightings. The algorithm proposed in [28] also uses the point-wise maximum combination in the Bellman inequality, and the authors propose an algorithm that computes the sequence of state-relevance weightings based on simulating the evolution of the system in a so-called *forward pass*. They consider a finite horizon setting and it is not clear how to extend the algorithm to an infinite horizon setting. A variety of other ADP algorithms compute lower bounds using theoretical tools different from the Bellman inequality, for example [29]–[31], each with its advantages and disadvantages, and none of which are similar to the algorithms we propose.

C. Contributions and Outline

In this paper, we propose a formulation that explicitly aims to maximize a point-wise maximum combination of multiple lower bounding approximate value functions, integrated with respect to the state-relevance weighting, which we refer to throughout as the *point-wise maximum objective*. Using this point-wise maximum objective leads to a non-convex optimization problem, in contrast to the existing methods discussed above that all maximize an objective function that is linear in the coefficients of the approximate value function. The motivation for the point-wise maximum objective is that a single choice of state-relevance weighting can be used to compute the entire family of approximate value functions. To see this consider that an approximate value function only contributes to the point-wise maximum objective if in some region of the state space it is greater than all the other functions in the family. This is in contrast to existing methods that require the designer to choose a separate weighting for computing each member of the family.

We propose using a gradient ascent algorithm to address the non-convex point-wise maximum objective, and combine this with the algorithm proposed in [27] for computing a family of approximate value function whose point-wise maximum combination satisfies the Bellman inequality. The benefits of gradient ascent in this setting are two fold:

1) At each iteration of the gradient ascent algorithm the objective function is linear in the coefficients of the approximate value function and hence the computation requirements are comparable with existing methods;
2) The computation of a gradient direction has the interpretation of reducing the support of the state-relevance weighting distribution to a region of the state space that is relevant for the current iteration. Thus the algorithm only requires selecting a single state-relevance weighting that places mass over the whole state space.

We refer to our proposed approach as the *accelerated* point-wise maximum approach to ADP because, compared to [27], on the numerical examples considered it requires comparable computation times, lower state-relevance weighting selection effort, and computes tighter lower bounds. In summary, the contributions of the paper are:

- We introduce the infinite dimensional point-wise maximum formulation of DP and prove under standard assumptions for SOC that it is equivalent to the infinite dimensional linear programming formulation of DP.
- To address the non-convex point-wise maximum objective, we propose a gradient ascent algorithm for finding approximate solutions, and we prove convergence properties of the algorithm.

The remainder of the paper is structured as follows. Section II presents the infinite dimensional point-wise maximum DP formulation. Section III introduces the approximation methods and our proposed algorithms. Section IV provides numerical results to demonstrate the sub-optimality bounds achieved and computation time required.

II. Dynamic Programming (DP) Formulation

A. Stochastic Optimal Control Formulation and Assumptions

We consider discrete time, infinite horizon, discounted cost, stochastic optimal control problems over continuous state and action spaces. The state of the system at time t is denoted by $x_t \in \mathcal{X} \subseteq \mathbb{R}^{n_x}$. The system state is influenced by the control decisions $u_t \in \mathcal{U} \subseteq \mathbb{R}^{n_u}$, and by the stochastic exogenous disturbance $\xi_t \in \Xi \subseteq \mathbb{R}^{n_\xi}$. In this setting, the states evolves according to the function $g : \mathcal{X} \times \mathcal{U} \times \Xi \to \mathcal{X}$ as $x_{t+1} = g(x_t, u_t, \xi_t)$, incurring the stage cost $\gamma^t l(x_t, u_t)$ at each time step, where $\gamma \in [0, 1)$ is the discount factor, and γ^t means exponentiation by the discrete time step t. By Π we denote the set of all feasible deterministic Markov policies, defined as $\{\pi(\cdot) | \pi(x) \in \mathcal{U}, \forall x \in \mathcal{X}\}$. The goal is to find a policy $x_t \mapsto \pi(x_t)$ that minimizes the cumulative cost over an infinite horizon, with initial condition $x_0 \in \mathcal{X}$,

$$V^*(x_0) := \inf_{\pi \in \Pi} \mathbb{E}\left[\sum\nolimits_{t=0}^{\infty} \gamma^t l(x_t, \pi(x_t))\right] \tag{1}$$

where $x_{t+1} = g(x_t, \pi(x_t), \xi_t)$ for all $t \geq 0$, which together with $\pi \in \Pi$ ensures that $x_t \in \mathcal{X}$ and $\pi(x_t) \in \mathcal{U}$ for all $t \geq 0$. The function $V^* : \mathcal{X} \to \mathbb{R}$ is the value function that represents the optimal cost-to-go from any state of the system if the optimal policy is played. The optimal policy π^* may require an infinite dimensional parameterization, hence problem (1) is intractable in general. Approximate policies can be categorized into four meta-classes: policy-, cost-, or value- function

approximations, and direct lookahead; see [32] and references therein for details of these policy meta-classes. The method presented in this paper approximates V^* and hence falls into the value function approximation category. As some direct lookahead policies utilise an approximation of V^*, the method presented in this paper can potentially be used to improve such policies, for example [33], [34], [35, Ch. 5].

To ensure that the problem is well posed we work in that same setting as [5, §6.3], specifically under [5, Assumption 4.2.1(a)] that the stage cost is lower semi-continuous, non-negative, and inf-compact, and also under [5, Assumptions 4.2.1(b), 4.2.2]. The assumptions ensure the value function is finite for all $x \in \mathcal{X}$ and that from the class of time-varying stochastic policies, the minimum is attained by a stationary deterministic policy, see [5, Theorem 4.2.3]. Finally, $\mathcal{F}(\mathcal{X} \times \mathcal{U})$ and $\mathcal{F}(\mathcal{X})$ are defined as the vector spaces of bounded, real-valued, Borel-measurable functions on $\mathcal{X} \times \mathcal{U}$ and $\mathcal{X}$ respectively, where [5, Definition 6.3.2, 6.3.4] provides the definitions of boundedness.

B. Linear Programming (LP) Formulation of DP

Solving the stochastic optimal control problem is equivalent to finding V^* as the solution of the Bellman equation [3],

$$V^*(x) = \underbrace{\inf_{u \in \mathcal{U}} \overbrace{\left\{ l(x,u) + \gamma \, \mathbb{E}\left[V^*\left(g(x,u,\xi)\right)\right] \right\}}^{(\mathcal{T}_u V^*)(x,u)}}_{(\mathcal{T} V^*)(x)}, \ \forall \, x \in \mathcal{X} \, . \tag{2}$$

$\mathcal{T}$ is known as the Bellman operator, and the $\mathcal{T}_u$ operator represents the cost of making decision u now and then playing optimally from the next time step forward. The optimal policy can be defined using V^* by,

$$\pi^*(x) \ = \ \left\{ \arg\min_{u \in \mathcal{U}} \ l(x,u) + \gamma \, \mathbb{E}\left[V^*\left(g(x,u,\xi)\right)\right] \right\} . \tag{3}$$

The existence of a V^* and π^* that are Borel measurable and attain the infimum is ensured by [5, Assumptions 4.2.1(a), 4.2.1(b), 4.2.2]. If $\nu(\cdot)$ is a finite measure on $\mathcal{X}$ that assigns positive mass to all open subsets of $\mathcal{X}$, then it can be shown that the solutions of the following linear program,

$$\max_{V} \ \int_{\mathcal{X}} V(x) \, \nu(\mathrm{d}x) \tag{4a}$$

$$\text{s.t. } V \in \mathcal{F}(\mathcal{X}) \tag{4b}$$

$$V(x) \ \leq \ \left(\mathcal{T}_u V\right)(x,u) \, , \quad \forall \, x \in \mathcal{X}, \, u \in \mathcal{U} \tag{4c}$$

satisfy (2) for ν-almost all (ν-a.a.) $x \in \mathcal{X}$, see [5, §6.3]. Constraint (4c) is referred to as the *Bellman Inequality*. A key feature of the LP formulation is that any choice of $\nu(\cdot)$ that places mass over the whole state space $\mathcal{X}$ leads (4) to recover a solution of the stochastic optimal control problem.

C. Point-wise Maximum Formulation of DP

Following [27], we introduce additional decision variables and use a point-wise maximum of value functions in the objective and the Bellman inequality constraint. It is seemingly redundant to add additional decision variables because (4) optimizes over $\mathcal{F}(\mathcal{X})$ as the space of bounded, real-valued, Borel-measurable functions, and hence it already recovers the optimal value function. However, the reason for considering a point-wise maximum formulation becomes clear in Section III when we consider approximating the optimal value function by restricting the space of decision variables to subspaces of $\mathcal{F}(\mathcal{X})$. As demonstrated in [27], the use of a point-wise maximum of value functions in the Bellman inequality constraint can lead to a tighter approximation in less computation time. The *point-wise maximum formulation* is thus stated as the following infinite dimensional optimization problem,

$$\max_{V_1,\ldots,V_J} \ \int_{\mathcal{X}} V_{\mathrm{PWM}}(x) \, \nu(\mathrm{d}x) \tag{5a}$$

$$\text{s.t. } V_j \in \mathcal{F}(\mathcal{X}) \, , \quad j = 1, \ldots, J \tag{5b}$$

$$V_{\mathrm{PWM}}(x) \leq \left(\mathcal{T}_u V_{\mathrm{PWM}}\right)(x,u), \ \forall \, x \in \mathcal{X}, \, u \in \mathcal{U} \tag{5c}$$

$$V_{\mathrm{PWM}}(x) = \max_{j=1,\ldots,J} V_j(x), \qquad \forall \, x \in \mathcal{X} \tag{5d}$$

where $J \in \mathbb{N}$ specifies the number of value function decision variables. The Bellman inequality constraint (5c) implies that feasible decisions for (5) will be point-wise under-estimators of V^*, and thus it is natural to combine a family of feasible but sub-optimal decisions using a point-wise maximum. Hence the motivation for using the objective (5a) is that, when optimizing over a subspace of $\mathcal{F}(\mathcal{X})$, it assess which combination of functions provides the tightest approximation with respect to the chosen measure $\nu(\cdot)$. We reiterate that the key difference from [27] is the use of the point-wise maximum V_{PWM} in the objective (5a), and that this is central to the algorithms proposed in Section III. The following lemma establishes some important properties of (5).

Lemma 2.1: Problems (4) and (5) are equivalent in the sense that there exist mappings between the feasible solutions and the optimal solutions of the two problems. Moreover, objective (5a) is jointly convex in the decision variables V_j, $j = 1, \ldots, J$.

Proof: Under the assumptions and definitions of Section II-A, one can easily see there is a mapping between feasible solutions since the space $\mathcal{F}(\mathcal{X})$ is closed under the maximum operation, i.e., $V_j = V$ for all $x \in \mathcal{X}$, $j = 1, \ldots, J$ in one direction, and $V = V_{\mathrm{PWM}}$ for all $x \in \mathcal{X}$ in the other direction. This gives equivalent objective value by construction, and thus $\int V^* d\nu$ is the optimal value for both (4) and (5).

The function V_{PWM} is convex in V_j, $j = 1, \ldots, J$ by definition of the max function over a finite number of elements. Thus (5a) is convex as integration is a linear operation, see for example [36, Lemma 2.1]. ■

Computing a solution of problem (5) poses the following difficulties

(D1) $\mathcal{F}(\mathcal{X})$ is an infinite dimensional space;
(D2) Objective (5a) involves a multidimensional integral over $\mathcal{X}$;
(D3) The $\mathcal{T}_u$-operator involves a multidimensional integral over Ξ;
(D4) Constraint (5c) involves an infinite number of constraints;
(D5) Constraint (5c) is non-convex in the decision variables;
(D6) The objective (5a) involves the maximization of a convex function;

Difficulties (D1-D4) apply also to problem (4) and a variety of approaches have been proposed to address them, see for example [23], [37]–[40]. In Section III we take inspiration from previous approaches to propose an approximation algorithm that additionally overcomes difficulties (D5-D6).

III. Approximate Dynamic Programming (ADP)

This section proposes an algorithm for computing an approximate value function that is feasible for problem (5) at every iteration and analyzes its convergence.

A. Approaches Adopted for Difficulties (D1), (D3) and (D4)

To overcome difficulty (D1), as suggested in [17], we restrict the value functions candidates to the span of a finite family of Borel-measurable basis functions $\phi_k : \mathcal{X} \to \mathbb{R}$, $k = 1, \dots, K$. We parameterize the restricted function space as,

$$\hat{\mathcal{F}}(\mathcal{X}) = \left\{ \alpha^\intercal \phi(x) \middle| \alpha \in \mathbb{R}^K \right\}, \quad \text{with } \phi(x) = \begin{bmatrix} \phi_1(x) \\ \vdots \\ \phi_K(x) \end{bmatrix}. \tag{6}$$

The benefit of this parameterization is that it is linear in the α parameter. Each approximate value function $\hat{V}_j$ is parameterized by its own vector that we denote α_j, i.e., $\hat{V}_j(x) = \alpha_j^\intercal \phi(x)$ for all $x \in \mathcal{X}$. For the numerical examples in Section IV we use the space of polynomial functions up to a certain degree by choosing the ϕ_k to be each of the monomials up to that degree.

