Operations Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

An Approximate Dynamic Programming Algorithm for Monotone Value Functions

Daniel R. Jiang, Warren B. Powell

To cite this article:
Daniel R. Jiang, Warren B. Powell (2015) An Approximate Dynamic Programming Algorithm for Monotone Value Functions. Operations Research

Published online in Articles in Advance 04 Nov 2015

. http://dx.doi.org/10.1287/opre.2015.1425

OPERATIONS RESEARCH
Articles in Advance, pp. 1–23
ISSN 0030-364X (print) | ISSN 1526-5463 (online)

http://dx.doi.org/10.1287/opre.2015.1425

An Approximate Dynamic Programming Algorithm for Monotone Value Functions

Daniel R. Jiang, Warren B. Powell
Department of Operations Research and Financial Engineering, Princeton University, Princeton, New Jersey 08540
{drjiang@princeton.edu, powell@princeton.edu}

Many sequential decision problems can be formulated as Markov decision processes (MDPs) where the optimal value function (or *cost-to-go* function) can be shown to satisfy a monotone structure in some or all of its dimensions. When the state space becomes large, traditional techniques, such as the backward dynamic programming algorithm (i.e., backward induction or value iteration), may no longer be effective in finding a solution within a reasonable time frame, and thus we are forced to consider other approaches, such as approximate dynamic programming (ADP). We propose a provably convergent ADP algorithm called *Monotone-ADP* that exploits the monotonicity of the value functions to increase the rate of convergence. In this paper, we describe a general finite-horizon problem setting where the optimal value function is monotone, present a convergence proof for Monotone-ADP under various technical assumptions, and show numerical results for three application domains: *optimal stopping*, *energy storage/allocation*, and *glycemic control for diabetes patients*. The empirical results indicate that by taking advantage of monotonicity, we can attain high quality solutions within a relatively small number of iterations, using up to two orders of magnitude less computation than is needed to compute the optimal solution exactly.

Keywords: approximate dynamic programming; monotonicity; optimal stopping; energy storage; glycemic control.
Subject classifications: dynamic programming/optimal control: Markov, finite state.
Area of review: Optimization.
History: Received July 2014; revisions received May 2015, July 2015; accepted August 2015. Published online in *Articles in Advance*.

1. Introduction

Sequential decision problems are an important concept in many fields, including operations research, economics, and finance. For a small, tractable problem, the *backward dynamic programming* (BDP) algorithm (also known as *backward induction* or *finite-horizon value iteration*) can be used to compute the optimal value function, from which we get an optimal decision making policy (Puterman 1994). However, the state space for many real-world applications can be immense, making this algorithm very computationally intensive. Hence, we must often turn to the field of approximate dynamic programming, which seeks to solve these problems via approximation techniques. One way to obtain a better approximation is to exploit (problem-dependent) structural properties of the optimal value function, and doing so often accelerates the convergence of ADP algorithms. In this paper, we consider the case where the optimal value function is monotone with respect to a partial order. Although this paper focuses on the theory behind our ADP algorithm and not a specific application, we first point out that our technique can be broadly utilized. Monotonicity is a very common property because it is true in many situations that "more is better." To be more precise, problems that satisfy *free disposal* (to borrow a term from economics) or *no holding costs* are likely to contain monotone structure. There are also less obvious ways that monotonicity can come into play, such as environmental variables that influence the stochastic evolution of a primary state variable (e.g., extreme weather can lead to increased expected travel times; high natural gas prices can lead to higher electricity spot prices). The following list is a small sample of real-world applications spanning the literature of the aforementioned disciplines (and their subfields) that satisfy the special property of monotone value functions.

Operations Research

- The problem of optimal replacement of machine parts is well studied in the literature (see e.g., Feldstein and Rothschild 1974, Pierskalla and Voelker 1976, and Rust 1987) and can be formulated as a regenerative optimal stopping problem in which the value function is monotone in the current health of the part and the state of its environment. Section 7 discusses this model and provides detailed numerical results.
- The problem of batch servicing of customers at a service station as discussed in Papadaki and Powell (2002) features a value function that is monotone in the number of customers. Similarly, the related problem of multiproduct batch dispatch studied in Papadaki and Powell (2003b) can be shown to have a monotone value function in the multidimensional state variable that contains the number of products awaiting dispatch.

Energy

• In the energy storage and allocation problem, one must optimally control a storage device that interfaces with the spot market and a stochastic energy supply (such as wind or solar). The goal is to reliably satisfy a possibly stochastic demand in the most profitable way. We can show that without holding costs, the value function is monotone in the resource (see Scott and Powell 2012 and Salas and Powell 2013). Once again, refer to §7 for numerical work in this problem class.

• The value function from the problem of maximizing revenue using battery storage while bidding hourly in the electricity market can be shown to satisfy monotonicity in the resource, bid, and remaining battery lifetime (see Jiang and Powell 2015).

Healthcare

• Hsih (2010) develops a model for optimal dosing applied to glycemic control in diabetes patients. At each decision epoch, one of several treatments (e.g., sensitizers, secretagogues, alpha-glucosidase inhibitors, or peptide analogs) with varying levels of "strength" (i.e., ability to decrease glucose levels) but also varying side effects, such as weight gain, needs to be administered. The value function in this problem is monotone whenever the utility function of the state of health is monotone. See §7 for the complete model and numerical results.

• Statins are often used as treatment against heart disease or stroke in diabetes patients with lipid abnormalities. The optimal time for statin initiation, however, is a difficult medical problem due to the competing forces of health benefits and side effects. Kurt et al. (2011) models the problem as an MDP with a value function monotone in a risk factor known as the lipid ratio.

Finance

• The problem of mutual fund cash balancing, described in Nascimento and Powell (2010), is faced by fund managers who must decide on the amount of cash to hold, taking into account various market characteristics and investor demand. The value functions turn out to be monotone in the interest rate and the portfolio's rate of return.

• The pricing problem for American options (see Luenberger 1998) uses the theory of optimal stopping and depending on the model of the price process, monotonicity can be shown in various state variables: for example, the current stock price or the volatility (see Ekström 2004).

Economics

• Kaplan and Violante (2014) model the decisions of consumers after receiving fiscal stimulus payments to explain observed consumption behavior. The household has both liquid and illiquid assets (the state variable), in which the value functions are clearly monotone.

• A classical model of search unemployment in economics describes a situation where at each period, a worker has a decision of accepting a wage offer or continuing to search for employment. The resulting value functions can be shown to be increasing with wage (see §10.7 of Stockey and Lucas 1989 and McCall 1970).

This paper makes the following contributions. We describe and prove the convergence of an algorithm, called *Monotone-ADP* (*M-ADP*) for learning monotone value functions by preserving monotonicity after each update. We also provide empirical results for the algorithm in the context of various applications in operations research, energy, and healthcare as experimental evidence that exploiting monotonicity dramatically improves the rate of convergence. The performance of *Monotone-ADP* is compared to several established algorithms: kernel-based reinforcement learning (Ormoneit and Sen 2002), approximate policy iteration (Bertsekas 2011), asynchronous value iteration (Bertsekas 2007), and Q-learning (Watkins and Dayan 1992).

The paper is organized as follows. Section 2 gives a literature review, followed by the problem formulation and algorithm description in §§3 and 4. Next, §5 provides the assumptions necessary for convergence, and §6 states and proves the convergence theorem, with several proofs of lemmas and propositions postponed until the appendix and online supplement (available as supplemental material at http://dx.doi.org/10.1287/opre.2015.1425). Section 7 describes numerical experiments over a suite of problems, with the largest one having a seven dimensional state variable and nearly 20 million states per time period. We conclude in §8.

2. Literature Review

General monotone functions (not necessarily a value function) have been extensively studied in the academic literature. The statistical estimation of monotone functions is known as *isotonic* or *monotone* regression and has been studied as early as 1955; see Ayer et al. (1955) or Brunk (1955). The main idea of isotonic regression is to minimize a weighted error under the constraint of monotonicity (see Barlow et al. 1972 for a thorough description). The problem can be solved in a variety of ways, including the *Pool Adjacent Violators Algorithm* (*PAVA*) described in Ayer et al. (1955). More recently, Mammen (1991) builds upon this previous research by describing an estimator that combines kernel regression and PAVA to produce a smooth regression function. Additional studies from the statistics literature include Mukerjee (1988), Ramsay (1998), and Dette et al. (2006). Although these approaches are outside the context of dynamic programming, that they were developed and well studied highlights the pertinence of monotone functions.

From the operations research literature, monotone value functions and conditions for monotone optimal policies are broadly described in Puterman (1994, §4.7) and some general theory is derived therein. Similar discussions of the topic can be found in Ross (1983), Stockey and Lucas

(1989), Müller (1997), and Smith and McCardle (2002). The algorithm that we describe in this paper is first used in Papadaki and Powell (2002) as a heuristic to solve the *stochastic batch service problem*, where the value function is monotone. However, the convergence of the algorithm is not analyzed and the state variable is scalar. Finally, in Papadaki and Powell (2003a), the authors prove the convergence of the *Discrete Online Monotone Estimation* (DOME) algorithm, which takes advantage of a monotonicity preserving step to iteratively estimate a discrete monotone function. DOME, though, was not designed for dynamic programming, and the proof of convergence requires independent observations across iterations, which is an assumption that cannot be made for Monotone-ADP.

Another common property of value functions, especially in resource allocation problems, is convexity/concavity. Rather than using a monotonicity preserving step as Monotone-ADP does, algorithms such as the *Successive Projective Approximation Routine* (*SPAR*) of Powell et al. (2004), the *Lagged Acquisition ADP Algorithm* of Nascimento and Powell (2009), and the *Leveling Algorithm* of Topaloglu and Powell (2003) use a concavity preserving step, which is the same as maintaining monotonicity in the slopes. The proof of convergence for our algorithm, Monotone-ADP, uses ideas found in Tsitsiklis (1994) (later also used in Bertsekas and Tsitsiklis 1996) and Nascimento and Powell (2009). Convexity has also been exploited successfully in multistage linear stochastic programs (see, e.g, Birge 1985, Pereira and Pinto 1991, and Asamov and Powell 2015). In our work, we take as inspiration the value of convexity demonstrated in the literature and show that monotonicity is another important structural property that can be leveraged in an ADP setting.

3. Mathematical Formulation

We consider a generic problem with a time horizon, $t = 0, 1, 2, \ldots, T$. Let $\mathcal{S}$ be the state space under consideration, where $|S| < \infty$, and let $\mathcal{A}$ be the set of actions or decisions available at each time step. Let $S_t \in \mathcal{S}$ be the random variable representing the state at time t and $a_t \in \mathcal{A}$ be the action taken at time t. For a state $S_t \in \mathcal{S}$ and an action $a_t \in \mathcal{A}$, let $C_t(S_t, a_t)$ be a contribution or reward received in period t and $C_T(S_T)$ be the terminal contribution. Let $A_t^\pi\colon \mathcal{S} \to \mathcal{A}$ be the decision function at time t for a policy π from the class Π of all admissible policies. Our goal is to maximize the expected total contribution, giving us the following objective function:

$$\sup_{\pi\in\Pi} \mathbf{E}\left[\sum_{t=0}^{T-1} C_t(S_t, A_t^\pi(S_t)) + C_T(S_T)\right],$$

where we seek a policy to choose the actions a_t sequentially based on the states S_t that we visit. Let $(W_t)_{t=0}^T$ be a discrete time stochastic process that encapsulates all of the randomness in our problem; we call it the *information process*. Assume that $W_t \in \mathcal{W}$ for each t and that there exists a state transition function $f\colon \mathcal{S} \times \mathcal{A} \times \mathcal{W} \to \mathcal{S}$ that describes the evolution of the system. Given a current state S_t, an action a_t, and an outcome of the information process W_{t+1}, the next state is given by

$$S_{t+1} = f(S_t, a_t, W_{t+1}). \tag{1}$$

Let $s \in \mathcal{S}$. The optimal policy can be expressed through a set of optimal value functions using the well-known Bellman's equation:

$$V_t^*(s) = \sup_{a\in\mathcal{A}}[C_t(s,a) + \mathbf{E}[V_{t+1}^*(S_{t+1}) \mid S_t = s, a_t = a]] \quad \text{for } t = 0, 1, 2, \ldots, T-1, \tag{2}$$

$$V_T^*(s) = C_T(s),$$

with the understanding that S_{t+1} transitions from S_t according to (1). In many cases, the terminal contribution function $C_T(S_T)$ is zero. Suppose that the state space $\mathcal{S}$ is equipped with a partial order, denoted $\preceq$, and the following monotonicity property is satisfied for every t:

$$s \preceq s' \implies V_t^*(s) \leqslant V_t^*(s'). \tag{3}$$

In other words, the optimal value function V_t^* is *order-preserving* over the state space $\mathcal{S}$. In the case where the state space is multidimensional (see §7 for examples), a common example of $\preceq$ is *componentwise inequality*, which we henceforth denote using the traditional $\leqslant$.

A second example that arises very often is the following definition of $\preceq$, which we call the *generalized componentwise inequality*. Assume that each state s can be decomposed into $s = (m, i)$ for some $m \in \mathcal{M}$ and $i \in \mathcal{I}$. For two states $s = (m, i)$ and $s' = (m', i')$, we have

$$s \preceq s' \iff m \leqslant m',\ i = i'. \tag{4}$$

In other words, we know that whenever i is held constant, then the value function is monotone in the "primary" variable m. An example of when such a model would be useful is when m represents the amount of some held resource that we are both buying and selling, while i represents additional *state-of-the-world* information, such as prices of related goods, transport times on a shipping network, or weather information. Depending on the specific model, the relationship between the value of i and the optimal value function may be quite complex and a priori unknown to us. However, it is likely to be obvious that for i held constant, the value function is increasing in m, the amount of resource that we own. Hence, the definition (4) is natural for this situation. The following proposition is given in the setting of the generalized componentwise inequality and provides a simple condition that can be used to verify monotonicity in the value function.

PROPOSITION 1. *Suppose that every $s \in \mathcal{S}$ can be written as $s = (m, i)$ for some $m \in \mathcal{M}$ and $i \in \mathcal{I}$, and let $S_t = (M_t, I_t)$ be the state at time t, with $M_t \in \mathcal{M}$ and $I_t \in \mathcal{I}$. Let the partial order $\preceq$ on the state space $\mathcal{S}$ be described by (4). Assume the following assumptions hold.*

(i) *For every $s, s' \in \mathcal{S}$ with $s \preceq s'$, $a \in \mathcal{A}$, and $w \in \mathcal{W}$, the state transition function satisfies*

$$f(s, a, w) \preceq f(s', a, w).$$

(ii) *For each $t < T$, $s, s' \in \mathcal{S}$ with $s \preceq s'$, and $a \in \mathcal{A}$,*

$$C_t(s, a) \leqslant C_t(s', a) \quad \text{and} \quad C_T(s) \leqslant C_T(s').$$

(iii) *For each $t < T$, M_t and W_{t+1} are independent.*

Then the value functions V_t^* satisfy the monotonicity property of (3).