To overcome difficulty (D3) we first use Jensen's inequality to switch the order of expectation and maximisation in the $\mathcal{T}_u V_{\mathrm{PWM}}$ term, thus providing a sufficient condition for constraint (5c). We then require that for each basis function $\mathbb{E}\left[\phi\left(g(x,u,\xi)\right)\right]$ has an analytic expression. In the case of polynomial basis functions and polynomial dynamics, this requires knowledge of the moments of the distribution of ξ up to the maximum degree of ξ in $\phi\left(g(x,u,\xi)\right)$. If the required moments are not analytically available, then the Monte Carlo sampling can be used to approximate them, and, as the distribution is stationary, this only needs to be computed once.

To overcome difficulty (D4), a variety of convex sufficient conditions techniques are proposed in the literature for approximating (5c) with a finite number of constraints, for example [20]–[22]. The applicable reformulation depends on the problem data and basis functions, and the algorithm we propose in the sequel applies for all such convex inner approximations. When all problem data is polynomial and polynomial basis functions are used, then constraint (5c) can be inner approximated using the sum-of-squares (SOS) S-procedure [21], [22]. A reformulation of the Bellman inequality to compute piecewise constant lower bounds of the optimal value function is given in [23], [24].

B. Proposed Approach for Difficulties (D2), (D5) and (D6)

The inclusion of the point-wise maximum value function in the objective (5a) is pivotal in the algorithm we propose, however, it precludes using previous approaches for evaluating the integral in the objective. To overcome difficulty (D2) we replace ν by a finitely supported distribution denoted c. Specifically, we choose c as a finite sum of N Dirac pulses located at $\{z_i\}_{i=1}^N \subset \mathcal{X}$. This violates the hypothesis for equivalence between (2) and (4), but reduces the multidimensional integral in (5a) to a sum over the locations of the Dirac pulses.

For clarity of presentation, we consider now an auxiliary problem that highlights our proposed approach for overcoming difficulties (D5) and (D6). We consider a family of functions defined by a finite set $\mathcal{A} \subset \mathbb{R}^K$, parameterized by $\bar{\alpha} \in \mathcal{A}$, and used in both a point-wise maximum objective and a point-wise maximum constraint. We then define,

$$\bar{V}(x) = \max_{\bar{\alpha} \in \mathcal{A}} \ \bar{\alpha}^\intercal \phi(x), \quad \forall x \in \mathcal{X}, \tag{7a}$$

and assume that $\mathcal{A}$ has been selected such that $\bar{V} \leq V^*$ for all $x \in \mathcal{X}$. Additionally, we introduce $f_{\mathrm{pwm}} : \mathbb{R}^K \to \mathbb{R}$ as the point-wise maximum objective function when adding an additional function $\alpha^\intercal \phi(x) \in \hat{\mathcal{F}}(\mathcal{X})$ to the function $\bar{V}$, i.e.,

$$f_{\mathrm{pwm}}(\alpha) = \frac{1}{N} \sum_{i=1}^{N} \max\left\{ \alpha^\intercal \phi(z_i), \bar{V}(z_i) \right\}, \tag{8}$$

where z_i are the points selected to overcome difficulty (D2). The auxiliary problem for maximizing f_{pwm} in the presence of $\bar{V}$ is,

$$\max_{\alpha \in \mathbb{R}^K} \ f_{\mathrm{pwm}}(\alpha) \tag{9a}$$

$$\text{s.t. } \alpha^\intercal \phi(x) \leq \left(\mathcal{T}_u \bar{V}\right)(x,u), \quad \forall x \in \mathcal{X}, \, u \in \mathcal{U}. \tag{9b}$$

With $\bar{V}$ as fixed parameters in (9) the constraint is convex in the decision variable α, thus overcoming difficulty (D5). Moreover, if $\bar{V}$ satisfies the Bellman inequality, then (9b) implies that $\max\left\{\alpha^\intercal \phi, \bar{V}\right\}$ also satisfies the Bellman inequality. The steps to show convexity of constraint (9b), first presented in [27], are provided in Appendix A for completeness. In Section III-D we present the proposed algorithm for iteratively adding elements to $\mathcal{A}$ in a greedy fashion, where the difference compared to [27] is the choice of objective weighting. To simplify the presentation, we introduce the notation,

$$\alpha \in \mathcal{S}(\mathcal{A}) \subseteq \mathbb{R}^K \quad \Rightarrow \quad \alpha \text{ satisfies (9b)}, \tag{10}$$

to represent the convex sufficient condition for the Bellman inequality constraint (9b). The algorithm we propose is applicable for the convex sufficient conditions discussed in Section III-A for overcoming difficulty (D4). Methods that use sampling to address the infinite number of constraints in (9b), for example [38], [39], [41], [42], are not applicable because they do not guarantee that (9b) is satisfied.

Despite the convexified constraint, (9) is still a non-convex problem due to (D6). In general, problem (9) will have multiple distinct local maxima and stationary points. The convexity of the objective means that given an element of the sub-differential, constructed at a particular point in the decision variable space α, it parameterizes a hyperplane that is a point-wise lower-bound on the objective function. Thus we propose to iteratively maximize along sub-gradient directions to overcome difficulty (D6), and in Section III-C we introduce the algorithm and its convergence properties.

C. First-order method for the point-wise maximum objective

We propose Algorithm 1 to improve f_{pwm} from a given feasible initial condition $\alpha^{(0)}$ using only first-order information of the objective function. The objective f_{pwm} is in general non-smooth as it is a maximum of functions, thus in line 4 we use the upper sub-differential for selecting gradient ascent directions, denoted as $\partial^+ f_{\text{pwm}}$ and defined in Appendix B-A; an upper sub-differential is considered because (9) is a maximization problem. Given a non-zero element from the upper sub-differential, i.e., an upper sub-gradient, in line 8 we update the decision variable by maximizing along the sub-gradient direction within the feasible region. The algorithm terminates when the change in objective value between two subsequent iterations is less than a pre-specified tolerance.

To compute an element from the upper sub-differential of f_{pwm}, to be used in line 4 of Algorithm 1, we introduce the following assumption on the basis functions.

Assumption 3.1: The basis functions in the set $\{\phi_k\}_{k=1}^K$ are finite for all $x \in \mathcal{X}$ and include the constant function.

Without loss of generality we take $\phi_1(x) = 1$ for all $x \in \mathcal{X}$. The finite part of Assumption 3.1 is required to ensure that the objective function (9a), $f_{\text{pwm}}(\alpha)$, is a proper function in the sense required for the necessary optimality condition used for Theorem 3.3, i.e., that $f_{\text{pwm}}(\alpha) > -\infty$ for at least one $\alpha \in \mathbb{R}^K$, and $f_{\text{pwm}}(\alpha) < +\infty$ for all $\alpha \in \mathbb{R}^K$. For a general choice of basis functions ϕ_k it is difficult to characterize the upper sub-differential set at non-smooth points. Instead, we work with a particular element from the upper sub-differential of f_{pwm} that is readily computable at α under Assumption 3.1,

$$\partial^+ f_{\text{pwm}}(\alpha) \ni \frac{1}{N} \sum_{i=1}^{N} \begin{cases} \phi(z_i) & \text{if } \alpha^\intercal \phi(z_i) \geq \bar{V}(z_i) \\ 0 & \text{if } \alpha^\intercal \phi(z_i) < \bar{V}(z_i) \end{cases}, \quad (11)$$

where the term inside the sum is an element from the upper sub-differential of $\max\{\alpha^\intercal \phi(x), \bar{V}(x)\}$. Given the element $\bar{d} \in \partial^+ f_{\text{pwm}}(\bar{\alpha})$ computed as per (11) at a point $\bar{\alpha} \in \mathbb{R}^K$, the hyperplane $(\alpha - \bar{\alpha})^\intercal \bar{d} + f(\bar{\alpha})$ is a supporting hyperplane of the convex function f_{pwm}. However, we note that for maximization of a convex function not all supporting hyperplanes are in the upper sub-differential. In the proof of Theorem 3.3 we show that (11) is indeed an element of the upper sub-differential.

Algorithm 1 can be seen as a method that iteratively adjusts the objective of line 8 along sub-gradient directions of problem (9). To ensure that an element from the argmax can always be computed in line 8, we introduce the following assumption on the choice of basis functions and inner approximation set.

Assumption 3.2: The basis function set $\{\phi_k\}_{k=1}^K$, Bellman inequality inner approximation set $\mathcal{S}(\mathcal{A})$, and problem data are such that the following optimization problem,

$$\max_{\alpha \in \mathbb{R}^K} \{ \alpha^\intercal \phi(z_i) \; ; \; \alpha \in \mathcal{S}(\mathcal{A}) \} \quad (12)$$

attains its maximum for all $\{z_i\}_{i=1}^N$.

Intuitively speaking Assumption 3.2 ensures a finite $\alpha^{(k+1)}$ for any $d^{(k)}$ because the objectives of (12) and Algorithm 1 line 8 are linear in α, and $d^{(k)}$ is a linear combination of the $\phi(z_i)$ vectors. Thus each solution of (12) defines a supporting hyperplane for the constraint set $\alpha \in \mathcal{S}(\mathcal{A})$, and this is used in the proof of Theorem 3.3 to show that a relaxation of line 8 attains its maximum and that this implies the attainment of line 8. This assumption is not overly restrictive as it can be ensured for any general problem instance by placing an upper bound on a norm of α, see [43] for example. For a particular problem instance the assumption can be verified by, for example, showing the existence of a strictly feasible point in the dual of (12) [44, Theorem 3.1] [45, Corollary 30.5.2].

Theorem 3.3: Under Assumptions 3.1 and 3.2, for any initial condition $\alpha^{(0)} \in \mathcal{S}(\mathcal{A})$ and any $\epsilon > 0$, Algorithm 1 generates a non-decreasing sequence $f_{\text{pwm}}(\alpha^{(k)})$ and terminates after a finite number of iterations. With $\epsilon = 0$, the sequence $f_{\text{pwm}}(\alpha^{(k)})$, converges to a finite value, and the sequences $\alpha^{(k)}$, $d^{(k)}$, satisfy the following necessary optimality condition for $\max_{\alpha \in \mathbb{R}^K} \{f_{\text{pwm}}(\alpha); \alpha \in \mathcal{S}(\mathcal{A})\}$ in the limit,

$$\lim_{k \to \infty} \left(\max_{\alpha \in \mathcal{S}(\mathcal{A})} \left(\alpha - \alpha^{(k)} \right)^\intercal d^{(k)} \right) = 0 . \quad (13)$$

Proof: see Appendix B-D. ∎

Theorem 3.3 guarantees that if the initial condition $\alpha^{(0)}$ strictly improves on $f_{\text{pwm}}(0)$, then $f_{\text{pwm}}(\alpha^{(k)})$ returned also strictly improves on $f_{\text{pwm}}(0)$. The convergence in finite iterations ensures that the algorithm is practical to implement, and the limiting behaviour suggests that the $\alpha^{(k)}$ returned could be close to a local maxima. Although Theorem 3.3 provides no insight into the rate of convergence, the numerical examples in Section IV demonstrate that significant improvement in f_{pwm} can be achieved with only a handful of iterations.

Algorithm 1 is a so-called *Minorize Maximize* algorithm for maximizing a convex function, and we now contrast with generic algorithms that exist in the literature for this same purpose. In the case where f_{pwm} is differentiable, then line 4 of Algorithm 1 becomes $d^{(k)} \leftarrow \nabla f_{\text{pwm}}(\alpha^{(k)})$ and is a special-case of the so-called *convex-concave procedure* introduced in [46], and for which convergence guarantees are given in [47, Theorem 4]. Algorithms applicable for non-smooth problems like (9) are presented together with convergence guarantees in [48, Theorem 3] and [49, Proposition 1]. Applying the algorithm from [48] or [49] to problem (9) would require using the lower sub-differential of f_{pwm} in line 4 of Algorithm 1. For a non-smooth convex function the lower sub-differential contains the upper sub-differential, thus allowing more flexibility on line 4 of Algorithm 1. However, [48] and [49] use the lower sub-differential also for defining necessary optimality conditions. This means that, compared to Algorithm 1, the algorithms from [48] and [49] may have additional points in their convergence set that are not local maxima of the non-smooth convex maximization problem.

One could consider solving problem (9) directly using opimization software appropriate for the non-convexity. For example, objective (9a) can be exactly reformulated by introducing a binary decision variable for the N points in the summation. For a problem with linear dynamics and quadratic stage cost, such a reformulation leads to a convex mixed-integer semi-definite program (MISDP) for which optimization software exists or can be adapted, for example [50]–[52].

Algorithm 1 Find points satisfying necessary optimality conditions of problem (9) with c as a sum of Dirac pulses

1: **procedure** INNERPROBLEM($\alpha^{(0)}$, $\mathcal{A}$, ϵ)
2: $k \leftarrow 0$
3: **repeat**
4: $d^{(k)} \leftarrow$ an element from $\partial^+ f_{\text{pwm}}\left(\alpha^{(k)}\right)$
5: **if** $\left(d^{(k)} = 0\right)$ **then**
6: $\alpha^{(k+1)} \leftarrow \alpha^{(k)}$
7: **else**
8: $\alpha^{(k+1)} \leftarrow \arg\max \left\{\alpha^\intercal\, d^{(k)}\,;\, \alpha \in \mathcal{S}(\mathcal{A})\right\}$
9: **end if**
10: $k \leftarrow k+1$
11: **until** $\left(f_{\text{pwm}}\left(\alpha^{(k)}\right) - f_{\text{pwm}}\left(\alpha^{(k-1)}\right)\right) < \epsilon.$
12: **return** $\alpha^{(k)}$
13: **end procedure**

This reformulation requires N binary the decision variables, and our testing indicates that, for the problems considered, $N \geq \mathcal{O}(10^5)$ is required to achieve a reasonable approximation quality. We do not consider such a reformulation in this work because the computation times are prohibitive.

D. Point-wise Maximum ADP Algorithm

In this section we propose Algorithm 2, which iteratively updates the value function estimate used in the objective and constraints of problem (9), i.e., $\bar{V}$. At each iteration of lines 7–11, a candidate approximate value function $\alpha^{(0)}$ is generated by solving (12) with z_i as one of the Dirac pulse locations from $c(\cdot)$. Algorithm 1 refines this candidate before it is added to the collection $\mathcal{A}$. This process of generating, refining, and adding is repeated for all z_i, $i = 1, \ldots, N$. The algorithm terminates when the improvement in $\bar{V}$ is below some pre-specified threshold. The following theorem formalises the convergence properties of Algorithm 2.

Theorem 3.4: For any set $\mathcal{A}$ such that $\bar{V}$ is a point-wise under-estimator of V^*, and for any ϵ_{IN}, $\epsilon_{\text{OUT}} > 0$, Algorithm 2 terminates after a finite number of iterations.

Proof: By Theorem 3.3 we have that line 9 of Algorithm 2 terminates after finite iterations for all $\epsilon_{\text{IN}} > 0$. The sequence $f^{(m)}$ is non-decreasing by definition as a point-wise maximum of functions and because elements are never removed from $\mathcal{A}$. The same reasoning as Appendix B-D establishes that $\max_{\{\bar{\alpha}\in\mathcal{A}\}} \bar{\alpha}^\intercal \phi(z_i)$ is bounded above for all $\{z_i\}_{i=1}^N$ at all iterations of Algorithm 2. Hence $f^{(m)}$ is bounded above and is thus a convergent sequence. Therefore, for all $\epsilon_{\text{OUT}} > 0$ there exists an $m \geq 1$ such that the condition on line 14 triggers. ■

Convergence of Algorithm 2 is guaranteed even without the refinement steps of Algorithm 1. However, the numerical results in Section IV show that convergence tends to be much slower and that significant improvements are achieved with only a few iterations of Algorithm 1. For this reason we refer to the combination of Algorithms 1 and 2 as the *accelerated* point-wise maximum approach to ADP, and in the numerical examples we highlight the acceleration achieved by the refinement steps of Algorithm 1.

Algorithm 2 Maximise the value of $\int \bar{V} dc$

1: **procedure** OUTERPROBLEM
2: **Select** $\mathcal{A}$, $\{z_i\}_{i=1}^N$ according to §III-F
3: **Select** ϵ_{IN}, $\epsilon_{\text{OUT}} < 0$
4: $m \leftarrow 0$
5: **repeat**
6: $f^{(m)} \leftarrow \frac{1}{N}\sum_{i=1}^N \left(\max_{\bar{\alpha}\in\mathcal{A}} \bar{\alpha}^\intercal \phi(z_i)\right)$
7: **for all** $\{z_i\}_{i=1}^N$ **do**
8: $\alpha^{(0)} \leftarrow \arg\max \{\alpha^\intercal \phi(z_i); \alpha \in \mathcal{S}(\mathcal{A})\}$
9: $\hat{\alpha} \leftarrow$ INNERPROBLEM$(\alpha^{(0)}, \mathcal{A}, \epsilon_{\text{IN}})$
10: $\mathcal{A} \leftarrow \hat{\alpha} \cup \mathcal{A}$
11: **end for**
12: $m \leftarrow m+1$
13: $f^{(m)} \leftarrow \frac{1}{N}\sum_{i=1}^N \left(\max_{\bar{\alpha}\in\mathcal{A}} \bar{\alpha}^\intercal \phi(z_i)\right)$
14: **until** $\left(f^{(m)} - f^{(m-1)}\right) < \epsilon_{\text{OUT}}$,
15: **return** $\mathcal{A}$
16: **end procedure**

The objective $\alpha^\intercal \phi(z_i)$ in line 8 of Algorithm 2 is chosen so that the $\alpha^{(0)}$ passed to Algorithm 1 has a non-zero sub-gradient $d^{(0)}$ (line 4 of Algorithm 1). To see this, note that the sub-gradient in (11) is non-zero if $\alpha^\intercal \phi(z_i)$ weakly dominates $\bar{V}(z_i)$ for at least one $i = 1, \ldots, N$. Thus, by Assumption 3.2, line 8 of Algorithm 2 computes an $\alpha^{(0)}$ that weakly dominates $\bar{V}$ at the chosen point z_i if such a solution exists in the feasible set $\alpha \in \mathcal{S}(\mathcal{A})$. Different objectives for line 8 of Algorithm 2 can be considered and still enjoy the convergence guarantee of Theorem 3.4. However, this would introduce a tuning parameter and empirical testing has shown no benefit when hand-tuning the objective.

E. Comparison with existing methods

As described in Section III-B, the approach for reformulating the Bellman inequality to overcome difficulty (D5) is inspired by our previous work [27], and in that paper we provide a comprehensive discussion for how it relates to [20], [26]. In [25] the authors propose the so-called *min-max* ADP policy, which, for a given state measurement x, is stated as,

$$\arg\min_{u\in\mathcal{U}} \left\{ \sup_{\alpha\in\mathbb{R}^K} l(x,u) + \gamma\, \mathbb{E}\left[\alpha^\intercal \phi(g(x,u,\xi))\right] \right\}, \qquad (14)$$

subject to the constraint on α that $\alpha^\intercal \phi$ satisfies the iterated Bellman inequality, i.e., $\alpha^\intercal \phi(x) \leq \mathcal{T}^M \alpha^\intercal \phi(x)$ for all $x \in \mathcal{X}$, with $M \in \mathbb{N}$ specifying the number of Bellman inequality iterations. Although this policy encodes a supremum over all approximate value functions that satisfy the iterated Bellman inequality, the policy uses only a single approximate value function, $\alpha^\intercal \phi$, for each x and u. In contrast, the approach we propose uses a point-wise maximum of multiple approximate value functions, i.e., the term $\alpha^\intercal \phi(g(x,u,\xi))$ in (14) would be replaced by, $\max\{\alpha^\intercal \phi(g(x,u,\xi)), \bar{V}(g(x,u,\xi))\}$. We do not pursue this further because the remaining approximation steps from [25] use the fact that (14) is affine in α, whereas the point-wise maximum is generally not affine in α.

F. Discussion and extensions

As stated in Section II-B, the infinite dimensional optimization problems (4) and (5) compute the optimal value function, V^*, for any choice of the measure ν that assigns positive mass to all open subsets of the state space. If we fix a specific such ν, this defines $\int V^* d\nu$ as the optimal value of the SOC problem. For any choice of the finitely supported distribution c, and hence the samples $\{z_i\}_{i=1}^N$, the family of approximate value functions computed by Algorithm 2 provides the lower bound $\int \bar{V} d\nu$ on the optimal value. Thus the obvious choice for c is to draw samples from ν, however, sampling c in different ways may tighten the lower bound $\int \bar{V} d\nu$, and for this reason c is the *state-relevance weighting* distribution of our proposed method.

We refer to our propose method as *accelerated* because the designer needs only to choose the distribution c, and from that the objective function used at every iteration of Algorithm 1 line 8 is a sub-gradient and at every iteration of Algorithm 2 line 8 is a single sample on which c is supported. This is in contrast to the method from [27] where the designer is required to choose a separate objective function for the computation of every approximate value function added to the point-wise maximum family. We note that a linear objective function is used at every iteration of Algorithm 1, and that the method from [27] allows any linear objective function to be used at each iteration. Thus the algorithms we propose in this paper can be seen as a special case of the method from [27], tailored towards maximizing the point-wise maximum objective value. That this specific choice of objective function accelerates the computation of a tighter lower bound on the optimal value likely requires that f_{pwm} is a sufficiently good approximation of $\int \max\{\alpha^\intercal \phi, \bar{V}\} d\nu$.

If the goal is to optimize the on-line performance of the greedy policy, it is again likely that different samples for c lead to differing on-line performance. Motivated by the performance bounds provided in [19], a reasonable choice is to place Algorithm 2 inside another iteration that updates c as samples from the discounted occupancy measure for the current greedy policy, computed empirically by simulating the system evolution using Monte Carlo sampling.

For real-time applications where the greedy policy must be computed very fast, it is necessary that the cardinality of $\mathcal{A}$ is small, and perhaps even a singleton. For examples with linear dynamics, quadratic stage costs, polytopic spaces, and using the space of quadratics for $\hat{\mathcal{F}}(\mathcal{X})$, then the greedy policy is a Quadratically Constrained Quadratic Program (QCQP), with the number of quadratic constraints equal to the cardinality of $\mathcal{A}$. In such examples, a low cardinality of $\mathcal{A}$ has clear benefits from an on-line compution perspective. In these cases it is beneficial to run Algorithm 2 twice. First, Algorithm 2 is run for as long as practical to achieve a good under-estimate of V^*, with a simple initialization, for example $\mathcal{A} = \{0\}$. Second, an adaptation of Algorithm 2 is used for updating two families of fixed functions, one family for the point-wise maximum objective f_{pwm}, denoted $\mathcal{A}_{\mathrm{obj}}$, and a separate family for the sufficient reformulation of the point-wise constraint, denoted $\mathcal{A}_{\mathrm{con}}$ and hence the constraint is $\mathcal{S}(\mathcal{A}_{\mathrm{con}})$. In this adaptation, a simple initialization is used for $\mathcal{A}_{\mathrm{obj}}$, for example $\mathcal{A}_{\mathrm{obj}} = \{0\}$. By contrast $\mathcal{A}_{\mathrm{con}}$ is initialised with the result from the first run of Algorithm 2, which could be a family of hundreds or thousands of functions. This adaptation adds the new element $\hat{\alpha}$ to both families, i.e., line 10 of Algorithm 2 becomes $\mathcal{A}_{\mathrm{obj}} \leftarrow \hat{\alpha} \cup \mathcal{A}_{\mathrm{obj}}$, $\mathcal{A}_{\mathrm{con}} \leftarrow \hat{\alpha} \cup \mathcal{A}_{\mathrm{con}}$. The adaptation of Algorithm 2 thus runs until the cardinality of $\mathcal{A}_{\mathrm{obj}}$ reaches that desired for the online policy. The benefit of this adaptation of is that the large family of fixed functions $\mathcal{A}_{\mathrm{con}}$ used in the constraint can allow for $\mathcal{A}_{\mathrm{obj}}$ to be a tighter approximation of V^*. This can in turn improve the online performance of the policy, as indicated by the performance bounds derived in [19], [35, Ch. 5], [53].

The approximate value function computed by Algorithm 2 can be used off-line to certify the empirical performance of alternative policies that do not use the approximate value function. In this case Algorithm 2 is run for as long as practical, then the chosen policy is simulated from a particular initial state, $\hat{x}$, for a time horizon such that γ^t has decayed sufficiently. The approximate value function evaluated at the initial state is a lower bound on $V^*(\hat{x})$ and thus provides a bound on the sub-optimality of the policy, and hence indicates the potential benefit of considering further alternatives.

In [27] the value function decision variable was also included in the right-hand-side of constraint (9b), i.e.,

$$\alpha^\intercal \phi(x) \leq \left(\mathcal{T}_u \left(\max\left\{\alpha^\intercal \phi(x), \bar{V}(x)\right\}\right)\right)(x,u)\,, \tag{15}$$

for all $x \in \mathcal{X}$ and $u \in \mathcal{U}$. This results in a bi-linear term in the constraint, and in that work the authors suggest gridding the multiplier of the bi-linear term. We do not consider this extension in the numerical examples because it adds significant computation time and empirically it provides little or no benefit for the examples considered.

Algorithm 1 can be extended to fit multiple new lower bounding functions at the same time. To exemplify, consider the case of adding two new lower bounding functions. The non-convex optimization problem then becomes,

$$\max_{\alpha,\beta \in \mathbb{R}^K} \ \frac{1}{N}\sum_{i=1}^N \max\left\{\alpha^\intercal \phi(z_i), \beta^\intercal \phi(z_i), \bar{V}(z_i)\right\} \tag{16a}$$

$$\text{s.t. } \alpha \in \mathcal{S}(\mathcal{A})\,, \quad \beta \in \mathcal{S}(\mathcal{A})\,. \tag{16b}$$

We construct an element from the upper subdifferential in a similar fashion,

$$\partial^+ \left(\max\left\{\alpha^\intercal \phi(x), \beta^\intercal \phi(x), \bar{V}(x)\right\}\right) = \begin{cases} \begin{bmatrix}\phi(x)^\intercal, 0\end{bmatrix}^\intercal & \text{if } \alpha^\intercal \phi(x) \geq \max\left\{\beta^\intercal \phi(x), \bar{V}(x)\right\} \\ \begin{bmatrix}0, \phi(x)^\intercal\end{bmatrix}^\intercal & \text{if } \beta^\intercal \phi(x) > \max\left\{\alpha^\intercal \phi(x), \bar{V}(x)\right\} \\ \begin{bmatrix}0, 0\end{bmatrix}^\intercal & \text{if } \bar{V}(x) > \max\left\{\alpha^\intercal \phi(x), \beta^\intercal \phi(x)\right\} \end{cases}. \tag{17}$$

As the constraints are separable, once the subdifferential element is computed, then Algorithm 1 line 8 can be solved in parallel for α and β, differing only in the objective vector.