Proof. See the online supplement.

There are other similar ways to check for monotonicity; for example, see Proposition 4.7.3 of Puterman (1994) or Theorem 9.11 of Stockey and Lucas (1989) for conditions on the transition probabilities. We choose to provide the above proposition because of its relevance to our example applications in §7.

The most traditional form of Bellman's equation has been given in (2), which we refer to as the *pre-decision state* version. Next, we discuss some alternative formulations from the literature that can be very useful for certain problem classes. A second formulation, called the Q-function (or *state-action*) form Bellman's equation, is popular in the field of reinforcement learning, especially in applications of the widely used Q-learning algorithm (see Watkins and Dayan 1992):

$$Q_t^*(s,a) = \mathbf{E}\Big[C_t(s,a) + \max_{a_{t+1}\in\mathscr{A}} Q_{t+1}^*(S_{t+1}, a_{t+1}) \,\Big|\, S_t = s, \; a_t = a\Big] \quad \text{for } t = 0, 1, 2, \ldots, T-1, \tag{5}$$
$$Q_T^*(s,a) = C_T(s),$$

where we must now impose the additional requirement that $\mathscr{A}$ is a finite set. Q^* is known as the *state-action value function* and the "state space" in this case is enlarged to be $\mathscr{S} \times \mathscr{A}$.

A third formulation of Bellman's equation is in the context of *post-decision states* (see Powell 2011 for a detailed treatment of this important technique). Essentially, the post-decision state, which we denote S_t^a, represents the state after the decision has been made, but before the random information W_{t+1} has arrived (the state-action pair is also a post-decision state). For example, in the simple problem of purchasing additional inventory x_t to the current stock R_t to satisfy a next-period stochastic demand, the post-decision state can be written as $R_t + x_t$, and the pre-decision state is R_t. It must be the case that S_t^a contains the same information as the state-action pair (S_t, a_t), meaning that regardless of whether we condition on S_t^a or (S_t, a_t), the conditional distribution of W_{t+1} is the same. The attractiveness of this method is that (1) in certain problems, S_t^a is of lower dimension than (S_t, a_t) and (2) when writing Bellman's equation in terms of the post-decision state space (using a redefined value function), the supremum and the expectation are interchanged, giving us some computational advantages. Let s^a be a post-decision state from the post-decision state space $\mathscr{S}^a$. Bellman's equation becomes

$$V_t^{a,*}(s^a) = \mathbf{E}\Big[\sup_{a\in\mathscr{A}}[C_{t+1}(S_{t+1}, a) + V_{t+1}^{a,*}(S_{t+1}^a)] \,|\, S_t^a = s^a\Big] \quad \text{for } t = 0, 1, 2, \ldots, T-2, \tag{6}$$
$$V_{T-1}^{a,*}(s^a) = \mathbf{E}[C_T(S_T) \,|\, S_{T-1}^a = s^a],$$

where $V^{*,a}$ is known as the *post-decision value function*. In approximate dynamic programming, the original Bellman's equation formulation (2) can be used if the transition probabilities are known. When the transition probabilities are unknown, we must often rely purely on *experience* or some form of *black box simulator*. In these situations, formulations (5) and (6) of Bellman's equation, where the optimization is within the expectation, become extremely useful. For the remainder of this paper, rather than distinguishing between the three forms of the value function (V^*, Q^*, and $V^{a,*}$), we simply use V^* and call it the *optimal value function*, with the understanding that it may be replaced with any of the definitions. Similarly, to simplify notation, we do not distinguish between the three forms of the state space ($\mathscr{S}$, $\mathscr{S} \times \mathscr{A}$, and $\mathscr{S}^a$) and simply use $\mathscr{S}$ to represent the domain of the value function (for some t).

Let $d = |\mathscr{S}|$ and $D = (T+1)|\mathscr{S}|$. We view the optimal value function as a vector in $\mathbb{R}^D$; that is to say, $V^* \in \mathbb{R}^D$ has a component at (t, s) denoted as $V_t^*(s)$. Moreover, for a fixed $t \leqslant T$, the notation $V_t^* \in \mathbb{R}^d$ is used to describe V^* restricted to t; i.e., the components of V_t^* are $V_t^*(s)$ with s varying over $\mathscr{S}$. We adopt this notational system for arbitrary value functions $V \in \mathbb{R}^D$ as well. Finally, we define the *generalized dynamic programming operator* $H\colon \mathbb{R}^D \to \mathbb{R}^D$, which applies the right-hand sides of either (2), (5), or (6) to an arbitrary $V \in \mathbb{R}^D$, i.e., replacing V_t^*, Q_t^*, and $V_t^{*,a}$ with V_t. For example, if H is defined in the context of (2), then the component of HV at (t, s) is given by

$$(HV)_t(s) = \begin{cases} \sup_{a\in\mathscr{A}}[C_t(s,a) + \mathbf{E}[V_{t+1}(S_{t+1}) \,|\, S_t = s, a_t = a]] \\ \qquad \text{for } t = 0, 1, 2, \ldots, T-1, \\ C_T(s) \quad \text{for } t = T. \end{cases} \tag{7}$$

For (5) and (6), H can be defined in an analogous way. We now state a lemma concerning useful properties of H. Parts of it are similar to Assumption 4 of Tsitsiklis (1994), but we can show that these statements always hold true for our more specific problem setting, where H is a generalized dynamic programming operator.

Lemma 1. *The following statements are true for H, when it is defined using* (2), (5), *or* (6).

(i) *H is monotone; i.e., for $V, V' \in \mathbb{R}^D$ such that $V \leqslant V'$, we have that $HV \leqslant HV'$ (componentwise).*

(ii) *For any $t < T$, let $V, V' \in \mathbb{R}^D$, such that $V_{t+1} \leqslant V'_{t+1}$. It then follows that $(HV)_t \leqslant (HV')_t$.*

(iii) *The optimal value function V^* uniquely satisfies the fixed point equation $HV = V$.*

(iv) *Let $V \in \mathbb{R}^D$ and e is a vector of ones with dimension D. For any $\eta > 0$,*

$$HV - \eta e \leqslant H(V - \eta e) \leqslant H(V + \eta e) \leqslant HV + \eta e.$$

Proof. See Appendix A.

4. Algorithm

In this section, we formally describe the Monotone-ADP algorithm. Assume a probability space $(\Omega, \mathcal{F}, \mathbf{P})$ and let $\bar{V}^n$ be the approximation of V^* at iteration n, with the random variable $S_t^n \in \mathcal{S}$ representing the state that is visited (by the algorithm) at time t in iteration n. The observation of the optimal value function at time t, iteration n, and state S_t^n is denoted $\hat{v}_t^n(S_t^n)$ and is calculated using the estimate of the value function from iteration $n-1$. The raw observation $\hat{v}_t^n(S_t^n)$ is then *smoothed* with the previous estimate $\bar{V}_t^{n-1}(S_t^n)$, using a stochastic approximation step, to produce the smoothed observation $z_t^n(S_t^n)$. Before presenting the description of the ADP algorithm, some definitions need to be given. We start with Π_M, the monotonicity preserving projection operator. Note that the term "projection" is being used loosely here; the space that we "project" onto actually changes with each iteration.

Definition 1. For $s^r \in \mathcal{S}$ and $z^r \in \mathbb{R}$, let (s^r, z^r) be a *reference point* to which other states are compared. Let $V_t \in \mathbb{R}^d$ and define the projection operator $\Pi_M\colon \mathcal{S} \times \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^d$, where the component of the vector $\Pi_M(s^r, z^r, V_t)$ at s is given by

$$\Pi_M(s^r, z^r, V_t)(s) = \begin{cases} z^r & \text{if } s = s^r, \\ z^r \vee V_t(s) & \text{if } s^r \preceq s,\ s \neq s^r, \\ z^r \wedge V_t(s) & \text{if } s^r \succeq s,\ s \neq s^r, \\ V_t(s) & \text{otherwise.} \end{cases} \tag{8}$$

In the context of the Monotone-ADP algorithm, V_t is the current value function approximation, (s^r, z^r) is the latest observation of the value (s^r is latest visited state), and $\Pi_M(s^r, z^r, V_t)$ is the updated value function approximation. Violations of the monotonicity property of (3) are corrected by Π_M in the following ways:

- if $z^r \geqslant V_t(s)$ and $s^r \preceq s$, then $V_t(s)$ is *too small* and is increased to $z^r = z^r \vee V_t(s)$ and
- if $z^r \leqslant V_t(s)$ and $s^r \succeq s$, then $V_t(s)$ is *too large* and is decreased to $z^r = z^r \wedge V_t(s)$.

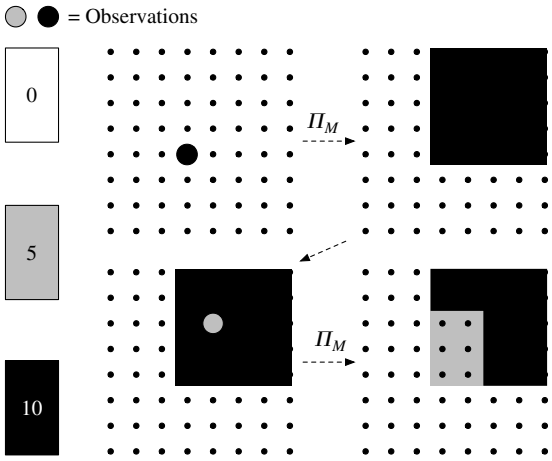
See Figure 1 for an example showing a sequence of two observations and the resulting projections in the Cartesian plane, where $\preceq$ is the componentwise inequality in two dimensions. We now provide some additional motivation for the definition of Π_M. Because $z_t^n(S_t^n)$ is the *latest* observed value and it is obtained via stochastic approximation (see the Step 2b of Figure 2), our intuition guides us to "keep" this value, i.e., by setting $\bar{V}_t^n(S_t^n) = z_t^n(S_t^n)$. For $s \in \mathcal{S}$ and $v \in \mathbb{R}$, let us define the set

$$\mathcal{V}_{\mathcal{M}}(s, z) = \{V \in \mathbb{R}^d\colon V(s) = z,\ V \text{ monotone over } \mathcal{S}\}$$

which fixes the value at s to be z while restricting to the set of all possible V that satisfy the monotonicity property (3). Now, to get the approximate value function of iteration n and time t, we want to find $\bar{V}_t^n$ that is close to $\bar{V}_t^{n-1}$ but also satisfies the monotonicity property:

$$\bar{V}_t^n \in \arg\min\{\|V_t - \bar{V}_t^{n-1}\|_2\colon V_t \in \mathcal{V}_{\mathcal{M}}(S_t^n, z_t^n(S_t^n))\}, \tag{9}$$

Figure 1. Example illustrating the projection operator Π_M.



where $\|\cdot\|_2$ is the Euclidean norm. Let us now briefly pause and consider a possible alternative, where we do not require $\bar{V}_t^n(S_t^n) = z_t^n(S_t^n)$. Instead, suppose we introduce a vector $\hat{V}_t^{n-1} \in \mathbb{R}^d$ such that $\hat{V}_t^{n-1}(s) = \bar{V}_t^{n-1}(s)$ for $s \neq S_t^n$ and $\hat{V}_t^{n-1}(S_t^n) = z_t^n(S_t^n)$. Next, project $\hat{V}_t^{n-1}$, the space of vectors V that are monotone over $\mathcal{S}$, to produce $\bar{V}_t^n$ (this would be a proper projection, where the space does not change). The problem with this approach arises in the early iterations where we have poor estimates of the value function: for example, if $\bar{V}_t^0(s) = 0$ for all s, then $\hat{V}_t^0$ is a vector of mostly zeros and the likely result of the projection, $\bar{V}_t^1$, would be the *original vector* $\bar{V}_t^0$—hence, no progress is made. A potential explanation for the failure of such a strategy is that it is a naive adaptation of the natural approach for a batch framework to a recursive setting.

The next proposition shows that this representation of $\bar{V}_t^n$ is equivalent to one that is obtained using the projection operator Π_M.

Proposition 2. *The solution to the* minimization (9) *can be characterized using* Π_M. *Specifically,*

$$\Pi_M(S_t^n, z_t^n(S_t^n), \bar{V}_t^{n-1}) \in \arg\min\{\|V_t - \bar{V}_t^{n-1}\|_2\colon V_t \in \mathcal{V}_{\mathcal{M}}(S_t^n, z_t^n(S_t^n))\},$$

so that we can write $\bar{V}_t^n = \Pi_M(S_t^n, z_t^n(S_t^n), \bar{V}_t^{n-1})$.

Proof. See Appendix B.

We now introduce, for each t, a (possibly stochastic) stepsize sequence $\alpha_t^n \leqslant 1$ used for smoothing in new observations. The algorithm only directly updates values (i.e., not including updates from the projection operator) for states that are visited, so for each $s \in \mathcal{S}$, let

$$\alpha_t^n(s) = \alpha_t^{n-1} \mathbf{1}_{\{s = S_t^n\}}.$$

Let $\hat{v}_t^n \in \mathbb{R}^d$ be a noisy observation of the quantity $(H\bar{V}^{n-1})_t$, and let $w_t^n \in \mathbb{R}^d$ represent the additive noise associated with the observation:

$$\hat{v}_t^n = (H\bar{V}^{n-1})_t + w_t^n.$$

Figure 2. Monotone-ADP algorithm.

Step 0a. Initialize $\bar{V}_t^0 \in [0, V_{\max}]$ for each $t \leqslant T-1$ such that monotonicity is satisfied within $\bar{V}_t^0$, as described in (3).
Step 0b. Set $\bar{V}_T^n(s) = C_T(s)$ for each $s \in \mathcal{S}$ and $n \leqslant N$.
Step 0c. Set $n = 1$.
Step 1. Select an initial state S_0^n.
Step 2. For $t = 0, 1, \ldots, (T-1)$:
Step 2a. Sample a noisy observation of the future value: $\hat{v}_t^n = (H\bar{V}^{n-1})_t + w_t^n$.
Step 2b. Smooth in the new observation with previous value at each s: $z_t^n(s) = (1-\alpha_t^n(s))\bar{V}_t^{n-1}(s) + \alpha_t^n(s)\hat{v}_t^n(s)$.
Step 2c. Perform monotonicity projection operator: $\bar{V}_t^n = \Pi_M(S_t^n, z_t^n(S_t^n), \bar{V}_t^{n-1})$.
Step 2d. Choose the next state S_{t+1}^n given $\mathcal{F}^{n-1}$.
Step 3. If $n < N$, increment n and return to *Step* 1.

Although the algorithm is asynchronous and only updates the value for S_t^n (therefore, it only needs $\hat{v}_t^n(S_t^n)$, the component of $\hat{v}_t^n$ at S_t^n), it is convenient to assume $\hat{v}_t^n(s)$ and $w_t^n(s)$ are defined for all s. We also require a vector $z_t^n \in \mathbb{R}^d$ to represent the "smoothed observation" of the future value; i.e., $z_t^n(s)$ is $\hat{v}_t^n(s)$ smoothed with the previous value $\bar{V}_t^{n-1}(s)$ via the stepsize $\alpha_t^n(s)$. Let us denote the history of the algorithm up until iteration n by the filtration $\{\mathcal{F}^n\}_{n\geqslant 1}$, where

$$\mathcal{F}^n = \sigma\{(S_t^m, w_t^m)_{m\leqslant n,\, t\leqslant T}\}.$$

A precise description of the algorithm is given in Figure 2. Notice from the description that if the monotonicity property (3) is satisfied at iteration $n-1$, then the fact that the projection operator Π_M is applied ensures that the monotonicity property is satisfied again at time n. Our benchmarking results of §7 show that maintaining monotonicity in such a way is an invaluable aspect of the algorithm that allows it to produce very good policies in a relatively small number of iterations. Traditional approximate (or asynchronous) value iteration, on which Monotone-ADP is based, is asymptotically convergent but *extremely* slow to converge in practice (once again, see §7). As we have mentioned, Π_M is not a standard projection operator, as it "projects" to a different space on every iteration, depending on the state visited and value observed; therefore, traditional convergence results no longer hold. The remainder of the paper establishes the asymptotic convergence of Monotone-ADP.

4.1. Extensions of Monotone-ADP

We now briefly present two possible extensions of Monotone-ADP. First, consider a discounted, infinite horizon MDP. An extension (or perhaps, simplification) to this case can be obtained by removing the loop over t (and all subscripts of t and T) and acquiring one observation per iteration, exactly resembling asynchronous value iteration for infinite horizon problems.

Second, we consider possible extensions when representations of the approximate value function other than lookup table are used; for example, imagine we are using basis functions $\{\phi_g\}_{g\in\mathcal{G}}$ for some feature set $\mathcal{G}$ combined with a coefficient vector θ_t^n (which has components θ_{tg}^n), giving the approximation

$$\bar{V}_t^n(s) = \sum_{g\in\mathcal{G}} \theta_{tg}^n \phi_g(s).$$

Equation (9) is the starting point for adapting Monotone-ADP to handle this case. An analogous version of this update might be given by

$$\theta_t^n \in \arg\min\{\|\theta_t - \theta_t^{n-1}\|_2 \colon \bar{V}_t^n(S_t^n) = z_t^n(S_t^n) \text{ and } \bar{V}_t^n \text{ monotone}\}, \tag{10}$$

where we have altered the objective to minimize distance in the coefficient space. Unlike (9), there is, in general, no simple and easily computable solution to (10), but special cases may exist. The analysis of this situation is beyond the scope of this paper and left to future work. In this paper, we consider the finite horizon case using a lookup table representation.

5. Assumptions

We begin by providing some technical assumptions that are needed for convergence analysis. The first assumption gives, in more general terms than previously discussed, the monotonicity of the value functions.

Assumption 1. *The two monotonicity assumptions are as follows.*

(i) *The terminal value function C_T is monotone over $\mathcal{S}$ with respect to $\preceq$.*

(ii) *For any $t < T$ and any vector $V \in \mathbb{R}^D$ such that V_{t+1} is monotone over $\mathcal{S}$ with respect to $\preceq$, it is true that $(HV)_t$ is monotone over the state space as well.*

The above assumption implies that for any choice of terminal value function $V_T^* = C_T$ that satisfies monotonicity, the value functions for the previous time periods are monotone as well. Examples of sufficient conditions include monotonicity in the contribution function plus a condition on the transition function, as in (i) of Proposition 1, or a condition on the transition probabilities, as in Proposition 4.7.3 of Puterman (1994). Intuitively speaking, when the statement *"starting with more at $t \Rightarrow$ ending with more at $t+1$"* applies, in expectation, to the problem at hand, Assumption 1 is satisfied. One obvious example that satisfies monotonicity occurs in resource or asset management scenarios; oftentimes in these problems, it is true that for any outcome of the random information W_{t+1} that occurs (e.g., random demand, energy production, or profits), we end with more of the resource at time $t+1$ whenever we start with more of the resource at time t. Mathematically, this property of resource allocation problems translates to the stronger statement:

$$(S_{t+1} \mid S_t = s, a_t = a) \preceq (S_{t+1} \mid S_t = s', a_t = a.) \quad \text{a.s.}$$

for all $a \in \mathcal{A}$ when $s \preceq s'$. This is essentially the situation that Proposition 1 describes.

Assumption 2. *For all $s \in \mathcal{S}$ and $t < T$, the sampling policy satisfies*

$$\sum_{n=1}^{\infty} \mathbf{P}(S_t^n = s \mid \mathcal{F}^{n-1}) = \infty \quad \text{a.s.}$$

By the Extended Borel-Cantelli Lemma (see Breiman 1992), any scheme for choosing states that satisfies the above condition will visit every state infinitely often with probability one.

Assumption 3. *Suppose that the contribution function $C_t(s,a)$ is bounded: without loss of generality, let us assume that for all $s \in \mathcal{S}$, $t < T$, and $a \in \mathcal{A}$, $0 \leqslant C_t(s,a) \leqslant C_{\max}$, for some $C_{\max} > 0$. Furthermore, suppose that $0 \leqslant C_T(s) \leqslant C_{\max}$ for all $s \in \mathcal{S}$ as well. This naturally implies that there exists $V_{\max} > 0$ such that $0 \leqslant V_t^*(s) \leqslant V_{\max}$.*

The next three assumptions are standard ones made on the observations $\hat{v}_t^n$, the noise w_t^n, and the stepsize sequence α_t^n; see Bertsekas and Tsitsiklis (1996) (e.g., Assumption 4.3 and Proposition 4.6) for additional details.

Assumption 4. *The observations that we receive are bounded (by the same constant $V_{\max}$): $0 \leqslant \hat{v}_t^n(s) \leqslant V_{\max}$ almost surely, for all $s \in \mathcal{S}$ and $t < T$.*

Note that the lower bounds of zero in Assumptions 3 and 4 are chosen for convenience and can be shifted by a constant to suit the application (as is done in §7).

Assumption 5. *The following holds almost surely: $\mathbf{E}[w_t^{n+1}(s) \mid \mathcal{F}^n] = 0$, for any state $s \in \mathcal{S}$ and $t < T$. This property means that w_t^n is a martingale difference noise process.*

Assumption 6. *For each $s \in \mathcal{S}$ and $t < T$, $s \in \mathcal{S}$, suppose $\alpha_t^n \in [0,1]$ is $\mathcal{F}^n$-measurable and*
(i) $\sum_{n=1}^{\infty} \alpha_t^n(s) = \infty$ a.s.,
(ii) $\sum_{n=1}^{\infty} \alpha_t^n(s)^2 < \infty$ a.s.

5.1. Remarks on Simulation

Before proving the theorem, we offer some additional comments regarding the assumptions as they pertain to simulation. If H is defined in the context of (2), then it is not easy to perform Step 2a of Figure 2,

$$\hat{v}_t^n = \left(H\bar{V}^{n-1}\right)_t + w_t^n,$$

such that Assumption 5 is satisfied. Because the supremum is outside of the expectation operator, an upward bias would be present in the observation $\hat{v}_t^n(s)$ *unless* the expectation can be computed exactly, in which case $w_t^n(s) = 0$ and we have

$$\hat{v}_t^n(s) = \sup_{a \in \mathcal{A}} [C_t(s,a) + \mathbf{E}[\bar{V}_{t+1}^{n-1}(S_{t+1}) \mid S_t = s, a_t = a]]. \quad (11)$$

Thus, any approximation scheme used to calculate the expectation inside of the supremum would cause Assumption 5 to be unsatisfied. When the approximation scheme is a sample mean, the bias disappears asymptotically with the number of samples (see Kleywegt et al. 2002, which discusses the *sample average approximation* or SAA method). It is therefore possible that although theoretical convergence is not guaranteed, a large enough sample may still achieve decent results *in practice*.

On the other hand, in the context of (5) and (6), the expectation and the supremum are interchanged. This means that we can trivially obtain an unbiased estimate of $(H\bar{V}^{n-1})_t$ by sampling *one outcome* of the information process W_{t+1}^n from the distribution $W_{t+1} \mid S_t = s$; computing the next state S_{t+1}^n; and solving a deterministic optimization problem (i.e., the optimization within the expectation). In these two cases, we would respectively use

$$\hat{v}_t^n(s,a) = C_t(s,a) + \max_{a_{t+1} \in \mathcal{A}} \bar{Q}_{t+1}^{n-1}(S_{t+1}^n, a_{t+1}) \quad (12)$$

and

$$\hat{v}_t^n(s^a) = \sup_{a \in \mathcal{A}} [C_{t+1}(S_{t+1}^n, a) + \bar{V}_{t+1}^{a,n-1}(S_{t+1}^{a,n})], \quad (13)$$

where $\bar{Q}_{t+1}^{n-1}$ is the approximation to Q_{t+1}^*, $\bar{V}_t^{a,n-1}$ is the approximation to $V_t^{a,*}$, and $S_{t+1}^{a,n}$ is the post-decision state obtained from S_{t+1}^n and a. Notice that (11) contains an expectation whereas (12) and (13) do not, making them particularly well suited for model-free situations, where distributions are unknown and only samples or experience are available. Hence, the best choice of model depends heavily upon the problem domain.

Finally, we give a brief discussion of the choice of stepsize. There are a variety of ways in which we can satisfy Assumption 6, and here we offer the simplest example. Consider any deterministic sequence $\{a^n\}$ such that the usual stepsize conditions are satisfied:

$$\sum_{n=0}^{\infty} a^n = \infty \quad \text{and} \quad \sum_{n=0}^{\infty} (a^n)^2 < \infty.$$

Let $N(s,n,t) = \sum_{m=1}^{n} \mathbf{1}_{\{s = S_t^m\}}$ be the random variable representing the total number of visits of state s at time t until iteration n. Then $\alpha_t^n = a^{N(S_t^n, n, t)}$ satisfies Assumption 6.

6. Convergence Analysis of the Monotone-ADP Algorithm

We are now ready to show the convergence of the algorithm. Note that although there is a significant similarity between this algorithm and the DOME algorithm described in Papadaki and Powell (2003a), the proof technique is very different. The convergence proof for the DOME algorithm cannot be directly extended to our problem because of differences in the assumptions.

Our proof draws on proof techniques found in Tsitsiklis (1994) and Nascimento and Powell (2009). In the latter, the authors prove convergence of a purely exploitative ADP algorithm given a concave, piecewise-linear value function

for the lagged asset acquisition problem. We cannot exploit certain properties inherent to that problem, but in our algorithm we assume exploration of all states, a requirement that can be avoided when we are able to assume concavity. Furthermore, a significant difference in this proof is that we consider the case where $\mathcal{S}$ may not be a total ordering. A consequence of this is that we extend to the case where the monotonicity property covers multiple dimensions (e.g., the relation on $\mathcal{S}$ is the componentwise inequality), which was not allowed in Nascimento and Powell (2009).

THEOREM 1. *Under Assumptions 1–6, for each $t \leqslant T$ and $s \in \mathcal{S}$, the estimate $\bar{V}_t^n(s)$ produced by the Monotone-ADP Algorithm of Figure 2 converge to the optimal value function $V_t^*(s)$ almost surely.*

Before providing the proof for this convergence result, we present some preliminary definitions and results. First, we define two deterministic bounding sequences, U^k and L^k. The two sequences U^k and L^k can be thought of, jointly, as a sequence of "shrinking" rectangles, with U^k being the upper bounds and L^k being the lower bounds. The central idea to the proof is showing that the estimates $\bar{V}^n$ enter (and stay) in smaller and smaller rectangles, for a fixed $\omega \in \Omega$ (we assume that the ω does not lie in a discarded set of probability zero). We can then show that the rectangles converge to the point V^*, which in turn implies the convergence of $\bar{V}^n$ to the optimal value function. This idea is attributed to Tsitsiklis (1994) and is illustrated in Figure 3.

The sequences U^k and L^k are written recursively. Let

$$U^0 = V^* + V_{\max} \cdot e, \quad L^0 = V^* - V_{\max} \cdot e, \tag{14}$$

and let

$$U^{k+1} = \frac{U^k + HU^k}{2}, \quad L^{k+1} = \frac{L^k + HL^k}{2}.$$

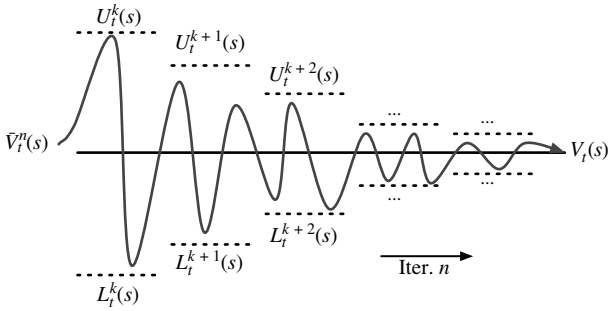
LEMMA 2. *For all $k \geqslant 0$, we have that*

$$HU^k \leqslant U^{k+1} \leqslant U^k, \quad HL^k \geqslant L^{k+1} \geqslant L^k.$$

Furthermore,

$$U^k \longrightarrow V^*, \quad L^k \longrightarrow V^*. \tag{15}$$

Figure 3. Central idea of convergence proof.



PROOF. The proof of this lemma is given in Bertsekas and Tsitsiklis (1996) (see Lemmas 4.5 and 4.6). The properties of H given in Proposition 1 are used for this result.

LEMMA 3. *The bounding sequences satisfy the monotonicity property; that is, for $k \geqslant 0$, $t \leqslant T$, $s \in \mathcal{S}$, $s' \in \mathcal{S}$ such that $s \preceq s'$, we have*

$$U_t^k(s) \leqslant U_t^k(s'), \quad L_t^k(s) \leqslant L_t^k(s').$$

PROOF. See Appendix C.

We continue with some definitions pertaining to the projection operator Π_M. A "$-$" in the superscript signifies "the value s is too small" and the "$+$" signifies "the value of s is too large."