IV. Numerical Examples

In the numerical examples we consider problems with linear dynamics, quadratic stage costs, hyper-cube constraints on the

input space, and use quadratics for the restricted function space $\hat{\mathcal{F}}(\mathcal{X})$. In Appendix C we provide the definition of $\hat{\mathcal{F}}(\mathcal{X})$, formulate line 8 of Algorithm 1 so that it can be passed to a standard solver, and verify that Assumption 3.1 holds. Assumption 3.2 was observed to hold empirically, in that the solver returned a finite, optimal solution at each iteration.

A. 10 Dimensional Linear-Quadratic Problems

We consider an input constrained linear-quadratic system, with dimension $n_x = 10$, $n_u = 3$, and $n_\xi = 10$. The system dynamics take the form $x_{t+1} = Ax_t + B_u u_t + B_\xi \xi_t$, where A, B_u, and B_ξ are matrices of compatible size, and the quadratic stage cost is $l(x,u) = x^\intercal I_{n_x} x + 0.5 u^\intercal I_{n_u} u$, where I_n denotes an identity matrix of size n, and we use discount factor $\gamma = 0.99$. The exogenous disturbance is distributed as $\xi_t \sim \mathcal{N}(0, 0.01 I_{n_\xi})$ with B_ξ as an identity matrix. The initial state is normally distributed as $x_0 \sim \mathcal{N}(0, 9 I_{n_x})$. The A and B_u matrices are randomly generated, with the A matrix scaled to be marginally stable, i.e., a spectral radius equal to 1. The lower and upper bounds for the input space $\mathcal{U}$ are chosen to make the constraints relevant for the whole horizon, while the state space $\mathcal{X}$ is unconstrained.

For this linear-quadratic setting, with polytopic input constraints, it is known that the optimal value function is convex and piece-wise quadratic [54, Theorem 6.7], hence we further restrict $\hat{\mathcal{F}}(\mathcal{X})$ to convex quadratics. We initialise $\mathcal{A}$ with the approach described in [19], i.e., we choose ρ as the initial state distribution and solve of the following,

$$\underset{\alpha \in \mathbb{R}^K}{\arg\max} \left\{ \int_{\mathcal{X}} \alpha^\intercal \phi(x)\, \rho(\mathrm{d}x) \;;\; \alpha \in \mathcal{S}(\{\alpha\}) \right\}. \tag{18}$$

In this setting (18) is a convex semi-definite program, the objective requires the first and second moments of ρ, and the constraint is reformulated as a linear matrix inequality with respect to α, see [20, §6]. The solution of (18) represents the lower bound proposed in [19].

To demonstrate the benefits of our proposed algorithms, we compare the following variants:

1) Algorithm 2 exactly as stated, which we refer to as *Algorithm 2 with refinement*.
2) Algorithm 2 with line 9 replaced by $\hat{\alpha} \leftarrow \alpha^{(0)}$, which we refer to as *Algorithm 2 without refinement*.

We use this terminology *with and without refinement* because the gradient steps of Algorithm 1, as executed by line 9 of Algorithm 2, essentially refine the initial guess $\alpha^{(0)}$ to an updated $\hat{\alpha}$ that has that same or greater value of the point-wise maximum objective, f_{pwm}, at that iteration. We choose c as $N = 10^6$ samples from the initial state distribution, i.e., from $\mathcal{N}(0, 9 I_{n_x})$, and we run the two variants described until $\mathcal{A}$ has 10^5 members. As N is greater than 10^5, this means that Algorithm 2 line 12 and after is never executed. We also tested with $N < 10^5$, but found that the sub-optimality bounds were significantly improved using a large N. We did not add more than 10^5 members to $\mathcal{A}$ because the computation time became prohibitive.

Figure 1 (a) shows the lower bound achieved by running Algorithm 2 with (red) and without (blue) refinement. The

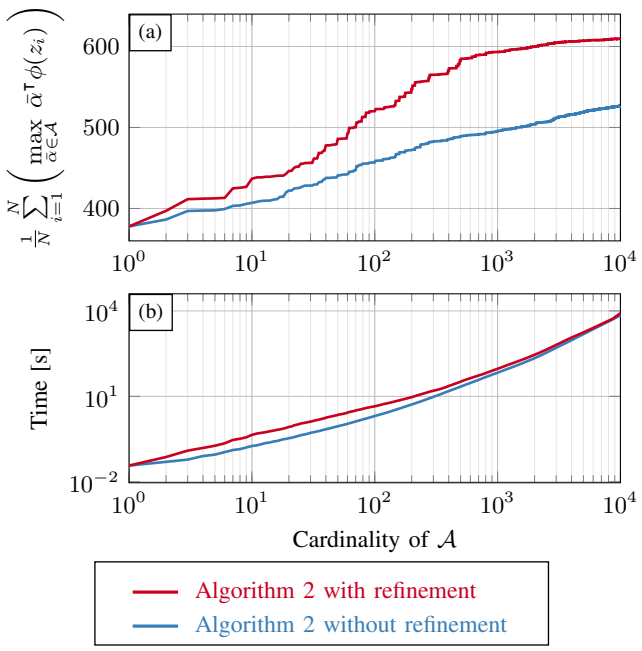

Fig. 1. Results for the two variants described in Section IV-A, i.e., Algorithm 2 with and without refinement. (a) the approximate value function $\bar{V}$ integrated with respect to the N samples. (b) the cumulative computation time, in seconds, for solving the optimization problems on Algorithm 2 line 8 and Algorithm 1 line 8, as performed on a 3.00Ghz Xeon processor.

key feature of the result is that, although Algorithm 2 without refinement is guaranteed to converge, the cardinality of $\mathcal{A}$ required to reach a reasonable lower bound is significant. Algorithm 2 with refinement, on the other hand, achieves a significantly better lower bound with over an order of magnitude lower cardinality of $\mathcal{A}$. The termination criteria on Algorithm 1 line 11 is selected as a relative tolerance of 0.1%. For the result in Figure 1 we observe a maximum of 4 solves of Algorithm 1 line 8 until the termination criteria triggers, i.e., the number of refinement steps performed. For the final 9000 members added to $\mathcal{A}$, only 1 or 0 solves of Algorithm 1 line 8 is required for convergence. Hence the computation times shown in Figure 1 (b) differ only slightly.

In summary, this result suggests that, compared to Algorithm 2 without refinement, Algorithm 2 with refinement provides an improved lower-bounding approximate value function with a moderate increase in computation time. Hence for the remainder of the numerical results we only consider Algorithm 2 with refinement.

B. Sub-optimality bounds

Comparing the lower bounds computed by various methods is meaningful if it significantly tightens the sub-optimality bound on the online performance for a particular policy. It requires an impractical amount of computation to simulate a policy from all $N = 10^6$ samples for sufficiently many disturbance realisations. Instead, we simulate a clipped LQR policy starting from 800 samples from c, each simulated for 10^3 time steps, and 10^3 extractions from the disturbance distribution used to evaluate the expectation. In Figure 2 we plot the percentage sub-optimality as certified by Algorithm 2

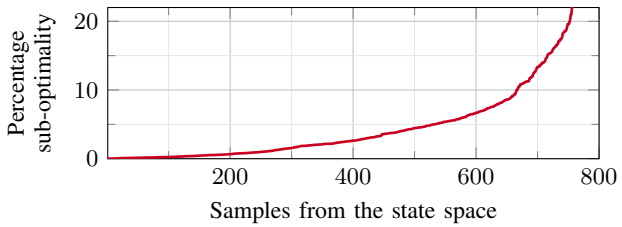

Fig. 2. Percentage sub-optimality bound of a clipped LQR policy for 800 randomly selected initial conditions, as certified by the lower-bound computed by Algorithm 2 with refinement. Averaged over the 800 initial conditions, the policy is certified to within 7.0% of the optimal.

TABLE I
COMPARING VARIOUS METHODS FOR CERTIFYING SUB-OPTIMALITY

Method used for compute the approximate value function	$\lvert\mathcal{A}\rvert$	Average % sub-optimality	Time [seconds]
Algorithm 2 with refinement	10^4	7.0	8338
	100	15.0	4.5
Hand-tuned sequence of $c(x)$ [27]	100	24.4	2.3
Iterated Bellman inequality [26]	100	53.7	2.8
Single Bellman inequality [19]	1	58.7	0.04

Note: column 3 is the bound on the percentage sub-optimality of the clipped LQR policy as certified by the respective method. Column 4 is the time required to compute $\mathcal{A}$, as performed on a 3.00Ghz Xeon processor.

with refinement for each of the initial conditions. As the initial conditions are drawn randomly, we make the figure more readable by ordering the samples such that the line is monotonically increasing. Averaging over the 800 initial condition samples, Algorithm 2 with refinement certifies this policy to be within 7.0% of the optimal. The figure also shows that the online performance of the clipped LQR policy is within 10% of the optimal for the majority of the initial conditions sampled, and up to to 300% sub-optimal (not shown for clarity) in the worst cases. This information could be used by the practitioner to guide an improved policy design by, for example, clustering the samples with similar sub-optimality and looking for patterns in the data.

C. Comparison with existing methods

We describe four alternative methods to compute a lower-bounding approximate value function, and in Table I we compare the average sub-optimality bound over the same 800 samples from Section IV-B. The result described in Section IV-B is shown as the first row of Table I. To demonstrate the benefit of adding more members to $\mathcal{A}$ using our proposed method, we run Algorithm 2 until $\mathcal{A}$ has only 100 members. As the computation of the members of $\mathcal{A}$ is performed offline, the extra time required to compute $|\mathcal{A}|=10^4$ versus $|\mathcal{A}|=100$ members is justified considering the significant improvement in the sub-optimality bound, 7.0% versus 15.0%.

As a point of comparison from our previous work, we use the hand-tuning method suggested [27], labelled as *Hand-tuned sequence of* $c(x)$ in Table I. To implement this method we run Algorithm 2 without refinement and make the modification that on line 8 we manually select a different objective at each iteration. We performed this for a range of options for the sequence of objectives and for the number of members added to $\mathcal{A}$, and the result shown in Table I is for the option

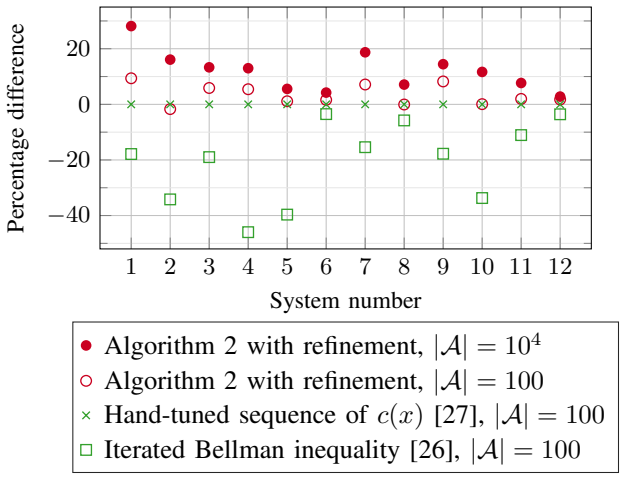

Fig. 3. Comparing the lower bound for randomly generated systems. Systems 1-6 have a discount factor $\gamma=0.99$, while systems 7-12 are respectively identical to systems 1-6 except that the discount factor is $\gamma=0.95$. For each of the four methods shown, the lower bound is computed as $\frac{1}{N}\sum_{i=1}^{N}(\max_{\bar{\alpha}\in\mathcal{A}}\bar{\alpha}^{\mathsf{T}}\phi(z_i))$, and the percentage difference is computed with respect to the best performing existing method, which is × in all cases.

achieving the tightest bound.

As a point of comparison from existing literature, we use the iterated Bellman inequality method proposed in [26], which computes multiple lower-bounding functions from the solution of one optimization problem. We use 100 Bellman inequality iterations and construct an approximate value function that is the point-wise maximum of the 100 lower-bounding functions computed, i.e., $|\mathcal{A}| = 100$. We note that more Bellman inequality iterations did not improve the result, and the iterated Bellman inequality methods proposed in [20], [25] also did not compute a tighter approximation. When using only 1 Bellman inequality this becomes the method proposed in [19] and is labelled as *Single Bellman inequality*. The results in Table I show that despite the low computation time, the sub-optimality bound is significantly less tight than our proposed method.

Another possible variant of Algorithm 2 is to perform multiple iterations of lines 7–11 in parallel, and then the step $\mathcal{A} \leftarrow \hat{\alpha} \cup \mathcal{A}$ (line 10) is performed once all parallel computations are complete. Our testing indicates that the tightest lower bound is achieved by Algorithm 2 as stated, i.e., new members are computed and added "one at a time".