DEFINITION 2. For $t < T$ and $s \in \mathcal{S}$, let $\mathcal{N}_t^-(s)$ be a random set representing the iterations for which s was increased by the projection operator at time t. Similarly, let $\mathcal{N}_t^+(s)$ represent the iterations for which s was decreased:

$$\mathcal{N}_t^{\Pi-}(s) = \{n\colon s \neq S_t^n \text{ and } \bar{V}_t^{n-1}(s) < \bar{V}_t^n(s)\},$$
$$\mathcal{N}_t^{\Pi+}(s) = \{n\colon s \neq S_t^n \text{ and } \bar{V}_t^{n-1}(s) > \bar{V}_t^n(s)\}.$$

DEFINITION 3. For $t < T$ and $s \in \mathcal{S}$, let $N^{\Pi-}(t, s)$ be the last iteration for which the state s was increased by Π_M at time t.

$$N_t^{\Pi-}(s) = \max \mathcal{N}_t^-(s).$$

Similarly, let

$$N_t^{\Pi+}(s) = \max \mathcal{N}_t^+(s).$$

Note that $N_t^{\Pi-}(s) = \infty$ if $|\mathcal{N}_t^-(s)| = \infty$ and $N_t^{\Pi+}(s) = \infty$ if $|\mathcal{N}_t^+(s)| = \infty$.

DEFINITION 4. Let N^Π be large enough so that for iterations $n \geqslant N^\Pi$, any state increased (decreased) finitely often by the projection operator Π_M is no longer affected by Π_M. In other words, if some state is increased (decreased) by Π_M on an iteration after N^Π, then that state is increased (decreased) by Π_M infinitely often. We can write the following:

$$N^\Pi = \max(\{N_t^{\Pi-}(s)\colon t < T,\ s \in \mathcal{S},\ N_t^{\Pi-}(s) < \infty\} \cup \{N_t^{\Pi+}(s)\colon t < T,\ s \in \mathcal{S},\ N_t^{\Pi+}(s) < \infty\}) + 1.$$

We now define, for each t, two random subsets $\mathcal{S}_t^-$ and $\mathcal{S}_t^+$ of the state space $\mathcal{S}$ where $\mathcal{S}_t^-$ contains states that are increased by the projection operator Π_M finitely often and $\mathcal{S}_t^+$ contains states that are decreased by the projection operator finitely often. The role that these two sets play in the proof is as follows:

- We first show convergence for states that are projected finitely often ($s \in \mathcal{S}_t^-$ or $s \in \mathcal{S}_t^+$).

• Next, because convergence already holds for states that are projected finitely often, we use an induction-like argument to extend the property to states that are projected infinitely often ($s \in \mathcal{S}\backslash\mathcal{S}_t^-$ or $s \in \mathcal{S}\backslash\mathcal{S}_t^+$). This step requires the definition of a tree structure that arranges the set of states and its partial ordering in an intuitive way.

Definition 5. For $t < T$, define

$$\mathcal{S}_t^- = \{s \in \mathcal{S}\colon N_t^{\Pi-}(s) < \infty\} \quad \text{and}$$
$$\mathcal{S}_t^+ = \{s \in \mathcal{S}\colon N_t^{\Pi+}(s) < \infty\},$$

to be random subsets of states that are projected finitely often.

Lemma 4. *The random sets $\mathcal{S}_t^-$ and $\mathcal{S}_t^+$ are almost surely nonempty.*

Proof. See Appendix D.

We now provide several remarks regarding the projection operator Π_M. The value of a state s can only be increased by Π_M if we visit a "smaller" state; i.e., $S_t^n \preceq s$. This statement is obvious from the second condition of (8). Similarly, the value of the state can only be decreased by Π_M if the visited state is "larger"; i.e., $S_t^n \succeq s$. Intuitively, it can be useful to imagine that, in some sense, the values of states can be "pushed up" from the "left" and "pushed down" from the "right."

Finally, because of our assumption that $\mathcal{S}$ is only a partial ordering, the update process (from Π_M) becomes more difficult to analyze than in the total ordering case. To facilitate the analysis of the process, we introduce the notions of *lower (upper) immediate neighbors* and *lower (upper) update trees*.

Definition 6. For $s = (m, i) \in \mathcal{S}$, we define the set of *lower immediate neighbors* $\mathcal{S}_L(s)$ in the following way:

$$\mathcal{S}_L(s) = \{s' \in \mathcal{S}\colon \; s' \preceq s,\; s' \neq s,\; \nexists\, s'' \in \mathcal{S},\, s'' \neq s,\, s'' \neq s',\, s' \preceq s'' \preceq s\}.$$

In other words, there does not exist s'' in between s' and s. The set of *upper immediate neighbors* $\mathcal{S}_U(s)$ is defined in a similar way:

$$\mathcal{S}_U(s) = \{s' \in \mathcal{S}\colon \; s' \succeq s,\; s' \neq s,\; \nexists\, s'' \in \mathcal{S},\, s'' \neq s,\, s'' \neq s',\, s' \succeq s'' \succeq s\}.$$

The intuition for the next lemma is that if some state s is increased by Π_M, then it must have been caused by visiting a lower state. In particular, *either* the visited state was one of the lower immediate neighbors *or* one of the lower immediate neighbors was also increased by Π_M. In either case, one of the lower immediate neighbors has the same value as s. This lemma is crucial later in the proof.

Lemma 5. *Suppose the value of s is increased by Π_M on some iteration n: $s \neq S_t^n$ and $\bar{V}_t^{n-1}(s) < \bar{V}_t^n(s)$. Then there exists another state $s' \in \mathcal{S}_L(s)$ (in the set of lower immediate neighbors) whose value is equal to the newly updated value: $\bar{V}_t^n(s') = \bar{V}_t^n(s)$.*

Proof. See Appendix E.

Definition 7. Consider some $\omega \in \Omega$. Let $s \in \mathcal{S}\backslash\mathcal{S}_t^-$, meaning that s is increased by Π_M infinitely often: $|\mathcal{N}_t^-(s)| = \infty$. A *lower update tree* $T_t^-(s)$ is an organization of the states in the set $\mathcal{L} = \{s' \in \mathcal{S}\colon s' \preceq s\}$ where the value of each node is an element of $\mathcal{L}$. The tree $T_t^-(s)$ is constructed according to the following rules.

(i) The root node of $T_t^-(s)$ has value s.

(ii) Consider an arbitrary node j with value s_j.

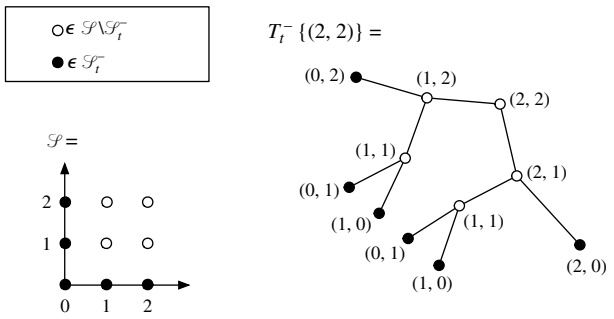
(a) If $s_j \in \mathcal{S}\backslash\mathcal{S}_t^-$, then for each $s_{jc} \in \mathcal{S}_L(s_j)$, add a child node with value s_{jc} to the node j.

(b) If $s_j \in \mathcal{S}_t^-$, then j is a leaf node (it does not have any child nodes).

The tree $T_t^-(s)$ is unique and can easily be built by starting with the root node and successively applying the rules. The *upper update tree* $T_t^+(s)$ is defined in a completely analogous way.

Note that the lower update tree is random and we now argue that for each ω, it is well defined. We observe that it cannot be the case for some state s to be an element of $\mathcal{S}\backslash\mathcal{S}_t^-$ while $\mathcal{S}_L(s') = \{\}$ because for it to be increased infinitely often, there must exist at least one "lower" state whose observations cause the monotonicity violations. Using this fact along with the finiteness of $\mathcal{S}$ and Lemma 4, which states that $\mathcal{S}_t^-$ is nonempty, it is clear that all paths down the tree reach a leaf node (i.e., an element of $\mathcal{S}_t^-$). The reason for discontinuing the tree at states in $\mathcal{S}_t^-$ is that our convergence proof employs an induction-like argument up the tree, starting with states in $\mathcal{S}_t^-$. Lastly, we remark that it is possible for multiple nodes to have the same value. As an illustrative example, consider the case with $\mathcal{S} = \{0, 1, 2\}^2$ with $\preceq$ being the component-wise inequality. Assume that for a particular $\omega \in \Omega$, $s = (s_x, s_y) \in \mathcal{S}_t^-$ if and only if $s_x = 0$ or $s_y = 0$ (lower boundary of the square). Figure 4 shows the realization of the lower update tree at evaluated at the state $(2, 2)$.

Figure 4. Illustration of the lower update tree.



The next lemma is a useful technical result used in the convergence proof.

LEMMA 6. *For any* $s \in \mathcal{S}$,

$$\lim_{m\to\infty}\left[\prod_{n=1}^{m}(1-\alpha_t^n(s))\right]=0 \quad \text{a.s.}$$

PROOF. See Appendix F.

With these preliminaries in mind (other elements will be defined as they arise), we begin the convergence analysis.

PROOF OF THEOREM 1. As previously mentioned, to show that the sequence $\bar{V}_t^n(s)$ (almost surely) converges to $V_t^*(s)$ for each t and s, we need to argue that $\bar{V}_t^n(s)$ eventually enters every rectangle (or "interval," when we discuss a specific component of the vector $\bar{V}^n$) defined by the sequence $L_t^k(s)$ and $U_t^k(s)$. Recall that the estimates of the value function produced by the algorithm are indexed by n and the bounding rectangles are indexed by k. Hence, we aim to show that for each k, we have that for n sufficiently large, it is true that $\forall s \in \mathcal{S}$,

$$L_t^k(s) \leqslant \bar{V}_t^n(s) \leqslant U_t^k(s). \tag{16}$$

Following this step, an application of (15) in Lemma 2 completes the proof. We show the second inequality of (16) and remark that the first can be shown in a completely symmetric way. The goal is then to show that $\exists N_t^k < \infty$ a.s. such that $\forall n \geqslant N_t^k$ and $\forall s \in \mathcal{S}$,

$$\bar{V}_t^n(s) \leqslant U_t^k(s). \tag{17}$$

Choose $\omega \in \Omega$. For ease of presentation, the dependence of the random variables on ω is omitted. We use backward induction on t to show this result, which is the same technique used in Nascimento and Powell (2009). The inductive step is broken up into two cases, $s \in \mathcal{S}_t^-$ and $s \in \mathcal{S}\backslash\mathcal{S}_t^-$.

Base case, $t = T$. Since for all $s \in \mathcal{S}$, k, and n, we have that (by definition) $\bar{V}_T^n(s) = U_T^k(s) = 0$, we can arbitrarily select N_T^k. Suppose that for each k, we choose $N_T^k = N^\Pi$, allowing us to use the property of N^Π that if $s \in \mathcal{S}_t^-$, then the estimate of the value at s is no longer affected by Π_M on iterations $n \geqslant N^\Pi$.

Induction hypothesis, $t+1$. Assume for $t+1 \leqslant T$ that $\forall k \geqslant 0$, $\exists N_{t+1}^k < \infty$ such that $N_{t+1}^k \geqslant N^\Pi$ and $\forall n \geqslant N_{t+1}^k$, we have that $\forall s \in \mathcal{S}$, $\bar{V}_{t+1}^n(s) \leqslant U_{t+1}^k(s)$.

Inductive step from $t+1$ *to* t. The remainder of the proof concerns this inductive step and is broken up into two cases, $s \in \mathcal{S}_t^-$ and $s \in \mathcal{S}\backslash\mathcal{S}_t^-$. For each s, we show the existence of a state dependent iteration $\tilde{N}_t^k(s) \geqslant N^\Pi$, such that for $n \geqslant \tilde{N}_t^k(s)$, (17) holds. The *state independent* iteration N_t^k is then taken to be the maximum of $\tilde{N}_t^k(s)$ over s.

Case 1: $s \in \mathcal{S}_t^-$. To prove this case, we induct forward on k. Note that we are still inducting backward on t, so the induction hypothesis for $t+1$ still holds. The inductive step is proved in essentially the same manner as Theorem 2 of Tsitsiklis (1994).

Base case, $k = 0$ (*within induction on* t). By Assumption 3 and (14), we have that $U_t^0(s) \geqslant V_{\max}$. But by Assumption 4, the updating equation (Step 2b of Figure 2), and the initialization of $\bar{V}_t^0(s) \in [0, V_{\max}]$, we can easily see that $\bar{V}_t^n(s) \in [0, V_{\max}]$ for any n and s. Therefore, $\bar{V}_t^n(s) \leqslant U_t^0(s)$, for any n and s, so we can choose N_t^0 arbitrarily. Let us choose $\tilde{N}_t^0(s) = N_{t+1}^0$, and since N_{t+1}^0 came from the induction hypothesis for $t+1$, it is also true that $\tilde{N}_t^0(s) \geqslant N^\Pi$.

Induction hypothesis, k (*within induction on* t). Assume for $k \geqslant 0$ that $\exists \tilde{N}_t^k(s) < \infty$ such that $\tilde{N}_t^k(s) \geqslant N_{t+1}^k \geqslant N^\Pi$ and $\forall n \geqslant \tilde{N}_t^k(s)$, we have $\bar{V}_t^n(s) \leqslant U_t^k(s)$.

Before we begin the inductive step from k to $k+1$, we define some additional sequences and state a few useful lemmas.

DEFINITION 8. The *positive incurred noise*, since a starting iteration m, is represented by the sequence $W_t^{n,m}(s)$. For $s \in \mathcal{S}$, it is defined as follows:

$$W_t^{m,m}(s) = 0,$$
$$W_t^{n+1,m}(s) = [(1-\alpha_t^n(s))W_t^{n,m}(s) + \alpha_t^n(s)w_t^{n+1}(s)]^+ \quad \text{for } n \geqslant m.$$

The term $W_t^{n+1,m}(s)$ is only updated from $W_t^{n,m}(s)$ when $s = S_t^n$, i.e., on iterations where the state is visited by the algorithm, because the stepsize $\alpha_t^n(s) = 0$ whenever $s \neq S_t^n$.

LEMMA 7. *For any starting iteration* $m \geqslant 0$ *and any state* $s \in \mathcal{S}$, *under Assumptions* 4, 5, *and* 6, $W_t^{n,m}(s)$ *asymptotically vanishes*:

$$\lim_{n\to\infty} W_t^{n,m}(s) = 0 \quad \text{a.s.}$$

PROOF. The proof is analogous to that of Lemma 6.2 in Nascimento and Powell (2009), which uses a martingale convergence argument.