The same trends were observed across different random realization of the A and B_u matrices and different discount factors, as shown in Figure 3. This figure shows that the hand-tuning method of [27] consistently outperformed the iterated Bellman inequality method of [26]. When running Algorithm 2 until $|\mathcal{A}| = 10^4$, the improvement in lower bound ranges from 2.8% to 28.2% on the systems considered. We also considered examples with a discount factor closer to 1, not shown in the interest of space, observing that with $\gamma = 0.9999$ the solver returned a finite, optimal solution at each iteration.

In summary, the comparisons in this section highlight that Algorithm 2 with refinement is an acceleration of the point-wise maximum approach proposed in [27] is the sense that lower state-relevance weighting selection effort is required, and it computes tighter lower bounds as the expense of an

acceptable increase in computation time.

D. Non-convex example

For linear quadratic examples, the optimization problems of Algorithm 1 line 8 and Algorithm 2 line 8 remain convex even when the stage cost or members of $\mathcal{A}$ are non-convex, see Appendix C. We highlight that the approach can accommodate a non-convex stage cost by considering a collision avoid example where a non-convex quadratic encodes the objective of avoiding a moving obstacle. The model represents two point-mass objects moving on a 2-dimensional plane, with each Cartesian coordinate modelled by double integrator dynamics. One sub-system is controlled by the two control actions that represent a driving force in each of the coordinate directions. The obstacle sub-system is uncontrolled and maintains constant velocity, with the two disturbances representing an exogenous driving force in each of the coordinate directions. The dynamics for a single coordinate of a single sub-system are,

$$\begin{bmatrix} x_{(\mathrm{u}),t+1} \\ \dot{x}_{(\mathrm{u}),t+1} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0.05 \\ 0 & 1 \end{bmatrix}}_{A_0} \begin{bmatrix} x_{(\mathrm{u}),t} \\ \dot{x}_{(\mathrm{u}),t} \end{bmatrix} + \underbrace{\begin{bmatrix} 0.00125 \\ 0.05 \end{bmatrix}}_{B_0} v\,, \quad (19)$$

where the subscript (u) distinguishes these states from those that follow and v is a scalar input. Letting the subscript (f) indicate the state, action, and disturbance for the full system, we order the the full state vector as,

$$x_{(\mathrm{f})} = \left[\, x_{(u)}, \dot{x}_{(u)}, y_{(u)}, \dot{y}_{(u)}, x_{(\xi)}, \dot{x}_{(\xi)}, y_{(\xi)}, \dot{y}_{(\xi)} \,\right], \quad (20)$$

where $x_{(\cdot)}$, $\dot{x}_{(\cdot)}$ and $y_{(\cdot)}$, $\dot{y}_{(\cdot)}$ represent the position and velocity in each coordinate direction of the horizontal plane, subscript (u) indicates the controlled sub-system driven by actions $u_t \in \mathbb{R}^2$, and subscript (ξ) indicates the obstacle sub-system driven by disturbances $\xi_t \in \mathbb{R}^2$. The dynamics of the combined system are thus,

$$x_{(\mathrm{f}),t+1} = \begin{bmatrix} A_1 & 0 \\ 0 & A_1 \end{bmatrix} x_{(\mathrm{f}),t} + \begin{bmatrix} B_1 \\ 0 \end{bmatrix} u_t + \begin{bmatrix} 0 \\ B_1 \end{bmatrix} \xi_t, \quad (21)$$

where $A_1 = \mathrm{diag}(A_0, A_0)$ and $B_1 = \mathrm{diag}(B_0, B_0)$. The stage cost includes a convex component for deviations from the origin,

$$\begin{aligned} l_1(x_{(\mathrm{f})}) = \; & 0.5\left(x_{(u)}^2 + y_{(u)}^2\right) + 0.005\left(x_{(\xi)}^2 + y_{(\xi)}^2\right) \\ & + 0.1\left(\dot{x}_{(u)}^2 + \dot{y}_{(u)}^2\right) + 0.001\left(\dot{x}_{(\xi)}^2 + \dot{y}_{(\xi)}^2\right), \end{aligned}$$

where a small penalty is applied to the obstacle states to satisfy the inf-compact assumption on the stage cost [5, Assumption 4.2.1(a)]. The non-convex collision avoidance term is,

$$l_2(x_{(\mathrm{f})}) = -200\left(\left(x_{(u)} - x_{(\xi)}\right)^2 + \left(y_{(u)} - y_{(\xi)}\right)^2\right) + 50,$$

which gives a penalty of 50 when the locations of the sub-systems coincide, and drops to zero when the distance between the sub-system locations is 0.5 meters. Thus the full loss function is,

$$l(x_{(\mathrm{f})}, u) \;=\; \max\left\{l_1(x_{(\mathrm{f})}), l_2(x_{(\mathrm{f})})\right\} \;+\; 0.05\, u^\intercal u, \quad (22)$$

and the discount factor is chosen as $\gamma = 0.97$. Each element of u_t is constrained to $[-1, 1]$, and the disturbance distribution is $\xi_t \sim \mathcal{N}(0, 0.4^2 I_2)$, chosen so that the exogenous forces acting on the obstacle are somewhat similar to the maximum force possible on the controlled sub-system.

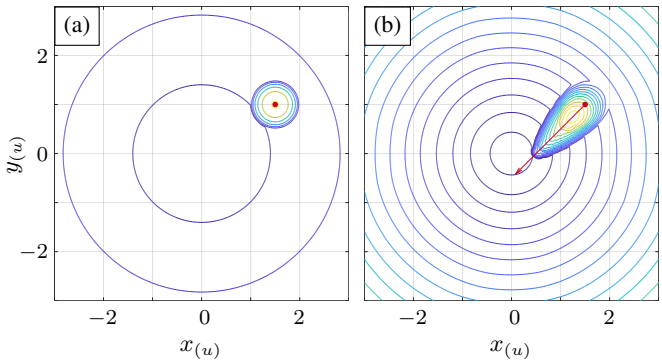

Fig. 4. Slices of: (a) the loss function, and (b) the approximate value function as a point-wise maximum of 2000 quadratics, for the collision avoidance example of Section IV-D. The red dot and arrow show the position and velocity of the obstacle to be avoided.

We use the space of quadratics as $\hat{\mathcal{F}}(\mathcal{X})$, initialize $\mathcal{A}$ with the solution of problem (18) with $\rho \sim \mathcal{N}(0, 2I_8)$, we draw $N = 10^6$ samples from the same distribution and run Algorithm 2 with refinement until $|\mathcal{A}| = 2000$. To visualize the stage cost and approximate value function, Figures 4(a) and 4(b) respectively, we show slices where the location of the controlled sub-system varies, i.e., $x_{(\mathrm{u})}$ and $y_{(\mathrm{u})}$, and all other states are fixed to: zero for the velocity of the controlled system, $(x_{(\xi)}, y_{(\xi)}) = (1.5, 1)$ for the obstacle location, and the red arrow indicating the velocity of the obstable, i.e., the head of the arrow is how far the obstacle would move in 20 time steps if zero exogenous force is applied. Figure 4 (a) shows that the stage cost has a strongly peaked concave quadratic centered at the location of the obstacle. Although the optimal value function is not computable due to the high dimension of the system, Figure 4 (b) shows that the approximate value function encodes features that we intuitively expect. Mainly that the region of high cost due to an unavoidable collision extends in the direction of the velocity of the obstacle.

This example highlights that the method proposed in this paper offers an approach with minimal tuning effort for approximating the value function of SOC problems with non-convex stage costs of the form (22).

V. Conclusion

We proposed an algorithm that computes a family of lower bounding approximate value functions in an iterative fashion, with the choice of initial state distribution as the only parameter to be selected by the designer. We motivate our algorithm by considering the non-convex objective of maximizing the point-wise maximum of lower bounding value functions, and use sub-gradient information to find (potentially) sub-optimal solutions. Testing our algorithm on linear-quadratic examples, we demonstrated a significant tightening of the lower bound compared to existing methods, achieved with a modest or negligible increase in the computation time.

As future work, we will investigate adaptations of the proposed algorithm that are tailored to policy performance.

This is a more challenging setting because the computation restrictions are more stringent for evaluation of a policy. Moreover, counter-examples can readily be constructed where an approximate value function that provides a relatively tight lower bound leads to a greedy policy with relatively poor online performance.

The concepts in this paper can be adapted to a finite horizon setting where one computes an approximate value function at each time step, with each value function represented as a point-wise maximum of functions. The core idea of applying gradient ascent to maximize the point-wise maximum objective can be directly applied. This would allow for time-varying dynamics and constraints sets, however, we expect that further developments would be required for the choice of state-relevance weighting at each time step.

APPENDIX A
REFORMULATION OF POINT-WISE MAXIMUM INEQUALITY

This appendix summarises our previous work [27] in the context of this paper.

A. Jensen's inequality and epigraph reformulation

The point-wise maximum constraint (5c) is equivalent to J separate constraints of the form,

$$l(x,u) \,+\, \gamma\, \mathbb{E}\left[\max_{k=1,\ldots,J} V_k\left(g(x,u,\xi)\right)\right] .$$

As $\max(\cdot)$ is a convex function, by Jensen's inequality a sufficient condition for constraint (5c) is,

$$\begin{aligned} V_j(x) \,\leq\, l(x,u) \,+\, \gamma \max_{k=1,\ldots,J} \mathbb{E}\left[V_k\left(g(x,u,\xi)\right)\right] , \\ \forall\, x\in\mathcal{X},\; u\in\mathcal{U},\; j=1,\ldots,J\,. \end{aligned} \tag{23}$$

An exact epigraph reformulation can now be applied [55, Theorem 1] with the epigraph variable denoted s_V, i.e.,

$$V_j(x) \,\leq\, l(x,u) \,+\, \gamma\, s_V^2\,,\; \forall\,(x,u,s_V)\in\mathcal{E}\,, \tag{24}$$

for $j=1,\ldots,J$, where the set $\mathcal{E}$ is defined as,

$$\mathcal{E} = \left\{ x,u,s_V \,\middle|\, \begin{array}{c} x\in\mathcal{X},\, u\in\mathcal{U}\,, s_V\in\mathbb{R}, \\ s_V^2 \geq \mathbb{E}\left[V_k\left(g(x,u,\xi)\right)\right]\;\forall\, k=1,\ldots,J \end{array} \right\} ,$$

We choose to square the epigraph variable s_V without loss of generality because [5, Assumptions 4.2.1(b)] implies that V^* is non-negative.

B. S-procedure reformulation

The S-procedure [56], [57] is used to obtain a sufficient condition for (24). Applying the S-procedure to the relevant part of $\mathcal{E}$ leads to,

$$\begin{aligned} V_j(x) \leq l(x,u) + \gamma\, s_V^2 - \sum_{k=1}^{J} \lambda_k \left(s_V^2 - \mathbb{E}\left[V_k\left(g(x,u,\xi)\right)\right]\right) , \\ \forall\, x\in\mathcal{X},\; u\in\mathcal{U},\; s_V\in\mathbb{R},\; j=1,\ldots,J, \end{aligned} \tag{25}$$

with $\lambda_k\in\mathbb{R}_+$ as the non-negative decision variables introduced by the S-procedure. Reformulation (25) still suffers from difficulty (D5): there will be J bilinear terms of the form $\lambda_k\,\mathbb{E}\left[V_k\left(g(x,u,\xi)\right)\right]$ in each of the J constraints. Note that the S-procedure, typically stated for polynomial inequalities, only requires that the functions in (25) are real-valued, which is ensured by the assumptions stated in Section II-A. Using the S-procedure for a non-polynomial setting is sometimes referred to as Lagrangian relaxation [57, Section 1.2.4]. The following implications summarize the approximation steps,

$$\text{(5c)} \overset{\text{(Jensen)}}{\Leftarrow} \text{(23)} \overset{\text{(Epigraph)}}{\Leftrightarrow} \text{(24)} \overset{\text{(S-procedure)}}{\Leftarrow} \text{(25)}\,.$$

In words, this reformulation is sufficient in the sense that if a family of functions $V_1,\ldots,V_J$ satisfies (25) then it also satisfies (5c) (but not necessarily the other way around). An equivalent or tighter approximation can be found by allowing the S-procedure multipliers to depend on the state and input, i.e., $\lambda_k : \mathcal{X}\times\mathcal{U}\to\mathbb{R}_+$. This would require the introduction of a restricted function space on $(\mathcal{X}\times\mathcal{U})$, denoted $\hat{\mathcal{F}}(\mathcal{X}\times\mathcal{U})$, and defined similar to $\hat{\mathcal{F}}(\mathcal{X})$ in (6).