To reemphasize the presence of ω, we note that the following definition and the subsequent lemma both use the realization $\tilde{N}_t^k(s)(\omega)$ from the ω chosen at the beginning of the proof.

DEFINITION 9. The other auxiliary sequence that we need is $X_t^n(s)$, which applies the smoothing step to $(HU^k)_t(s)$. For any state $s \in \mathcal{S}$, let

$$X_t^{\tilde{N}_t^k(s)}(s) = U_t^k(s),$$
$$X_t^{n+1}(s) = (1-\alpha_t^n(s))X_t^n(s) + \alpha_t^n(s)(HU^k)_t(s) \quad \text{for } n \geqslant \tilde{N}_t^k(s).$$

LEMMA 8. *For* $n \geqslant \tilde{N}_t^k(s)$ *and state* $s \in \mathcal{S}_t^-$,

$$\bar{V}_t^n(s) \leqslant X_t^n(s) + W_t^{n,\tilde{N}_t^k(s)}(s).$$

PROOF. See Appendix G.

Inductive step from k to $k+1$. If $U_t^k(s) = (HU^k)_t(s)$, then by Lemma 2, we see that $U_t^k(s) = U_t^{k+1}(s)$ so $\bar{V}_t^n \leqslant U_t^k(s) \leqslant U_t^{k+1}(s)$ for any $n \geqslant \tilde{N}_t^k(s)$ and the proof is complete. Since we know that $(HU^k)_t(s) \leqslant U_t^k(s)$ by Lemma 2, we can now assume that $s \in \mathcal{K}$, where

$$\mathcal{K} = \{s' \in \mathcal{S}\colon (HU^k)_t(s') < U_t^k(s')\}.$$

In this case, we can define

$$\delta_t^k = \min_{s \in \mathcal{S}_t^- \cap \mathcal{K}} \left(\frac{U_t^k(s) - (HU^k)_t(s)}{4} \right) > 0.$$

Choose $\tilde{N}_t^{k+1}(s) \geqslant \tilde{N}_t^k(s)$ such that

$$\prod_{n=\tilde{N}_t^k(s)}^{\tilde{N}_t^{k+1}(s)-1} (1 - \alpha_t^n(s)) \leqslant \frac{1}{4}$$

and for all $n \geqslant \tilde{N}_t^{k+1}(s)$,

$$W_t^{n,\tilde{N}_t^k(s)}(s) \leqslant \delta_t^k.$$

Note that $\tilde{N}_t^{k+1}(s)$ clearly exists because both sequences converge to zero, by Lemma 6 and 7. Recursively using the definition of $X_t^n(s)$, we get that

$$X_t^n(s) = \beta_t^n(s) U_t^k(s) + (1 - \beta_t^n(s))(HU^k)_t(s),$$

where $\beta_t^n(s) = \prod_{l=\tilde{N}_t^k(s)}^{n-1}(1 - \alpha_t^l(s))$. Notice that for $n \geqslant \tilde{N}_t^{k+1}(s)$, we know that $\beta_t^n(s) \leqslant \frac{1}{4}$, so we can write

$$\begin{aligned} X_t^n(s) &= \beta_t^n(s)\, U_t^k(s) + (1 - \beta_t^n(s))(HU^k)_t(s) \\ &= \beta_t^n(s)[U_t^k(s) - (HU^k)_t(s)] + (HU^k)_t(s) \\ &\leqslant \tfrac{1}{4} U_t^k(s) + \tfrac{3}{4}(HU^k)_t(s) \\ &= \tfrac{1}{2}[U_t^k(s) + (HU^k)_t(s)] - \tfrac{1}{4}[U_t^k(s) - (HU^k)_t(s)] \\ &\leqslant U_t^{k+1}(s) - \delta_t^k. \end{aligned} \tag{18}$$

We can apply Lemma 8 and (18) to get

$$\begin{aligned} \bar{V}_t^n(s) &\leqslant X_t^n(s) + W_t^{n,\tilde{N}_t^k(s)}(s) \\ &\leqslant (U_t^{k+1}(s) - \delta_t^k) + \delta_t^k \\ &= U_t^{k+1}(s), \end{aligned}$$

for all $n \geqslant \tilde{N}_t^{k+1}(s)$. Thus, the inductive step from k to $k+1$ is complete.

Case 2: $s \in \mathcal{S}\backslash\mathcal{S}_t^-$. Recall that we are still in the inductive step from $t+1$ to t (where the hypothesis was the existence of N_{t+1}^k). As previously mentioned, the proof for this case relies on an induction-like argument over the tree $T_t^-(s)$. The following lemma is the core of our argument, and the proof is provided below.

Lemma 9. *Consider some $k \geqslant 0$ and a node j of $T_t^-(s)$ with value $s_j \in \mathcal{S}\backslash\mathcal{S}_t^-$ and let the $C_j \geqslant 1$ child nodes of j be denoted by the set $\{s_{j,1}, s_{j,2}, \ldots, s_{j,C_j}\}$. Suppose that for each $s_{j,c}$ where $1 \leqslant c \leqslant C_j$, we have that $\exists\, \tilde{N}_t^k(s_{j,c}) < \infty$ such that $\forall\, n \geqslant \tilde{N}_t^k(s_{j,c})$,*

$$\bar{V}_t^n(s_{j,c}) \leqslant U_t^k(s_{j,c}). \tag{19}$$

Then $\exists\, \tilde{N}_t^k(s_j) < \infty$ such that $\forall\, n \geqslant \tilde{N}_t^k(s_j)$,

$$\bar{V}_t^n(s_j) \leqslant U_t^k(s_j).$$

Proof. First, note that by the induction hypothesis, part (ii) of Lemmas 1, and 2, we have the inequality

$$(H\bar{V}^n)_t(s) \leqslant (HU^k)_t(s) \leqslant U_t^k(s). \tag{20}$$

We break the proof into several steps.

Step 1. Let us consider the iteration $\tilde{N}$ defined by

$$\tilde{N} = \min(n \in \mathcal{N}_t^{\Pi-}(s_j)\colon n \geqslant \max_c \tilde{N}_t^k(s_{j,c})),$$

which exists because $s_j \in \mathcal{S}\backslash\mathcal{S}_t^-$ and is increased infinitely often. This means that Π_M increased the value of state s_j on iteration $\tilde{N}$. As the first step, we show that $\forall\, n \geqslant \tilde{N}$,

$$\bar{V}_t^n(s_j) \leqslant U_t^k(s_j) + W_t^{n,\tilde{N}}(s_j), \tag{21}$$

using an induction argument.

Base case, $n = \tilde{N}$. Using Lemma 5, we know that for some $c \in \{1, 2, \ldots, C_j\}$, we have

$$\begin{aligned} \bar{V}_t^n(s_j) = \bar{V}_t^n(s_{j,c}) &\leqslant U_t^k(s_{j,c}) \\ &\leqslant U_t^k(s_j) + W_t^{n,\tilde{N}}(s_j). \end{aligned}$$

The fact that $\tilde{N} \geqslant \tilde{N}_t^k(s_{j,c})$ for every c justifies the first inequality and the second inequality above follows from the monotonicity within U^k (see Lemma 3) and that $W_t^{\tilde{N},\tilde{N}}(s_j) = 0$.

Induction hypothesis, n. Suppose (21) is true for n where $n \geqslant \tilde{N}$.

Inductive step from n to $n+1$. Consider the following two cases:

(I) Suppose $n+1 \in \mathcal{N}_t^{\Pi-}(s_j)$. The proof for this is exactly the same as for the base case, except we use $W_t^{n+1,\tilde{N}}(s_j) \geqslant 0$ to show the inequality. Again, this step depends heavily on Lemma 5 and on every child node representing a state that satisfies (19).

(II) Suppose $n+1 \notin \mathcal{N}_t^{\Pi-}(s_j)$. There are again two cases to consider:

(A) Suppose $S_t^{n+1} = s_j$. Then

$$\begin{aligned} \bar{V}_t^{n+1}(s_j) &= z_t^{n+1}(s_j) \\ &= (1 - \alpha_t^{n+1}(s_j))\bar{V}_t^n(s_j) + \alpha_t^{n+1}(s_j)\hat{v}_t^{n+1}(s_j) \\ &\leqslant (1 - \alpha_t^{n+1}(s_j))(U_t^k(s_j) + W_t^{n,\tilde{N}}(s_j)) \\ &\quad + \alpha_t^{n+1}(s_j)[(H\bar{V}^n)_t(s_j) + w_t^{n+1}(s_j)] \\ &\leqslant U_t^k(s_j) + W_t^{n+1,\tilde{N}}(s_j), \end{aligned}$$

where the first inequality follows from the induction hypothesis for n and the second inequality follows by (20).

(B) Suppose $S_t^{n+1} \neq s_j$. This means the stepsize $\alpha_t^{n+1}(s_j) = 0$, which in turn implies the noise sequence remains unchanged: $W_t^{n+1,\tilde{N}}(s_j) = W_t^{n,\tilde{N}}(s_j)$. Because the value of s_j is not increased at $n+1$,

$$\begin{aligned}\bar{V}_t^{n+1}(s_j) &\leqslant \bar{V}_t^n(s_j)\\ &\leqslant U_t^k(s_j) + W_t^{n,\tilde{N}}(s_j)\\ &= U_t^k(s_j) + W_t^{n+1,\tilde{N}}(s_j).\end{aligned}$$

Step 2. By Lemma 7, we know that $W_t^{n,\tilde{N}}(s_j) \to 0$ and thus, for a given $\epsilon > 0$, $\exists \tilde{N}_t^{k,\epsilon}(s_j) < \infty$ such that $\forall n \geqslant \tilde{N}_t^{k,\epsilon}(s_j)$,

$$\bar{V}_t^n(s_j) \leqslant U_t^k(s_j) + \epsilon. \tag{22}$$

Let $\epsilon = U_t^k(s_j) - V_t^*(s_j) > 0$. Since $U_t^k(s_j) \searrow V_t^*(s_j)$, we also have that $\exists k' > k$ such that

$$U_t^{k'}(s_j) - V_t^*(s_j) < \epsilon/2.$$

Combining with the definition of ϵ, we have

$$U_t^k(s_j) - U_t^{k'}(s_j) > \epsilon/2.$$

Applying (22), we know that $\exists \tilde{N}_t^{k',\epsilon/2}(s_j) < \infty$ such that $\forall n \geqslant \tilde{N}_t^{k',\epsilon/2}(s_j)$,

$$\begin{aligned}\bar{V}_t^n(s_j) &\leqslant U_t^{k'}(s_j) + \epsilon/2\\ &\leqslant U_t^k(s_j) - \epsilon/2 + \epsilon/2\\ &\leqslant U_t^k(s_j).\end{aligned}$$

Therefore, we can choose $\tilde{N}_t^k(s_j) = N_t^{k',\epsilon/2}(s_j)$ to conclude the proof.

We now present a simple algorithm that incorporates the use of Lemma 9 to obtain $\tilde{N}_t^k(s)$ when $s \in \mathcal{S}\backslash\mathcal{S}_t^-$. Denote the *height* (longest path from root to leaf) of $T_t^-(s)$ by $H_t^-(s)$. The *depth* of a node j is the length of the path from the root node to j.

Step 0. Set $h = H_t^-(s) - 1$. The child nodes of any node of depth h are leaf nodes that represent states in $\mathcal{S}_t^-$—the conditions of Lemma 9 are thus satisfied by *Case 1*.

Step 1. Consider all nodes j_h (with value s_h) of depth h in $T_t^-(s)$. An application of Lemma 9 results in $\tilde{N}_t^k(s_h)$ such that $\bar{V}_t^n(s_h) \leqslant U_t^k(s_h)$ for all $n \geqslant \tilde{N}_t^k(s_h)$.

Step 2. If $h = 0$, we are done. Otherwise, decrement h and note that once again, the conditions of Lemma 9 are satisfied for any node of depth h. Return to *Step* 1.

At the completion of this algorithm, we have the desired $\tilde{N}_t^k(s)$ for $s \in \mathcal{S}\backslash\mathcal{S}_t^-$. By its construction, we see that $\tilde{N}_t^k(s) \geqslant N^{\Pi}$, and the final step of the inductive step from $t+1$ to t is to define $N_t^k = \max_{s\in\mathcal{S}} \tilde{N}_t^k(s)$. The proof is complete.

7. Numerical Results

Theoretically, we have shown that Monotone-ADP is an asymptotically convergent algorithm. In this section, we discuss its empirical performance. There are two main questions that we aim to answer using numerical examples:

1. How much does the monotonicity preservation operator, Π_M, increase the rate of convergence, compared to other popular approximate dynamic programming or reinforcement learning algorithms?

2. How much computation can we save by solving a problem to near-optimality using Monotone-ADP compared to solving it to full optimality using backward dynamic programming?

To provide insight into these questions, we compare Monotone-ADP against four ADP algorithms from the literature (kernel-based reinforcement learning, approximate policy iteration with polynomial basis functions, asynchronous value iteration, and Q-learning) across a set of fully benchmarked problems from three distinct applications (optimal stopping, energy allocation/storage, and glycemic control for diabetes patients). Throughout our numerical work, we assume that the model is known and thus compute the observations $\hat{v}_t^n$ using (11). For Monotone-ADP, asynchronous value iteration, and Q-learning, the sampling policy we use is the ε-greedy exploration policy (explore with probability ε; follow current policy otherwise)—we found that across our range of problems, choosing a relatively large ε (e.g., $\varepsilon \in [0.5, 1]$) generally produced the best results. These same three algorithms also require the use of a stepsize, and in all cases we use the adaptive stepsize derived in George and Powell (2006). Moreover, note that since we are interested in comparing the performance of approximation algorithms against optimal benchmarks, it is necessary to sacrifice a bit of realism and discretize the distribution of W_{t+1} so that an exact optimal solution can be obtained using backward dynamic programming (BDP). However, this certainly is not necessary in practice; Monotone-ADP handles continuous distributions of W_{t+1} perfectly well, especially if (12) and (13) are used.

Before moving on to the applications, let us briefly introduce each of the algorithms that we use in the numerical work. For succinctness, we omit step-by-step descriptions of the algorithms and instead point the reader to external references.