C. Overcoming difficulty (D5) for (9)

The auxiliary problem (9) introduced in Section III-B has a form similar to (23) except that the only constraint included is the one with the decision variable on the left side of the inequality. Letting V_1 in (23) correspond to $\alpha^\intercal\phi$ in (9) and applying reformulation (25) we get the following sufficient condition for (9b),

$$\begin{aligned} \alpha^\intercal\phi(x) \,\leq\, l(x,u) \,+\, \gamma\, s_V^2 \\ - \sum_{\bar{\alpha}\in\mathcal{A}} \lambda_{\bar{\alpha}} \left(s_V^2 - \bar{\alpha}^\intercal\mathbb{E}\left[\phi(g(x,u,\xi))\right]\right) , \end{aligned}$$

for all $x\in\mathcal{X}$, $u\in\mathcal{U}$, $s_V\in\mathbb{R}$, where the multipliers $\lambda_{\bar{\alpha}}$ are additional non-negative decision variable for each element of $\mathcal{A}$. As the $\bar{\alpha}$ are fixed parameters in problem (9), it is clear that this reformulation is linear in the decision variables α and $\lambda_{\bar{\alpha}}$. When the problem data and basis functions are polynomial, the infinite constraints are reformulated in the usual way, see [20, Appendix A] for example, the result is a single Linear Matrix Inequality (LMI) constraint.

APPENDIX B
PROPERTIES OF THE INNER PROBLEM OF SECTION III-C

All the material in this appendix is formulated for a minimization optimization objective, chosen to make the results readily comparable with existing optimization literature. Problem (9) and Algorithm 1 are readily converted to minimization problems by taking the negative of the objective.

A. Differentiability definitions

We provide for completeness the definitions of the **regular** and **general** sub-differential as taken from [58, §7.D, §8.A]. The definition of the sub-differential commonly used for convex optimization problems is special case of the regular sub-differential defined below [58, Proposition 8.12], required here because (9a) is non-convex when cast as a minimization problem. We require additionally the general sub-differential definition because (9a) is non-smooth.

Given a function $f:\mathbb{R}^n \to \mathbb{R}\cup\{-\infty,\infty\}$, a vector $d\in\mathbb{R}^n$ is a *regular lower subgradient* of f at the point $x\in\mathbb{R}^n$ if the following one-sided limit condition holds,

$$\liminf_{z\to x,\, z\neq x} \frac{f(z)-f(x)-(z-x)^\intercal\, d}{\|z-x\|} \;\geq\; 0\,.$$

The *regular lower subdifferential* of f at x, denoted $\hat\partial f(x)$ is the set of regular lower subgradients of f at x. A vector $d\in\mathbb{R}^n$ is a *general lower subgradient* of f at the point x if there exists sequences $x^{(i)} \xrightarrow{f} x$ and $d^{(i)}\to d$ with $d^{(i)}\in\hat\partial f(x^{(i)})$, where the notation $\xrightarrow{f}$ stands for f-attentive, defined as,

$$x^{(i)} \xrightarrow{f} x \quad\Leftrightarrow\quad x^{(i)}\to x \;\text{ with }\; f(x^{(i)})\to f(x)\,.$$

The *general lower subdifferential* of f at x, denoted $\partial f(x)$ is the set of general lower subgradients of f at x. At a point x where f is finite, the set $\partial f(x)$ and $\hat\partial f(x)$ are closed, with $\hat\partial f(x)$ convex and $\hat\partial f(x)\subseteq\partial f(x)$. The function f is *subdifferentially regular* at a point x if $\hat\partial f(x)=\partial f(x)$. These definitions and properties correspond to [58, Definition 8.3, Theorem 8.6, Definition 7.25]. Note that if f is differentiable at x, then $\hat\partial f(x)=\{\nabla f(x)\}$, i.e., a singleton, and if additionally f is smooth on a neighbourhood of x, then $\partial f(x)=\{\nabla f(x)\}$ also. For the standard definitions of the gradient $\nabla f(x)$ of a function f at a differentiable point x, the reader is referred to [59, §B.5]. The regular and general *upper* subdifferential are computed as $-\hat\partial\,(-f)$ and $-\partial\,(-f)$, and denoted $\hat\partial^+ f$ and $\partial^+ f$, respectively.

B. Necessary condition for local optimality

A function $f:\mathbb{R}^n\to\mathbb{R}$ is proper, for a minimisation objective, if $f(x)<+\infty$ for at least one $x\in\mathbb{R}^n$, and $f(x)>-\infty$ for all $x\in\mathbb{R}^n$. Consider the minimization of a proper, lower-semi-continuous function $f:\mathbb{R}^n\to\mathbb{R}$ over a closed set $\mathcal{C}\subseteq\mathbb{R}^n$, i.e., $\min_{x\in\mathcal{C}} f(x)$. As per [58, Theorem 8.15], a necessary condition for the local optimality of a point $\overline{x}\in\mathcal{C}$ is:

$$0 \;\in\; \partial f(\overline{x}) \;+\; N_{\mathcal{C}}(\overline{x})\,, \tag{26}$$

where $N_{\mathcal{C}}$ is the *general normal cone* of the set $\mathcal{C}$ at the point $\overline{x}$, see [58, Definition 6.3]. If in addition $\mathcal{C}$ is a convex set, then this condition is equivalent to the existence of a $d\in\partial f(\overline{x})$ satisfying

$$(z-\overline{x})^\intercal\; d \;\geq\; 0\,,\quad \forall\, z\in\mathcal{C}\,. \tag{27}$$

see [58, Theorem 6.9]. Note further that if f is convex then these conditions are necessary and sufficient for $\overline{x}$ to be globally optimal. A stationary point of the optimisation problem $\min_{x\in\mathcal{C}} f(x)$ is one satisfying $0\in\partial\left(f(\overline{x})+\delta_{\mathcal{C}}(\overline{x})\right)$, where $\delta_{\mathcal{C}}$ is the indicator function of the set $\mathcal{C}$. All stationary points satisfy (26) as,

$$\partial\left(f(\overline{x})+\delta_{\mathcal{C}}(\overline{x})\right) \;\subseteq\; \partial f(\overline{x})+\partial\delta_{\mathcal{C}}(\overline{x}) \;=\; \partial f(\overline{x})+N_{\mathcal{C}}(\overline{x})\,.$$

If $\mathcal{C}$ is convex, then the inclusion becomes equality at a point $\overline{x}$ where f is sub-differentially regular, [58, Corollary 10.9].

C. Proof of convergence for a more general problem statement

To streamline the proof of Theorem 3.3, we consider here a more a general problem statement, and in Appendix B-D we show that problem (9) and Algorithm 1 has this form. Given a proper, lower-semi-continuous, **concave function** $f:\mathbb{R}^n\to\mathbb{R}$ and a **convex constraint** set $\mathcal{C}\subseteq\mathbb{R}^n$ such that f is **bounded below** on $\mathcal{C}$, we consider the optimization problem,

$$\min\; f(x),\;\text{ s.t. } x\in\mathcal{C}\,. \tag{28}$$

We show that Algorithm 3 finds points that satisfy (27); note that in line 4 we use the lower sub-differential because (28) is a minimization problem.

Algorithm 3 Find points satisfying necessary optimality conditions of problem (28)

1: **procedure** MINIMIZECONCAVEFUNCTION($x^{(0)}$, ϵ)
2: $\quad k\leftarrow 0$
3: $\quad$ **repeat**
4: $\quad\quad d^{(k)}\leftarrow$ an element from $\partial f\left(x^{(k)}\right)$
5: $\quad\quad$ **if** $\left(d^{(k)}=0\right)$ **then**
6: $\quad\quad\quad x^{(k+1)}\leftarrow x^{(k)}$
7: $\quad\quad$ **else**
8: $\quad\quad\quad x^{(k+1)}\leftarrow x^*\in\arg\min\left\{x^\intercal\, d^{(k)},\text{ s.t. } x\in\mathcal{C}\right\}$
9: $\quad\quad$ **end if**
10: $\quad\quad k\leftarrow k+1$
11: $\quad$ **until** $\left(f\left(x^{(k)}\right)-f\left(x^{(k-1)}\right)\right)<\epsilon$,
12: $\quad$ **return** $x^{(k)}$
13: **end procedure**

Theorem B.1: For any initial condition $x^{(0)}\in\mathcal{C}$ and any $\epsilon>0$, Algorithm 3 generates a non-decreasing sequence $f\left(x^{(k)}\right)$ and terminates after a finite number of iterations. With $\epsilon=0$, and assuming that the $\arg\min$ on line 8 is always attained, the sequence $f\left(x^{(k)}\right)$, converges to a finite value, and the sequences $x^{(k)}$, $d^{(k)}$, satisfy condition (27) in the following sense,

$$\lim_{k\to\infty}\left(\min_{x\in\mathcal{C}}\left(x-x^{(k)}\right)^\intercal d^{(k)}\right)\;=\;0\,.$$

Proof of Theorem B.1:

The sub-differential gives majorizing functions

We first show that given any element of the general lower subdifferential, $d^{(k)}\in\partial f\left(x^{(k)}\right)$, the surrogate function,

$$s_k\left(x\right)\;=\;\left(x-x^{(k)}\right)^\intercal\, d^{(k)}\;+\;f\left(x^{(k)}\right), \tag{29}$$

is a point-wise upper-bound of the concave function f. This is trivial for a differentiable point $x^{(k)}$ as we have that the gradient is the only element of both the general lower and upper subdifferential of f at $x^{(k)}$ and hence $s_k\left(x\right)$ is a global upper-bound of f. Pathological functions where the gradient is not an element of general subdifferential at a differentiable point are excluded by virtue of the f being concave.

At a non-differentiable point $x^{(k)}$, as f is concave, the regular lower subdifferential is empty at this point. Thus the general lower subdifferential is defined by the limits along

all sequences of differentiable points leading to $x^{(k)}$. As the regular lower and upper subdifferential are equal at all points along any such sequence we have that,

$$\partial f(x^{(k)}) \subset -\partial\left(-f\right)(x^{(k)}).$$

Thus is remains to show that $\partial\left(-f\right)(x^{(k)})$ contains only supporting hyperplanes of the hypograph of f at $x^{(k)}$. As $-f$ is convex, we have by [58, Proposition 8.12] that the general lower subdifferential of $-f$ is,

$$\begin{aligned}&\partial\left(-f(x^{(k)})\right) = \hat{\partial}\left(-f(x^{(k)})\right)\\ &= \left\{-d \in \mathbb{R}^n \middle| f(x) \leq f(x^{(k)}) + \left(x - x^{(k)}\right)^{\mathsf{T}} d,\, \forall x \in \mathcal{X}\right\},\end{aligned}$$

Thus we have shown that the surrogate function $s_k(x)$ is a point-wise upper-bound of the concave function f at any point $x^{(k)}$ that it is constructed.

Termination in finite iterations

By definition, the minimization problem on line 8 returns $x^{(k+1)}$ satisfying the optimality condition,

$$\left(x - x^{(k+1)}\right)^{\mathsf{T}} d^{(k)} \geq 0, \quad \forall\, x \in \mathcal{C}. \tag{30}$$

Combining the properties of the surrogate function s_k with the definition of line 8 as a minimization problem, we have that,

$$f\left(x^{(k)}\right) = s_k\left(x^{(k)}\right) \geq s_k\left(x^{(k+1)}\right) \geq f\left(x^{(k+1)}\right), \tag{31}$$

with $x^{(k)}, x^{(k+1)} \in \mathcal{C}$ ensured by the constraints of line 8. The equality is by (29), the first inequality is by definition of the minimization on line 8, and the final inequality is by the fact that the surrogate is a point-wise upper-bound.

By the assumption that f is bounded below on $\mathcal{C}$, the sequences $f\left(x^{(k)}\right)$ and $s_k\left(x^{(k)}\right)$, for $k \geq 0$, are convergent, hence Cauchy. Therefore, for all $\epsilon > 0$ there must exist a $k \geq 1$ such that the condition on line 11 triggers.