Kernel-Based Reinforcement Learning (KBRL)

Kernel regression, which dates back to Nadaraya (1964), is arguably the most widely used nonparametric technique in statistics. Ormoneit and Sen (2002) develop this powerful idea in the context of approximating value functions using an approximate value iteration algorithm—essentially, the Bellman operator is replaced with a kernel-based approximate Bellman operator. For our finite-horizon case, the algorithm works backwards from $t = T - 1$ to produce

kernel-based approximations of the value function at each t, using a fixed-size batch of observations each time. The most attractive feature of this algorithm is that no structure whatsoever needs to be known about the value function, and in general, a decent policy can be found. One major drawback, however, is that KBRL cannot be implemented in a recursive way; the number of observations used per time period needs to be specified beforehand, and if the resulting policy is poor, then KBRL needs to completely start over with a larger number of observations. A second major drawback is *bandwidth selection*—in our empirical work, the majority of the time spent with this algorithm was focused on tuning the bandwidth, with guidance from various "rules of thumb," such as the one found in Scott (2009). Our implementation uses the popular *Gaussian kernel*, given by

$$K(s, s') = \frac{1}{2\pi} \exp\left(\frac{\|s - s'\|_2^2}{2b} \right),$$

where s and s' are two states and b the bandwidth (tuned separately for each problem). The detailed description of the algorithm can be found in the original paper, Ormoneit and Sen (2002).

Approximate Policy Iteration with Polynomial Basis Functions (API)

Based on the exact policy iteration algorithm (which is well known to possess finite time convergence), certain forms of approximate policy iteration have been applied successfully in a number of real applications (Bertsekas 2011 provides an excellent survey). The basis functions that we employ are all possible monomials up to degree 2 over all dimensions of the state variable (i.e., we allow interactions between every pair of dimensions). Because there are typically a small number of basis functions, policies produced by API are very fast to evaluate, regardless of the size of the state space. The exact implementation of our API algorithm, specialized for finite-horizon problems, is given in Powell (2011, §10.5). One drawback of API that we observed is an inadequate exploration of the state space for certain problems; even if the initial state S_0 is fully randomized, the coverage of the state space in a much later time period, $t' \gg 0$, may be sparse. To combat this issue, we introduced artificial exploration to time periods t' with poor coverage by adding simulations that started at t' rather than 0. The second drawback is that we observed what we believe to be *policy oscillations*, where the policies go from good to poor in an oscillating manner. This issue is not well understood in the literature but is discussed briefly in Bertsekas (2011).

Asynchronous Value Iteration (AVI)

This algorithm is an elementary algorithm of approximate dynamic programming/reinforcement learning. As the basis for Monotone-ADP, we include it in our comparisons to illustrate the utility of Π_M. In fact, AVI can be recovered from Monotone-ADP by simply removing the monotonicity preservation step; it is a lookup table based algorithm where only one state (per time period) is updated at every iteration. More details can be found in Sutton and Barto (1998, §4.5), Bertsekas (2007, §2.6), or Powell (2011, §10.2).

Q-Learning (QL)

Due to Watkins and Dayan (1992), this reinforcement-learning algorithm estimates the values of state-action pairs, rather than just the state, to handle the model-free situation. Its crucial drawback, however, is it can only be applied in problems with very small action spaces—for this reason, we only show results for Q-learning in the context of our optimal stopping application, where the size of the action space is two. To make the comparison between Q-learning and the algorithms as fair as possible, we improve the performance of Q-learning by taking advantage of our *known model* and compute the expectation at every iteration (as we do in AVI). This slight change from the original formulation, of course, does not alter its convergence properties.

Backward Dynamic Programming (BDP)

This is the well known, standard procedure for solving finite-horizon MDPs. Using significant computational resources, we employ BDP to obtain the optimal benchmarks in each of the example applications in this section. A description of this algorithm, which is also known as *backward induction* or *finite-horizon value iteration*, can be found in Puterman (1994).

7.1. Evaluating Policies

We follow the method in Powell (2011) for evaluating the policies generated by the algorithms. By the principle of dynamic programming, for particular value functions $V \in \mathbb{R}^D$, the decision function at time t, A_t: $\mathcal{S} \to \mathcal{A}$ can be written as

$$A_t(s) = \arg\max_{a \in \mathcal{A}} [C_t(s, a) + \mathbf{E}[V_{t+1}(S_{t+1}) \mid S_t = s,\ a_t = a]].$$

To evaluate the *value of a policy* (i.e., the expected contribution over the time horizon) using simulation, we take a test set of $L = 1{,}000$ sample paths, denoted $\hat{\Omega}$, compute the contribution for each $\omega \in \hat{\Omega}$ and take the empirical mean:

$$F(V, \omega) = \sum_{t=0}^{T} C_t(S_t(\omega), A_t(S_t(\omega))) \quad \text{and}$$

$$\bar{F}(V) = L^{-1} \sum_{\omega \in \hat{\Omega}} F(V, \omega).$$

For each problem instance, we compute the optimal policy using backward dynamic programming. We then compare the performance of approximate policies generated by each

of the above approximation algorithms to that of the optimal policy (given by $V_0^*(S_0)$), as a percentage of optimality.

We remark that although a complete numerical study is not the main focus of this paper, the results below do indeed show that Monotone-ADP provides benefits in each of these nontrivial problem settings.

7.2. Regenerative Optimal Stopping

We now present a classical example application from the fields of operations research, mathematics, and economics: regenerative optimal stopping (also known as *optimal replacement* or *optimal maintenance*; see Pierskalla and Voelker 1976 for a review). The optimal stopping model described in this paper is inspired by that of Feldstein and Rothschild (1974), Rust (1987), and Kurt and Kharoufeh (2010) and is easily generalizable to higher dimensions. We consider the decision problem of a firm that holds a depreciating asset that it needs to either sell or replace (known formally as *replacement investment*), with the depreciation rate determined by various other economic factors (giving a multidimensional state variable). The competing forces can be summarized to be the revenue generated from the asset (i.e., production of goods, tax breaks) versus the cost of replacement and the financial penalty when the asset's value reaches zero.

Consider a depreciating asset whose value over discrete time indices $t \in \{0, 1, \ldots, T\}$ is given by a process $\{X_t\}_{t=0}^T$ where $X_t \in \mathscr{X} = \{0, 1, \ldots, X_{\max}\}$. Let $\{Y_t\}_{t=0}^T$ with $Y_t = (Y_t^i)_{i=1}^{n-1} \in \mathscr{Y}$ describe external economic factors that affect the depreciation process of the asset. Each component $Y_t^i \in \{0, 1, \ldots, Y_{\max}^i\}$ contributes to the overall depreciation of the asset. The asset's value either remains the same or decreases during each time period t. Assume that for each i, higher values of the factor Y_t^i correspond to more positive influences on the value of the asset. In other words, the probability of its value depreciating between time t and $t+1$ increases as Y_t^i decreases.

Let $S_t = (X_t, Y_t) \in \mathscr{S} = \mathscr{X} \times \mathscr{Y}$ be the n-dimensional state variable. When $X_t > 0$, we earn a revenue of P for some $P > 0$, and when the asset becomes worthless (i.e., when $X_t = 0$), we suffer a penalty of $-F$ for some $F > 0$. At each stage, we can either choose to replace the asset by taking action $a_t = 1$ for some cost $r(X_t, Y_t)$, which is nonincreasing in X_t and Y_t, or do nothing by taking action $a_t = 0$ (therefore, $\mathscr{A} = \{0, 1\}$). Note that when the asset becomes worthless, we are forced to pay the penalty F in addition to the replacement cost $r(X_t, Y_t)$. Therefore, we can specify the following contribution function:

$$C_t(S_t, a_t) = P \cdot \mathbf{1}_{\{X_t > 0\}} - F \cdot \mathbf{1}_{\{X_t = 0\}} - r(X_t, Y_t)(1 - \mathbf{1}_{\{a_t = 0\}} \mathbf{1}_{\{X_t > 0\}}).$$

Let $f^-: \mathscr{X} \times \mathscr{Y} \to [0, 1]$ be a nonincreasing function in all of its arguments. The process X_t obeys the following dynamics. If $X_t = 0$ or if $a_t = 1$, then $X_{t+1} = X_{\max}$ with probability 1 (regeneration or replacement). Otherwise, X_t decreases with probability $f^-(X_t, Y_t)$ or stays at the same level with probability $1 - f^-(X_t, Y_t)$. The transition function is written

$$X_{t+1} = (X_t \cdot \mathbf{1}_{\{U_{t+1} > f(X_t, Y_t)\}} + [X_t - \epsilon_{t+1}^X]^+ \cdot \mathbf{1}_{\{U_{t+1} \leqslant f(X_t, Y_t)\}}) \cdot \mathbf{1}_{\{a_t = 0\}} \mathbf{1}_{\{X_t > 0\}} + X_{\max}(1 - \mathbf{1}_{\{a_t = 0\}} \mathbf{1}_{\{X_t > 0\}}),$$

where U_{t+1} are i.i.d. uniform random variables over the interval $[0, 1]$ and ϵ_{t+1}^X are i.i.d. *discrete* uniform random variables over $\{1, 2, \ldots, \epsilon_{\max}\}$. The first part of the transition function covers the case where we wait ($a_t = 0$) and the asset still has value ($X_t > 0$); depending on the outcome of U_{t+1}, its value either remains at its current level or depreciates by some random amount ϵ_{t+1}^X. The second part of the formula reverts X_{t+1} to $X_{\max}$ whenever $a_t = 1$ or $X_t = 0$.

Let Y_t^i evolve stochastically such that if $a_t = 0$ and $X_t > 0$, $Y_{t+1}^i \leqslant Y_t^i$ with probability 1. Otherwise, the external factors also reset: $Y_{t+1}^i = Y_{\max}^i$:

$$Y_{t+1}^i = [Y_t^i - \epsilon_{t+1}^i]^+ \cdot \mathbf{1}_{\{a_t = 0\}} \mathbf{1}_{\{X_t > 0\}} + Y_{\max}^i \cdot (1 - \mathbf{1}_{\{a_t = 0\}} \mathbf{1}_{\{X_t > 0\}}),$$

where ϵ_{t+1}^i are i.i.d. (across i and t) Bernoulli with a fixed parameter p^i. Thus, we take the information process to be $W_{t+1} = (U_{t+1}, \epsilon_{t+1}^X, \epsilon_{t+1}^1, \epsilon_{t+1}^2, \ldots, \epsilon_{t+1}^n)$, which is independent of S_t. Moreover, we take $C_T(x, y) = 0$ for all $(x, y) \in \mathscr{X} \times \mathscr{Y}$. The following proposition establishes that Assumption 1 is satisfied.

Proposition 3. *Under the regenerative optimal stopping model, define the Bellman operator H as in (7) and let $\preceq$ be the componentwise inequality over all dimensions of the state space. Then Assumption 1 is satisfied. In particular, this implies that the optimal value function is monotone: for each $t \leqslant T$, $V_t^*(X_t, Y_t)$ is nondecreasing in both X_t and Y_t (i.e., in all n dimensions of the state variable S_t).*

Proof. See the online supplement.

7.2.1. Parameter Choices. In the numerical work of this paper, we consider five variations of the problem, where the dimension varies across $n = 3$, $n = 4$, $n = 5$, $n = 6$, and $n = 7$, and the labels assigned to these are R3, R4, R5, R6, and R7, respectively. The following set of parameters are used across all of the problem instances. We use $X_{\max} = 10$ and $Y_{\max}^i = 10$, for $i = 1, 2, \ldots, n-1$ over a finite time horizon $T = 25$. The probability of the i-th external factor Y_t^i decrementing, p_i, is set to $p_i = i/(2n)$, for $i = 1, 2, \ldots, n-1$. Moreover, the probability of the asset's value X_t depreciating is given by

$$f^-(X_t, Y_t) = 1 - \frac{X_t^2 + \|Y_t\|_2^2}{X_{\max}^2 + \|Y_{\max}\|_2^2},$$

and we use $\epsilon_{\max} = 5$. The revenue is set to be $P = 100$ and the penalty to be $F = 1{,}000$. Finally, we let the replacement cost be:

$$r(X_t, Y_t) = 400 + \frac{2}{n}(X_{\max}^2 - X_t^2 + \|Y_{\max}\|_2^2 - \|Y_t\|_2^2),$$

Table 1. Basic properties of regenerative optimal stopping problem instances.

Label	State space	Eff. state space	Action space	CPU (sec.)
R3	1,331	33,275	2	49
R4	14,641	366,025	2	325
R5	161,051	4,026,275	2	3,957
R6	1,771,561	44,289,025	2	49,360
R7	19,487,171	487,179,275	2	620,483

which ranges from 400 to 600. All of the policies that we compute assume an initial state of $S_0 = (X_{\max}, Y_{\max})$. For each of the five problem instances, Table 1 gives the cardinalities of the state space; effective state space (i.e., $(T+1)|\mathcal{S}|$); and action space along with the computation time required to solve the problem exactly using backward dynamic programming. In the case of R7, we have an effective state space of nearly half a billion, which requires more than a week of computation time to solve exactly.

7.2.2. Results. Figure 5 displays the empirical results of running Monotone-ADP and each of the aforementioned ADP/RL algorithms on the optimal stopping instances R3–R7. Because of the vast difference in size of the problems (e.g., R7 is 14,000 times larger than R3), each problem was run for a different number of iterations. First, we point out that AVI and QL barely make any progress, even in the smallest instance R3. However, this observation only partially attests to the value of the simple Π_M operator; it is not entirely surprising because AVI and QL only update one state (or one state-action) per iteration. The fact that Monotone-ADP also outperforms both KBRL and API (in the area of 10%–15%) makes a stronger case for Monotone-ADP because in both KBRL and API, there is a notion of generalization to the entire state space. As we mentioned earlier, aside from the larger optimality gap, the main concerns with KBRL and API are, respectively, bandwidth selection and policy oscillations.

Question (2) concerns the computation requirement for Monotone-ADP. The optimality of the Monotone-ADP policies versus the computation times needed for each are shown on the semilog plots in Figure 6 below. In addition, the single point to the right represents the amount of computation needed to produce the exact optimal solution using BDP. The horizontal line represents 90% optimality (near-optimal). The plots show that for every problem instance, we can reach near-optimality using Monotone-ADP with around an *order of magnitude* less computation than if we used BDP to compute the true optimal. In terms of a percentage, Monotone-ADP required 5.3%, 5.2%, 4.5%, 4.3%, and 13.1% of the optimal solution computation time to reach a near-optimal solution in each of the respective problem instances. From Table 1, we see that for larger problems, the amount of computation needed for an exact optimal policy is unreasonable for any real-world application. Combined with the fact that it is extremely easy to find examples of far more complex problems (the relatively small $X_{\max}$ and $Y^i_{\max}$ makes this example still somewhat tractable), it should be clear that exact methods are not a realistic option, even given the attractive theory of finite time convergence.

7.3. Energy Storage and Allocation

The recent surge of interest in renewable energy leads us to present a second example application in area energy storage. The goal of this specific problem is to optimize revenues while satisfying demand, in the presence of (1) a storage

Figure 5. Comparison of Monotone-ADP to other ADP/RL algorithms.

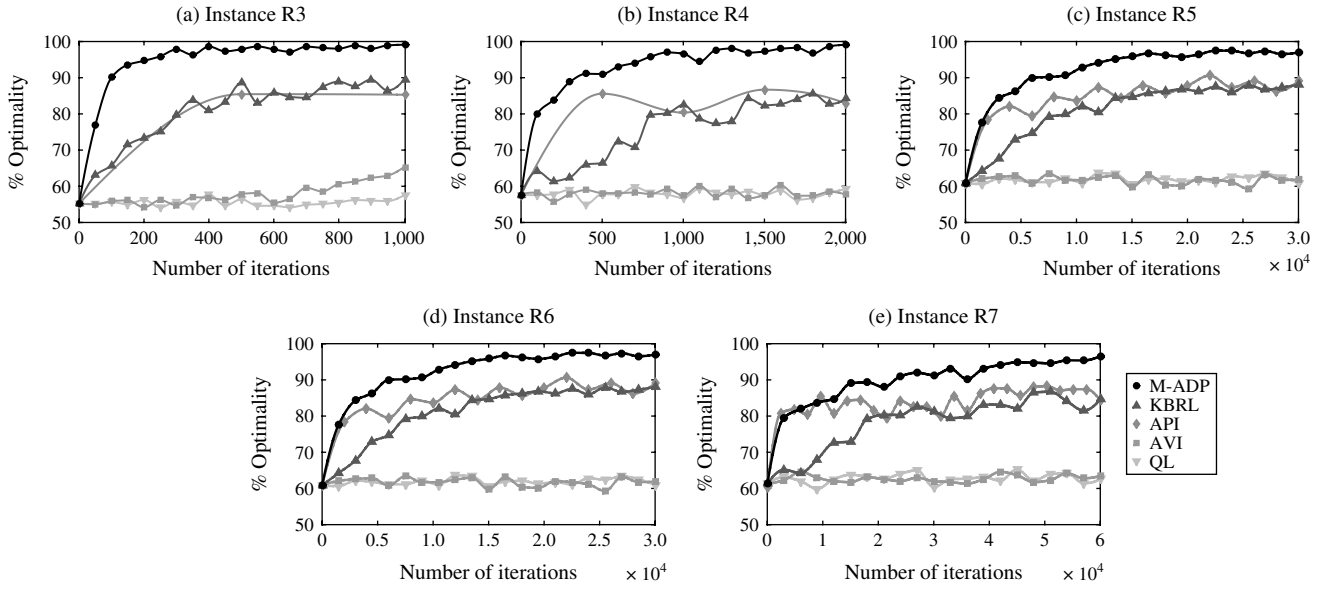
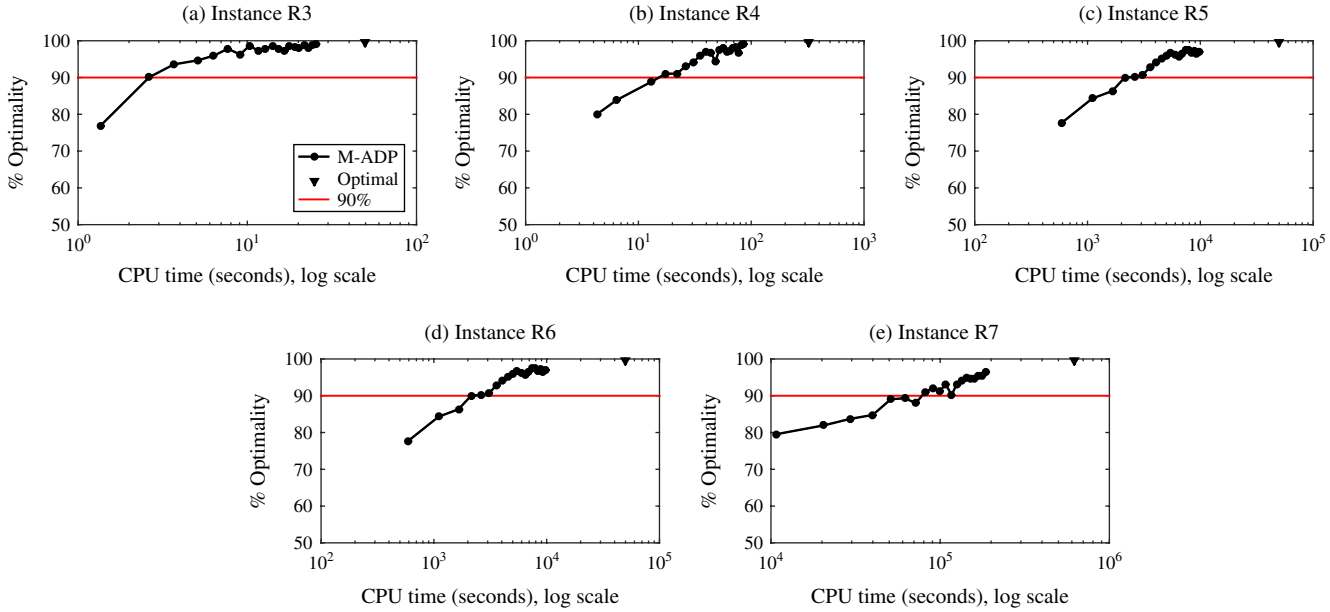


Figure 6. Computation times (seconds) of Monotone-ADP vs. backward dynamic programming.

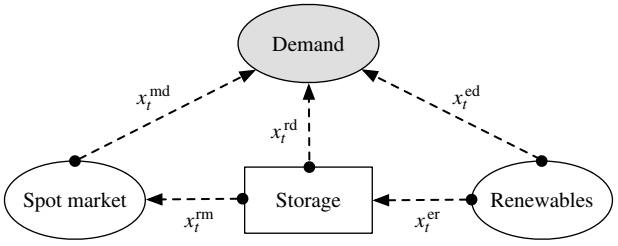


device, such as a battery, and (2) a (stochastic) renewable source of energy, such as wind or solar. Our action or decision vector is an *allocation decision*, containing five dimensions that describe how the energy is transferred within our network, consisting of nodes for the storage device, the spot market, demand, and the source of renewable generation (see Figure 7). Similar problems from the literature that share the common theme of storage include Secomandi (2010), Carmona and Ludkovski (2010), and Kim and Powell (2011), to name a few.

Let the state variable be $S_t = (R_t, E_t, P_t, D_t) \in \mathcal{S}$, where R_t is the amount of energy in storage at time t, E_t is the amount of renewable generation available at time t, P_t is the price of energy on the spot market at time t, and D_t is the demand that needs to be satisfied at time t. We define the discretized state space $\mathcal{S}$ by allowing (R_t, E_t, P_t, D_t) to take on all integral values contained within the hyperrectangle

$$[0, R_{\max}] \times [E_{\min}, E_{\max}] \times [P_{\min}, P_{\max}] \times [D_{\min}, D_{\max}],$$

Figure 7. Network diagram for the energy storage problem.



where $R_{\max} \geqslant 0$, $E_{\max} \geqslant E_{\min} \geqslant 0$, $P_{\max} \geqslant P_{\min} \geqslant 0$, and $D_{\max} \geqslant D_{\min} \geqslant 0$. Let γ_c and γ_d be the maximum rates of charge and discharge from the storage device, respectively. The decision vector is given by (refer again to Figure 7 for the meanings of the components)

$$x_t = (x_t^{ed}, x_t^{md}, x_t^{rd}, x_t^{er}, x_t^{rm})^{\mathsf{T}} \in \mathcal{X}(S_t),$$

where the feasible set $\mathcal{X}(S_t)$ is defined by $x_t \in \mathbb{N}^5$ intersected with the following, mostly intuitive, constraints:

$$(x_t^{ed}, x_t^{md}, x_t^{rd}, x_t^{er}, x_t^{rm})^{\mathsf{T}} \geqslant 0, \tag{23}$$

$$x_t^{ed} + x_t^{rd} + x_t^{md} = D_t, \tag{24}$$

$$x_t^{rd} + x_t^{rm} \leqslant R_t,$$

$$x_t^{er} + x_t^{ed} \leqslant E_t,$$

$$x_t^{rd} + x_t^{rm} \leqslant \gamma^d,$$

$$x_t^{er} \leqslant \min\{R_{\max} - R_t, \gamma_c\}. \tag{25}$$

Note that whenever the energy in storage combined with the amount of renewable generation is not enough to satisfy demand, the remainder is purchased from the spot market. The contribution function is given by

$$C_t(S_t, x_t) = P_t(D_t + x_t^{rm} - x_t^{md}).$$

To describe the transition of the state variable, first define $\phi = (0, 0, -1, 1, -1)^{\mathsf{T}}$ to be the flow coefficients for a decision x_t with respect to the storage device. We also assume that the dependence on the past for the stochastic processes $\{E_t\}_{t=0}^T$, $\{P_t\}_{t=0}^T$, and $\{D_t\}_{t=0}^T$ is at most Markov

of order one. Let $[\cdot]_b^a = \min(\max(\cdot, b), a)$. Thus, we can write the transition function for S_t using the following set of equations:

$$R_{t+1} = R_t + \phi^{\mathrm{T}} x_t \quad \text{and} \quad \begin{aligned} E_{t+1} &= [E_t + \hat{E}_{t+1}]_{E_{\min}}^{E_{\max}}, \\ P_{t+1} &= [P_t + \hat{P}_{t+1}]_{P_{\min}}^{P_{\max}}, \\ D_{t+1} &= [D_t + \hat{D}_{t+1}]_{D_{\min}}^{D_{\max}}, \end{aligned} \tag{26}$$

where the information process $W_{t+1} = (\hat{E}_{t+1}, \hat{P}_{t+1}, \hat{D}_{t+1})$ is independent of S_t and x_t (the precise processes used are given in §7.3.1). Note that the combined transition function is monotone in the sense of (i) of Proposition 1, where $\preceq$ is the componentwise inequality.

Proposition 4. *Under the energy storage model, define the Bellman operator H as in (7), with $\mathscr{A}$ replaced with the state dependent $\mathscr{X}(s)$, and let $\preceq$ be the componentwise inequality over all dimensions of the state space. Then Assumption 1 is satisfied. In particular, this implies that the optimal value function is monotone: for each $t \leqslant T$, $V_t^*(R_t, E_t, P_t, D_t)$ is nondecreasing in R_t, E_t, P_t, and D_t.*

Proof. See the online supplement.

7.3.1. Parameter Choices. In our experiments, a continuous distribution D with density f_D is discretized over a set $\mathscr{X}_D$ by assigning each outcome $x \in \mathscr{X}_D$ the probability $f_D(x)/\sum_{x' \in \mathscr{X}_D} f_D(x')$. We consider two instances of the energy storage problem for $T = 25$: the first is labeled S1 and has a smaller storage device and relatively low variance in the change in renewable supply $\hat{E}_{t+1}$, while the second, labeled S2, uses a larger storage device and has relatively higher variance in $\hat{E}_{t+1}$. We take $R_{\min} = 0$ with $R_{\max} = 30$ for S1 and $R_{\max} = 50$ for S2, and we set $\gamma_c = \gamma_d = 5$ for S1 and S2. The stochastic renewable supply has characteristics given by $E_{\min} = 1$ and $E_{\max} = 7$, with $\hat{E}_{t+1}$ being i.i.d. discrete uniform random variables over $\{0, \pm 1\}$ for S1 and $\hat{E}_{t+1}$ being i.i.d. $\mathcal{N}(0, 3^2)$ discretized over the set $\{0, \pm 1, \pm 2, \ldots, \pm 5\}$. For both cases, we have $P_{\min} = 30$ and $P_{\max} = 70$ with $\hat{P}_{t+1} = \epsilon_{t+1}^P + \mathbf{1}_{\{U_{t+1} < p\}} \epsilon_{t+1}^J$, to simulate price spikes (or jumps). The noise term ϵ_{t+1}^P is $\mathcal{N}(0, 2.5^2)$ discretized over $\{0, \pm 1, \pm 2, \ldots, \pm 8\}$; the noise term ϵ_{t+1}^J is $\mathcal{N}(0, 50^2)$ discretized over $\{0, \pm 1, \pm 2, \ldots, \pm 40\}$; and U_{t+1} is $\mathcal{U}(0, 1)$ with $p = 0.031$. Lastly, for the demand process, we take $D_{\min} = 0$, $D_{\max} = 7$, and $D_t + \hat{D}_{t+1} = \lfloor 3 - 4\sin(2\pi(t+1)/T) \rfloor + \epsilon_{t+1}^D$, where ϵ_{t+1}^D is $\mathcal{N}(0, 2^2)$ discretized over $\{0, \pm 1, \pm 2\}$, to roughly model the seasonality that often exists in observed energy demand. For both problems, we use an initial state of an empty storage device and the other dimensions of the state variable set to their minimum values: $S_0 = (R_{\min}, E_{\min}, P_{\min}, D_{\min})$. Table 2 summarizes the sizes of these two problems. Since the size of the action space is not constant over the state space, we report the average, i.e., $|\mathscr{S}|^{-1} \sum_s |\mathscr{X}(s)|$. The maximum size of the feasible set over the state space is also given (and is the same for both S1 and S2). Finally, we remark that the greater than linear increase in computation time for S2 compared to S1 is due to the larger action space and the larger number of random outcomes that need to be evaluated.

Table 2. Basic properties of energy storage problem instances.

Label	State space	Eff. state space	Action space	CPU (sec.)
S1	71,176	1,850,576	165, Max: 623	41,675
S2	117,096	3,044,496	178, Max: 623	115,822

7.3.2. Results. For this problem, we did not implement QL because of the impracticality of working with state-action pairs in problem domains with vastly larger action space than optimal stopping. The state-dependent action space also introduces implementation difficulties. With regard to KBRL, API, and AVI, the results for energy storage tell a similar story as before, as seen in Figure 8. The computational gap between Monotone-ADP and BDP, however, has increased even further. As illustrated in Figure 9, the ratio of the amount of computation time for Monotone-ADP to reach near-optimality to the amount of computation needed for BDP stands at 1.9% for S1 and 0.7% for S2, reaching two orders of magnitude.