Convergence to necessary conditions for optimality

For $\epsilon = 0$ we have from the argument above that the sequences $f\left(x^{(k)}\right)$ and $s_k\left(x^{(k)}\right)$ converge to a finite value. To show that the sequence $x^{(k)}$ satisfies condition (27) in the limit, we need to show that,

$$\lim_{k\to\infty}\left(\sup_{d\in\partial f(x^{(k)})}\left(\min_{x\in\mathcal{C}}\left(x - x^{(k)}\right)^{\mathsf{T}} d\right)\right) \geq 0.$$

To show this it is sufficient to show that the sequence $x^{(k)}$ converges to an optimal point of $\min_{x\in\mathcal{C}} s_k(x)$, i.e., we show that sequences $x^{(k)}$, $d^{(k)}$, satisfy,

$$\lim_{k\to\infty}\left(\min_{x\in\mathcal{C}}\left(x - x^{(k)}\right)^{\mathsf{T}} d^{(k)}\right) = 0. \tag{32}$$

The $\min$ here is attained by the assumption in the theorem statement that line 8 of Algorithm 3 attains at every iteration. To show that the limit in (32) exists and equals zero, we first consider for the sake of contradiction that the sequences $x^{(k)}$, $d^{(k)}$ satisfy,

$$\liminf_{k\to\infty}\left(\min_{x\in\mathcal{C}}\left(x - x^{(k)}\right)^{\mathsf{T}} d^{(k)}\right) = -\delta < 0.$$

By definition of the $\liminf$, for every $k \geq 0$ there exists a $j \geq k$ for which,

$$\min_{x\in\mathcal{C}}\left(x - x^{(j)}\right)^{\mathsf{T}} d^{(j)} \leq -\frac{\delta}{2}. \tag{33}$$

By definition of line 8 as a minimization problem we have for this pair k, j that,

$$\begin{aligned}s_{j+1}\left(x^{(j+1)}\right) &\overset{(31)}{\leq} s_j\left(x^{(j+1)}\right)\\ &\overset{(29)}{=} \left(x^{(j+1)} - x^{(j)}\right)^{\mathsf{T}} d^{(j)} + f\left(x^{(j)}\right)\\ &\overset{(33)}{\leq} -\frac{\delta}{2} + f\left(x^{(j)}\right)\\ &\overset{(31)}{\leq} -\frac{\delta}{2} + s_k\left(x^{(k)}\right).\end{aligned}$$

Repeating this argument starting from $j+1$, we readily establish that,

$$\limsup_{k\to\infty} s_k\left(x^{(k)}\right) \leq \limsup_{N\to\infty}\left(s_0\left(x^{(0)}\right) - N\frac{\delta}{2}\right) = -\infty,$$

which contradicts the previous conclusion that the sequence $s_k\left(x^{(k)}\right)$ converges to a finite value. Moreover we have that,

$$\min_{x\in\mathcal{C}}\left(x - x^{(k)}\right)^{\mathsf{T}} d^{(k)} \leq 0, \quad \text{for } k \geq 0,$$

because $x^{(k)} \in \mathcal{C}$, for $k \geq 0$. Thus, by contradiction we have shown that,

$$\begin{aligned}0 &\geq \limsup_{k\to\infty}\left(\min_{x\in\mathcal{C}}\left(x - x^{(k)}\right)^{\mathsf{T}} d^{(k)}\right)\\ &\geq \liminf_{k\to\infty}\left(\min_{x\in\mathcal{C}}\left(x - x^{(k)}\right)^{\mathsf{T}} d^{(k)}\right) \geq 0,\end{aligned}$$

and hence the limit in (32) exists and equals zero. ■

Note that if line 5 of Algorithm 3 triggers, then the subgradient is zero and condition (27) is satisfied. In this case the $x^{(k)}$ returned is a global maximizer of the concave function f. Note also that for a positive ϵ, if the condition on line 11 triggers with $f\left(x^{(k)}\right) = f\left(x^{(k-1)}\right)$, then $x^{(k-1)}$ satisfies (27). To show this, first note that by (29) and (31) we have,

$$\begin{aligned}f\left(x^{(k-1)}\right) &\overset{(31)}{=} s_{k-1}\left(x^{(k)}\right)\\ &\overset{(29)}{=} \left(x^{(k)} - x^{(k-1)}\right)^{\mathsf{T}} d^{(k-1)} + f\left(x^{(k-1)}\right).\end{aligned}$$

From this we substitute $x^{(k)\mathsf{T}} d^{(k-1)} = x^{(k-1)\mathsf{T}} d^{(k-1)}$ into the optimality condition (30) that $x^{(k)}$ satisfies, and we get that the $d^{(k-1)} \in \partial f\left(x^{(k-1)}\right)$ from line 4 of Algorithm 3 satisfies condition (27) at $x^{(k-1)}$.

D. Proof of convergence for Algorithm 1

Proof of Theorem 3.3:

We show that the objective function f_{pwm} and the convex constraint $\alpha \in \mathcal{S}(\mathcal{A})$ satisfy the assumptions of Theorem B.1. Casting (9) as a minimization problem, the objective is,

$$-f_{\text{pwm}}(\alpha) = -\sum_{i=1}^{N}\left(\max\left\{\alpha^{\mathsf{T}}\phi(z_i), \bar{V}(z_i)\right\}\right).$$

The two elements of the $\max$ are linear in the decision variable α, and thus the objective is concave in α.

We now show that $-f_{\text{pwm}}$ is bounded below on the constraint set. The assumption that $\bar{V}$ is a point-wise lower bound of V^* means that for any α satisfying constraint (9b), the function $\alpha^\intercal\phi(x)$ is also a point-wise lower bound of $V^*(x)$ for all $x\in\mathcal{X}$. This is ensured by the Bellman operator $\mathcal{T}$ being monotone and γ-contractive. Moreover, under [5, Assumptions 4.2.1(a), 4.2.1(b), 4.2.2] we have that $V^*(x)$ is finite for all $x\in\mathcal{X}$. Thus, we have that all elements of the sum in f_{pwm} are bounded above, and hence $-f_{\text{pwm}}$ is bounded below for all $\alpha\in\mathcal{S}(\mathcal{A})$.

Next we show that equation (11) correctly computes an element of the general upper subdifferential of f_{pwm}. For an α where f_{pwm} is differentiable, we have that $\alpha^\intercal\phi(z_i)\neq\bar{V}(z_i)$ for all $i=1,\dots,N$, and thus equation (11) computes the gradient at this point. The objective function f_{pwm} is non-differentiable for an α where $\alpha^\intercal\phi(z_i)=\bar{V}(z_i)$ for at least one point $i=1,\dots,N$. Letting $\mathcal{I}_<(\alpha)$, $\mathcal{I}_=(\alpha)$, and $\mathcal{I}_>(\alpha)$ denote the indices $i=1,\dots,N$ where $\alpha^\intercal\phi(z_i)$ is respectively less than, equal, and greater than $\bar{V}(z_i)$, we define $\delta_{\min}(\alpha)$ as,

$$\delta_{\min}(\alpha) \;=\; \min_{i\,\in\,\left(\mathcal{I}_<(\alpha)\,\cup\,\mathcal{I}_>(\alpha)\right)} \left|\alpha^\intercal\phi(z_i)-\bar{V}(z_i)\right| .$$

Recall that under Assumption 3.1, ϕ_1 is taken to be the constant function and let e_1 denote a vector with 1 as the first element and zero otherwise. Thus for all $\delta\in(0,\delta_{\min})$ we have that f_{pwm} is differentiable at $(\alpha+\delta\,e_1)$ with gradient given by equation (11). For any sequence $\delta\;\to\;0$, the sequence $(\alpha+\delta\,e_1)$ is f_{pwm}-attentive, i.e., $f_{\text{pwm}}\left(\alpha+\delta\,e_1\right)\to f_{\text{pwm}}\left(\alpha\right)$ by continuity of f_{pwm}. As the gradient is the same for all $\delta\in(0,\delta_{\min})$, equation (11) correctly computes an element of the general upper subdifferential of f_{pwm} at α.

Finally, we need to show that the maximum on line 8 of Algorithm 1 is always attained. First note that the objective coefficient vector on line 8 of Algorithm 1 is given by,

$$d^{(k)} \;=\; \sum\nolimits_{\left(\mathcal{I}_>(\alpha^{(k)})\,\cup\,\mathcal{I}_=(\alpha^{(k)})\right)} \phi(z_i)\,.$$

By Assumption 3.2 we have that,

$$\max_{\alpha\in\mathbb{R}^K}\left\{\alpha^\intercal\phi(z_i)\,;\,\text{s.t.}\,\alpha\in\mathcal{S}(\mathcal{A})\right\}$$

attains its maximum for all z_i, $i=1,\dots,N$, and denote f_i^* as the optimal value. Thus the hyperplanes $\alpha^\intercal\phi(z_i)\leq f_i^*$ are all supporting hyperplanes of the convex constraint set $\mathcal{S}(\mathcal{A})$. The following finite dimensional linear program relaxation of line 8 of Algorithm 1 also attains its maximum,

$$\begin{aligned}\max_{\alpha\in\mathbb{R}^K}\;\; & \frac{1}{N}\sum\nolimits_{(\mathcal{I}_>\,\cup\,\mathcal{I}_=)}\alpha^\intercal\,\phi(z_i)\\ \text{s.t.}\;\; & \alpha^\intercal\phi(z_i)\leq f_i^*\,,\;\; i=1,\dots,N\,.\end{aligned}\tag{34}$$

To show this, first observe that (34) is feasible and bounded above by $\frac{1}{N}\sum_{(\mathcal{I}_>\,\cup\,\mathcal{I}_=)} f_i^*$, and thus by [45, Corollary 27.3.2] problem (34) attains its maximum. Finally, by [45, Corollary 27.3.3] we have that attainment for (34) implies attainment for line 8 of Algorithm 1.

We have shown that the assumptions of Theorem 3.3 satisfy also the assumptions of Theorem B.1 and hence the claims follow from Theorem B.1. ∎

APPENDIX C
PROBLEM SETTING FOR SECTION IV

A. Quadratic Basis Functions

The space of quadratic functions is parameterized by a constant offset $s\in\mathbb{R}$, a linear co-efficient $p\in\mathbb{R}^{n_x}$, and a quadratic coefficient as a symmetric matrix $P\in\mathbb{S}^{n_x}$. Thus we express the restricted function space as,

$$\hat{\mathcal{F}}(\mathcal{X}) = \left\{\hat{V}(x)\,\middle|\,\begin{matrix}V(x)=x^\intercal Px+p^\intercal x+s\\ P\in\mathbb{S}^{n_x}\,,\;p\in\mathbb{R}^{n_x}\,,\;s\in\mathbb{R}\end{matrix}\right\}.\tag{35}$$

Thus the α is the stacked vector of s, p, and the unique elements of P, and the basis functions ϕ are the monomials of x up to degree two, which clearly satisfy Assumption 3.1. Similar to Section III-A we use a subscript on s, p, and P to label the approximate value function they correspond to, for example, $\hat{V}_j\in\hat{\mathcal{F}}(\mathcal{X})$ is equivalent to $\hat{V}_j(x)=x^\intercal P_jx+p_j^\intercal x+s_j$ for all $x\in\mathcal{X}$. This space of convex quadratic functions is considered by restricting matrix P to be positive semi-definite.

B. Formulating line 8 of Algorithm 1 for commercial solver

See Section IV for the definitions of A, B_u, and B_ξ as the linear dynamics, and Appendix C-A for the specification of the quadratic basis functions. We introduce $\underline{u}_i,\overline{u}_i\in\mathbb{R}$, $i=1,\dots,n_u$, with $\underline{u}_i<\overline{u}_i$, to denote the lower and upper bounds that describe each coordinate of the $\mathcal{U}\subseteq\mathbb{R}^{n_u}$ space. The quadratic stage cost is condensed into the matrix $L\in\mathbb{R}^{(n_x+n_u+1)\times(n_x+n_u+1)}$ that takes the form $l(x,u)=[x^\intercal,u^\intercal,1]\,L\,[x^\intercal,u^\intercal,1]^\intercal$. The notation $\operatorname{diag}(\cdot)$ places the vector argument on the diagonal of an otherwise zero matrix, and e_i is the standard basis column vector with 1 in the i^{th} element and zeros elsewhere, with the dimension clear from context. We overload the notation $\hat{V}$ and introduce the notation $\hat{\mathbb{V}}$ as the following matrices,

$$\hat{V} \;=\; \begin{bmatrix}P & 0 & \frac{1}{2}p\\ \star & 0 & 0\\ \star & \star & s\end{bmatrix},$$

$$\hat{\mathbb{V}} \;=\; \begin{bmatrix}A^\intercal PA & A^\intercal PB_u & \frac{1}{2}A^\intercal p+A^\intercal PB_\xi\mathbb{E}\left[\xi\right]\\ \star & B_u^\intercal PB_u & \frac{1}{2}B_u^\intercal p+B_u^\intercal PB_\xi\mathbb{E}\left[\xi\right]\\ \star & \star & s+\operatorname{tr}\left(B_\xi^\intercal PB_\xi\mathbb{E}\left[\xi\xi^\intercal\right]\right)\end{bmatrix},$$

where $\star$ indicates that the matrix is symmetric. Again, any subscript $\hat{V}_{(\cdot)}$, $\mathbb{V}_{(\cdot)}$ also applies to s, p, and P. Both matrices are symmetric with dimension (n_x+n_u+1).

Using this notation, the point-wise maximum Bellman inequality (9b), repeated here for convenience,

$$\alpha^\intercal\phi(x)\leq\left(\mathcal{T}_u\bar{V}\right)(x,u)\,,\;\;\forall\,x\in\mathcal{X},\,u\in\mathcal{U}\,,$$

is sufficiently reformulated as the following LMI:

$$\begin{aligned}0\preceq\; & -\begin{bmatrix}\hat{V} & 0\\ \star & 0\end{bmatrix}+\begin{bmatrix}L & 0\\ \star & 0\end{bmatrix}+\begin{bmatrix}0 & 0\\ \star & \gamma\end{bmatrix}\\ & -\sum_{\bar{\alpha}\in\mathcal{A}}\lambda_{\bar{\alpha}}\begin{bmatrix}-\hat{\mathbb{V}}_{\bar{\alpha}} & 0\\ \star & 1\end{bmatrix}\\ & -\sum_{i=1}^{n_u}\lambda_i\begin{bmatrix}0_{n_x\times n_x} & 0 & 0\\ \star & -\operatorname{diag}(e_i) & \frac{1}{2}(\underline{u}_i+\overline{u}_i)e_i\\ \star & \star & -\underline{u}_i\,\overline{u}_i\end{bmatrix}.\end{aligned}\tag{37}$$

The s, p, and P in $\hat{V}$ are decision variables, as well as the $\lambda_i \in \mathbb{R}_+$ and $\lambda_{\bar{\alpha}} \in \mathbb{R}_+$, with everything else as fixed problem data. The λ_i are the auxiliary variables introduced when using the S-procedure to reformulate the for all $u \in \mathcal{U}$ part of the constraint in Appendix A-C, while the $\lambda_{\bar{\alpha}}$ are the auxiliary variables described in Appendix A-C. The objective function on line 8 of Algorithm 1 is linear in the decision variables, and when computed as per line 4 of Algorithm 1 it requires computation of the first and second moments of the z_i for the indices, $i = 1, \ldots, N$, where the approximate value function under consideration dominates $\bar{V}$. Letting μ and Σ denote the first and second moments respectively, the problem on line 8 of Algorithm 1 becomes

$$\max_{s,p,P} \left\{ \operatorname{tr}(P\Sigma) + p^{\mathsf{T}}\mu + s, \text{ s.t.(37)} \right\}, \tag{38}$$

where $\operatorname{tr}(\cdot)$ denotes the trace of a square matrix. Note that the constraint $P \succeq 0$ can be added to restrict to the space of convex quadratic functions.

REFERENCES

[1] F. Oldewurtel, A. Parisio, C. N. Jones, D. Gyalistras, M. Gwerder, V. Stauch, B. Lehmann, and M. Morari, "Use of model predictive control and weather forecasts for energy efficient building climate control," *Energy and Buildings*, vol. 45, pp. 15–27, 2012.

[2] C. Briat, A. Gupta, and M. Khammash, "Antithetic integral feedback ensures robust perfect adaptation in noisy biomolecular networks," *Cell Systems*, vol. 2, no. 1, pp. 15 – 26, 2016.

[3] R. E. Bellman, "On the theory of dynamic programming," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 38, no. 8, pp. 716–719, 1952.

[4] D. P. Bertsekas and S. E. Shreve, *Stochastic Optimal Control: The Discrete-TIme Case*. Athena Scientific, 1996.

[5] O. Hernández-Lerma and J. B. Lasserre, *Discrete-time Markov control processes: basic optimality criteria*. Springer Science & Business Media, New York, 1996.

[6] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, 2nd ed. The MIT Press, 2018.

[7] J. B. Rawlings and D. Q. Mayne, *Model Predictive Control: Theory and Design*. Nob Hill Publishing, LLC, Madison, WI, 2009.

[8] P. J. Goulart, E. C. Kerrigan, and J. M. Maciejowski, "Optimization over state feedback policies for robust control with constraints," *Automatica*, vol. 42, no. 4, pp. 523–533, 2006.

[9] A. Nemirovski and A. Shapiro, "Convex approximations of chance constrained programs," *SIAM Journal on Optimization*, vol. 17, no. 4, pp. 969–996, 2006.

[10] P. J. Werbos, "Beyond regression: new tools for prediction and analysis in the behavioral sciences," Ph.D. dissertation, Harvard University, Cambridge Massachusetts, 8 1974.

[11] J. Si, A. G. Barto, W. B. Powell, and D. Wunsch, *Handbook of learning and approximate dynamic programming*. John Wiley & Sons, 2004, vol. 2.

[12] A. Bemporad and M. Morari, "Robust model predictive control: A survey," in *Robustness in identification and control*, A. Garulli and A. Tesi, Eds. Springer London, 1999, pp. 207–226.

[13] D. Bernardini and A. Bemporad, "Scenario-based model predictive control of stochastic constrained linear systems," in *IEEE Conference on Decision and Control (CDC)*, Dec 2009, pp. 6333–6338.

[14] G. Schildbach, L. Fagiano, C. Frei, and M. Morari, "The scenario approach for stochastic model predictive control with bounds on closed-loop constraint violations," *Automatica*, vol. 50, no. 12, pp. 3009–3018, 2014.

[15] W. B. Powell, *Approximate Dynamic Programming, Solving the Curses of Dimensionality*, 2nd ed. John Wiley & Sons, Inc., Hoboken, New Jersey, 2011.

[16] D. P. Bertsekas, *Dynamic programming and optimal control, approximate dynamic programming*, 4th ed. Athena Scientific Belmont, MA, 2012, vol. 2.

[17] P. J. Schweitzer and A. Seidmann, "Generalized polynomial approximations in Markovian decision processes," *Journal of Mathematical Analysis and Applications*, vol. 110, pp. 568–582, 1985.

[18] F. D'Epenoux, "Sur un probleme de production et de stockage dans l'aléatoire," *Revue Française de Recherche Opérationelle*, vol. 14, (English translation: Management Science, Vol. 10, 1963, pp. 98–108) 1960.

[19] D. P. De Farias and B. Van Roy, "The linear programming approach to approximate dynamic programming," *Operations Research*, vol. 51, no. 6, pp. 850–865, 2003.

[20] Y. Wang, B. O'Donoghue, and S. Boyd, "Approximate dynamic programming via iterated bellman inequalities," *International Journal of Robust and Nonlinear Control*, vol. 25, no. 10, pp. 1472–1496, 2014.

[21] T. Summers, K. Kunz, N. Kariotoglou, M. Kamgarpour, S. Summers, and J. Lygeros, "Approximate dynamic programming via sum of squares programming," in *European Control Conference (ECC)*, Zürich, Switzerland, July 2013, pp. 191–197.

[22] C. Savorgnan, J. B. Lasserre, and M. Diehl, "Discrete-time stochastic optimal control via occupation measures and moment relaxations," in *Conference on Decision and Control (CDC), held jointly with the Chinese Control Conference (CCC)*. Shanghai: IEEE, December 2009, pp. 519–524.

[23] S. Darbha, K. Krishnamoorthy, M. Pachter, and P. Chandler, "State aggregation based linear programming approach to approximate dynamic programming," in *Conference on Decision and Control (CDC)*. Atlanta: IEEE, December 2010, pp. 935–941.

[24] K. Krishnamoorthy, M. Pachter, S. Darbha, and P. Chandler, "Approximate dynamic programming with state aggregation applied to UAV perimeter patrol," *International Journal of Robust and Nonlinear Control*, vol. 21, no. 12, pp. 1396–1409, 2011.

[25] B. O'Donoghue, Y. Wang, and S. Boyd, "Min-max approximate dynamic programming," in *International Symposium on Computer-Aided Control System Design (CACSD)*. IEEE, 2011, pp. 424–431.

[26] ——, "Iterated approximate value functions," in *European Control Conference (ECC)*. Zürich, Switzerland: IEEE, July 2013, pp. 3882–3888.

[27] P. Beuchat, J. Warrington, and J. Lygeros, "Point-wise maximum approach to approximate dynamic programming," in *IEEE Conference on Decision and Control (CDC)*, Dec 2017, pp. 3694–3701.

[28] M. Hohmann, J. Warrington, and J. Lygeros, "A moment and sum-of-squares extension of dual dynamic programming with application to nonlinear energy storage problems," *preprint arXiv:1807.05947*, July 2018.

[29] M. V. Pereira and L. M. Pinto, "Multi-stage stochastic optimization applied to energy planning," *Mathematical programming*, vol. 52, no. 1-3, pp. 359–375, 1991.

[30] B. Lincoln and A. Rantzer, "Relaxing dynamic programming," *IEEE Transactions on Automatic Control*, vol. 51, no. 8, pp. 1249–1260, August 2006.

[31] B. P. G. Van Parys, P. J. Goulart, and M. Morari, "Infinite horizon performance bounds for uncertain constrained systems," *IEEE Transactions on Automatic Control*, vol. 58, no. 11, pp. 2803–2817, November 2013.

[32] W. B. Powell, "A unified framework for stochastic optimization," *European Journal of Operational Research*, vol. 275, no. 3, pp. 795–821, 2019.

[33] D. P. Bertsekas, "Dynamic programming and suboptimal control: A survey from ADP to MPC," *European Journal of Control*, vol. 11, no. 4-5, pp. 310–334, 2005.

[34] B. Stellato, T. Geyer, and P. J. Goulart, "High-speed finite control set model predictive control for power electronics," *IEEE Transactions on Power Electronics*, vol. 32, no. 5, pp. 4007–4020, 2017.

[35] D. P. Bertsekas, *Reinforcement learning and optimal control*, 1st ed. Athena Scientific Belmont, MA, 2019.

[36] ——, "Stochastic optimization problems with nondifferentiable cost functionals," *Journal of Optimization Theory and Applications*, vol. 12, no. 2, pp. 218–231, 1973.

[37] D. P. De Farias and B. Van Roy, "On constraint sampling in the linear programming approach to approximate dynamic programming," *Mathematics of Operations Research*, vol. 29, no. 3, pp. 462–478, 2004.

[38] A. Keshavarz and S. Boyd, "Quadratic approximate dynamic programming for input-affine systems," *International Journal of Robust and Nonlinear Control*, vol. 24, no. 3, pp. 432–449, 2012.

[39] T. Sutter, P. M. Esfahani, and J. Lygeros, "Approximation of constrained average cost Markov control processes," in *Conference on Decision and Control (CDC)*, Los Angeles, California, USA, December 2014, pp. 6597–6602.

[40] M. Kamgarpour and T. Summers, "On infinite dimensional linear programming approach to stochastic control," in *IFAC World Congress*, vol. 50, no. 1, Toulouse, France, July 2017, pp. 6148 – 6153.

[41] D. P. De Farias and B. Van Roy, "On constraint sampling in the linear programming approach to approximate dynamic programming," *Mathematics of Operations Research*, vol. 29, no. 3, pp. 462–478, 2004.
[42] N. Kariotoglou, M. Kamgarpour, T. H. Summers, and J. Lygeros, "The linear programming approach to reach-avoid problems for Markov decision processes," *Journal of Artificial Intelligence Research*, vol. 60, pp. 263–285, 2017.
[43] P. Mohajerin Esfahani, T. Sutter, D. Kuhn, and J. Lygeros, "From infinite to finite programs: explicit error bounds with applications to approximate dynamic programming," *SIAM Journal on Optimization*, vol. 28, no. 3, pp. 1968–1998, 2018.
[44] L. Vandenberghe and S. Boyd, "Semidefinite programming," *SIAM review*, vol. 38, no. 1, pp. 49–95, 1996.
[45] R. T. Rockafellar, *Convex analysis*. Princeton university press, 2015.
[46] A. L. Yuille and A. Rangarajan, "The concave-convex procedure," *Neural Computation*, vol. 15, no. 4, pp. 915–936, 2003.
[47] G. R. Lanckriet and B. K. Sriperumbudur, "On the convergence of the concave-convex procedure," in *Advances in Neural Information Processing Systems 22*, Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, Eds. Curran Associates, Inc., 2009, pp. 1759–1767.
[48] T. Pham Dinh and H. A. Le Thi, "Convex analysis approach to dc programming: Theory, algorithms and applications," *Acta Mathematica Vietnamica*, vol. 22, no. 1, pp. 289–355, 1997.
[49] K. Khamaru and M. Wainwright, "Convergence guarantees for a class of non-convex and non-smooth optimization problems," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. Stockholmsmässan, Stockholm Sweden: PMLR, 10–15 Jul 2018, pp. 2601–2610.
[50] N. V. Sahinidis, "BARON: A general purpose global optimization software package," *Journal of Global Optimization*, vol. 8, no. 2, pp. 201–205, Mar 1996.
[51] M. R. Kılınç and N. V. Sahinidis, "Exploiting integrality in the global optimization of mixed-integer nonlinear programming problems with BARON," *Optimization Methods and Software*, vol. 33, no. 3, pp. 540–562, Jun 2018.
[52] T. Gally, M. E. Pfetsch, and S. Ulbrich, "A framework for solving mixed-integer semidefinite programs," *Optimization Methods and Software*, vol. 33, no. 3, pp. 594–632, 2018.
[53] P. N. Beuchat, A. Georghiou, and J. Lygeros, "Performance guarantees for model-based approximate dynamic programming in continuous spaces," *IEEE Transactions on Automatic Control, early access*, Mar 2019.
[54] F. Borrelli, A. Bemporad, and M. Morari, *Predictive Control for linear and hybrid systems*. Cambridge University Press, July 2017.
[55] P. N. Beuchat and J. Lygeros, "Approximate dynamic programming via penalty functions," in *IFAC World Congress*, vol. 50, 2017, pp. 11 814–11 821.
[56] I. Petersen, V. A. Ugrinovskii, and A. V. Savkin, *Robust Control Design Using H-∞ Methods*. Springer Science & Business Media, 2012.
[57] L. El Ghaoui and S.-I. Niculescu, "Robust decision problems in engineering: a linear matrix inequality approach," in *Advances in Linear Matrix Inequality Methods in Control*. SIAM Society for Industrial and Applied Mathematics, 2000, ch. 1, pp. 3–37.
[58] R. T. Rockafellar and R. J.-B. Wets, *Variational Analysis*, 3rd ed. Springer Science & Business Media, 2009.
[59] D. P. Bertsekas, *Nonlinear Programming*, 3rd ed. Athena Scientific, MA, 2016.

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 Zürich, Switzerland, in 2014. He obtained his Ph.D. degree in 2019 from ETH Zürich, pursing his research at the Automatic Control Laboratory there. He is currently working as a Teaching Fellow at the University of Melbourne in 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 building control, and coordinated flight.

Joseph Warrington is a Senior Staff Research Engineer at Home Experience LLC, Cambridge UK. Until December 2019 he was 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.

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, U.K.. In 1996 he obtained a Ph.D. degree in automatic control 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, U.K., 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 serves as the Treasurer of the International Federation of Automatic Control.