7.4. Glycemic Control for Diabetes

Our final application is in the area of healthcare, concerning optimal treatment decisions for glycemic control (i.e., the management of blood glucose) in Type 2 diabetes patients over a time horizon $t \in \{0, 1, \ldots, T\}$. The model is based primarily on the work Hsih (2010) (with a few extensions) and also exhibits similarities to the ones described in Kurt et al. (2011), Mason et al. (2012), and Mason et al. (2014). Because diabetes is a chronic disease, its patients require long-term, personalized treatment plans that take into account a variety of factors, such as measures of blood glucose, negative side effects, and medical costs. The idea of the model is to maximize a utility function (often chosen to be related to *quality-adjusted life years*, or QALYs in the literature) over time. The action at each time period is a treatment decision that has a stochastic effect on the patient's state of health, represented via the state variable $S_t = (H_t^a, H_t^b, H_t^c, H_t^d)$, where $H_t^a \in \mathscr{H}^a = \{H_{\min}^a, \ldots, H_{\max}^a\}$ and $H_t^b \in \mathscr{H}^b = \{H_{\min}^b, \ldots, H_{\max}^b\}$ are measures of blood glucose; $H_t^c \in \mathscr{H}^c = \{H_{\min}^c, \ldots, H_{\max}^c\}$ is a patient's BMI (body mass index); and $H_t^d \in \mathscr{H}^d = \{H_{\min}^d, \ldots, H_{\max}^d\}$ is the severity of side effects (e.g., gastrointestinal discomfort or hypoglycemia). More specifically, let H_t^a be the FPG (fasting plasma glucose) level, a short term indicator of blood glucose and H_t^b be the HbA_{1c} (glycated hemoglobin) level, an indicator of average blood glucose over a longer term of a few months. The set of available treatment decisions is

Figure 8. Comparison of Monotone-ADP to other ADP/RL algorithms.

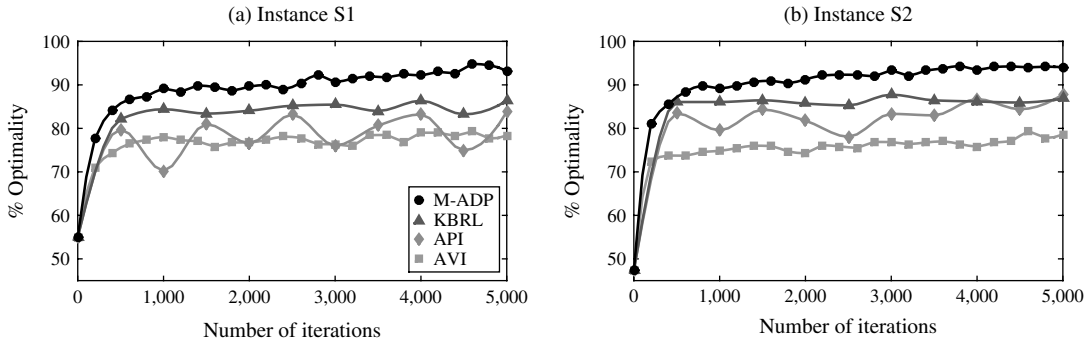
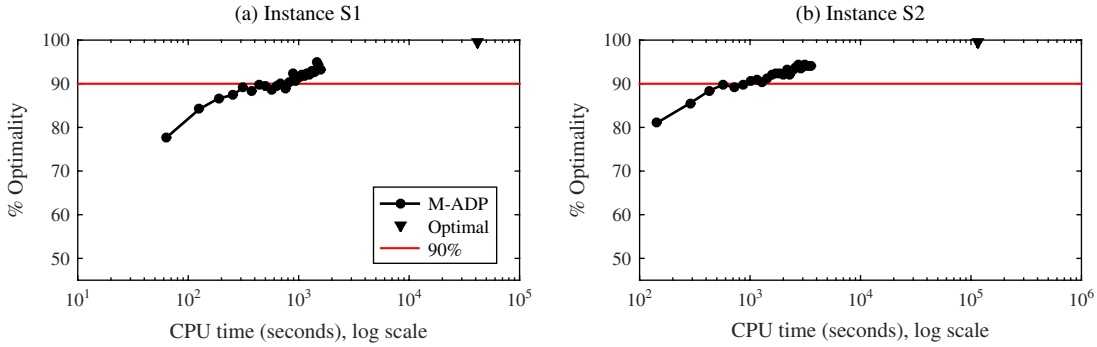


Figure 9. Computation times (seconds) of Monotone-ADP vs. backward dynamic programming.



$\mathscr{X} = \{0, 1, 2, 3, 4\}$. Below is basic summary of the various treatments:

- *No treatment*, $a_t = 0$. It is often the case that the usual recommendations of diet and exercise can be sufficient when the patient's glucose levels are in the normal range; there is no cost and no risk of increasing the severity of side effects.
- *Insulin sensitizers*, $a_t = 1$. These drugs increase the patient's sensitivity to insulin; the two most common types are called *biguanides* and *thiazolidinediones* (TZDs).
- *Secretagogues*, $a_t = 2$. Pancreatic β-cells are responsible for the release of insulin in the human body. Secretagogues act directly on β-cells to increase insulin secretion.
- *Alpha-glucosidase inhibitors*, $a_t = 3$. As the name suggests, this treatment disrupts the enzyme *alpha-glucosidase*, which breaks down carbohydrates into simple sugars, and therefore decreases the rate at which glucose is absorbed into the bloodstream.
- *Peptide analogs*, $a_t = 4$. By acting on certain *incretins*, which are hormones that affect the insulin production of pancreatic β-cells, this type of medication is able to regulate blood glucose levels.

We make the assumption that the patient under consideration has levels of H_t^a, H_t^b, H_t^c, and H_t^d *higher* (i.e., worse) than the normal range (as is typical of a diabetes patient) and only model this regime. Therefore, assume that we have *nonincreasing* utility functions u_t^i: $\mathscr{H}^i \to \mathbb{R}$ for $i \in \{a, b, c, d\}$ and that the cost of treatment is $P \geqslant 0$. The contribution function at time t is

$$C_t(S_t, a_t) = u_t^a(H_t^a) + u_t^b(H_t^b) + u_t^b(H_t^b) + u_t^d(H_t^d) - P \cdot \mathbf{1}_{\{a_t \neq 0\}}.$$

Furthermore, the information process in this problem is the stochastic effect of the treatment on the patient. This effect is denoted $W_{t+1} = (\hat{H}_{t+1}^a, \hat{H}_{t+1}^b, \hat{H}_{t+1}^c, \hat{H}_{t+1}^d)$ with the transitions given by $H_{t+1}^i = [H_t^i + \hat{H}_{t+1}^i]_{H_{\min}^i}^{H_{\max}^i}$ for each $i \in \{a, b, c, d\}$. Of course, the distribution of W_{t+1} depends on the treatment decision x_t, but we assume it is independent of the state variable S_t.

Notice that in this problem, the contribution function is nonincreasing with respect to the state variable, reversing the monotonicity in the value function as well.

Proposition 5. *Under the glycemic control model, define the Bellman operator H as in (7), with $\mathscr{A}$ replaced with the set of treatment decisions $\mathscr{X}$, and let $\preceq$ be the componentwise inequality over all dimensions of the state space. Then Assumption 1 is satisfied, with the direction of $\preceq$ reversed. In particular, this implies that the optimal value function is monotone: for each $t \leqslant T$, $V_t^*(H_t^a, H_t^b, H_t^c, H_t^d)$ is nonincreasing in H_t^a, H_t^b, H_t^c, and H_t^d.*

Proof. Proposition 1 can be applied, with the direction of the inequalities reversed.

7.4.1. Parameter Choices. We remark that the parameters used in our experimental work are not realistic or estimated from data, but chosen so that the resulting MDP is interesting and not too easy to solve (we found that the original parameters from Hsih 2010 created a problem that Monotone-ADP could solve to near optimality in as little as 50 iterations, making for a weak comparison). We consider two variations of the glycemic control problem, labeled G1 and G2, where the only difference is that for G1, we assume there is no cost of treatment, i.e., $P = 0$, and for G2, we set $P = 2$. This comparison is made to illustrate that a trivial, seemingly inconsequential change to the problem can create dramatic difficulties for certain ADP algorithms, as we see in the next section.

The finite time horizon for glycemic control is chosen as $T = 12$ (time is typically measured in units of a few months for a problem such as this). The lower and upper bounds of the state variable are given by

$$(H^a_{\min}, H^b_{\min}, H^c_{\min}, H^d_{\min}) = (68, 4, 19, 0),$$
$$(H^a_{\max}, H^b_{\max}, H^c_{\max}, H^d_{\max}) = (300, 20, 50, 10).$$

For each health indicator $i \in \{a, b, c\}$, the utility function is taken to have the form $u^i_t(h) = k^i \log(H^i_{\max} - h)$ for all t. The values of the constant are $k^a = 4.586$, $k^b = 7.059$, and $k^c = 5.771$. Additionally, let the utility function for side effects be given by $u^d_t(h) = -10h$.

Next, we assume that the distribution of $(\hat{H}^a_{t+1}, \hat{H}^b_{t+1}, \hat{H}^c_{t+1})^\intercal$ conditional on $x_t = x$ is multivariate normal for all t, with mean vector μ^x and covariance matrix Σ^x:

$$\mu^x = \begin{bmatrix} \mu^{x,a} \\ \mu^{x,b} \\ \mu^{x,c} \end{bmatrix} \quad \text{and} \quad \Sigma^x = \begin{bmatrix} \sigma^{x,aa} & \sigma^{x,ab} & \sigma^{x,ac} \\ \sigma^{x,ab} & \sigma^{x,bb} & \sigma^{x,bc} \\ \sigma^{x,ac} & \sigma^{x,bc} & \sigma^{x,cc} \end{bmatrix}.$$

Discretization of these continuous distributions is performed in the same way as described in §7.3.1. The set of values onto which we discretize is a hyperrectangle, where dimension i takes values between $\mu^{x,i} \pm 3\sqrt{\sigma^{x,ii}}$, for $i \in \{a, b, c\}$. The distribution of $\hat{H}^d_{t+1}$ conditional on $x_t = x$ (change in side effect severity for treatment x) is a discrete distribution that takes values $\hat{h}^{x,d}$ and with probabilities $p^{x,d}$ (both vectors), for all t. The numerical values of these parameters are given in Table 3. Finally, as we did for the previous two problems, Table 4 shows state space and computation time information for the glycemic control problem.

Table 4. Basic properties of glycemic control problem instances.

Label	State space	Eff. state space	Action space	CPU (sec.)
G1/G2	1,312,256	17,059,328	5	201,925

The initial state is set to $S_0 = (H^a_{\max}, H^b_{\max}, H^c_{\max}, H^d_{\min})$ to represent an unhealthy diabetes patient who has not undergone any treatment (and therefore, no side effects).

7.4.2. Results. In the numerical work for glycemic control, we show a slightly different algorithmic phenomenon. Recall that in G1, there no cost of treatment, and consequently, the contribution function is independent of the treatment decision. It turns out that after relatively few iterations, *all* of the ADP algorithms are able to learn that there is *some value to be gained* by applying treatment. Figure 10(a) shows that they end up achieving policies between 75% and 90% of optimality, with Monotone-ADP outperforming KBRL by roughly 10% and AVI (the worst performing) by only 15%. What if we add in a seemingly minor treatment cost of $P = 2$? Figure 10(b), on the other hand, shows a dramatic failure of AVI: it never improves beyond 10% of optimality. API shows similar behavior, and KBRL performed slightly worse (approx. 5%) than it did on G1 and reached a plateau in the middle iterations. The reason for AVI's poor performance is that it updates values too slowly from the initialization of $\bar{V}^0$ (usually constant over $\mathcal{S}$)—in other words, the future value term of Bellman's equation, $\mathbf{E}[\bar{V}^n_{t+1}(S_{t+1}) \mid S_t = s, a_t = a]$, is unable to compensate for the treatment cost of $P = 2$ quickly enough. We therefore conclude that it can be crucially important to update large swathes of states at a time, while observing the structural behavior. Even though KBRL and API do generalize to the entire state space, our observations from comparing G1 and G2 point to the additional value gained from using structural information. The computation time results of Figure 11 are similar to that of the previous examples. Monotone-ADP once again produces near-optimal solutions with significantly less computation: a ratio of 1.5% for G1 and 1.2% for G2.

8. Conclusion

In this paper, we formulated a general sequential decision problem with the property of a monotone value function. We then formally described an ADP algorithm, Monotone-ADP, first proposed in Papadaki and Powell

Table 3. Parameter values for glycemic control problem.

Treatment	$\mu^{x,a}$	$\mu^{x,b}$	$\mu^{x,c}$	$\sigma^{x,aa}$	$\sigma^{x,bb}$	$\sigma^{x,cc}$	$\sigma^{x,ab}$	$\sigma^{x,ac}$	$\sigma^{x,bc}$	$\hat{h}^{x,d}$	$p^{x,d}$
$x_t = 0$	30	3	2	25	8	8	0.8	0.5	0.2	$[-1, 0]$	$[0.8, 0.2]$
$x_t = 1$	−25	−1	3	100	16	25	1.2	0.5	0.2	$[0, 1, 2]$	$[0.8, 0.1, 0.1]$
$x_t = 2$	−45	−3	5	100	16	25	1.2	0.5	0.1	$[0, 1, 2]$	$[0.75, 0.15, 0.1]$
$x_t = 3$	−10	−1	−1	81	10	16	0.6	0.5	0.5	$[0, 1, 2]$	$[0.8, 0.1, 0.1]$
$x_t = 4$	−10	−1	−4	81	10	16	1.2	0.5	0.5	$[0, 1, 2]$	$[0.7, 0.2, 0.1]$

Figure 10. Comparison of Monotone-ADP to other ADP/RL algorithms.

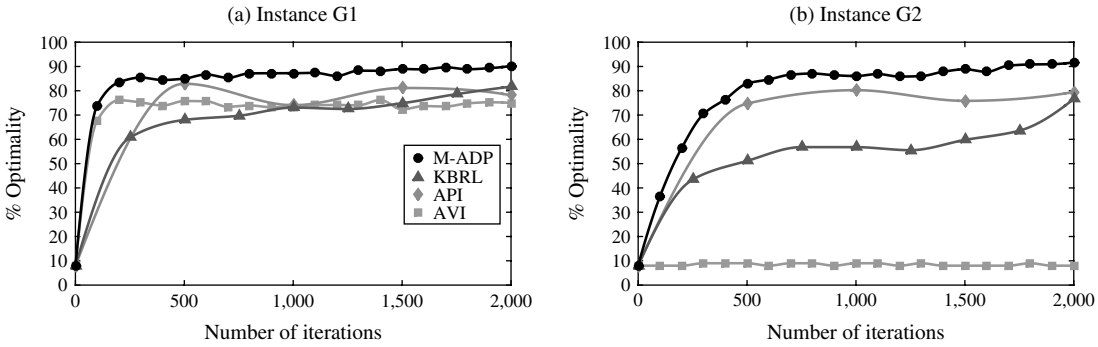
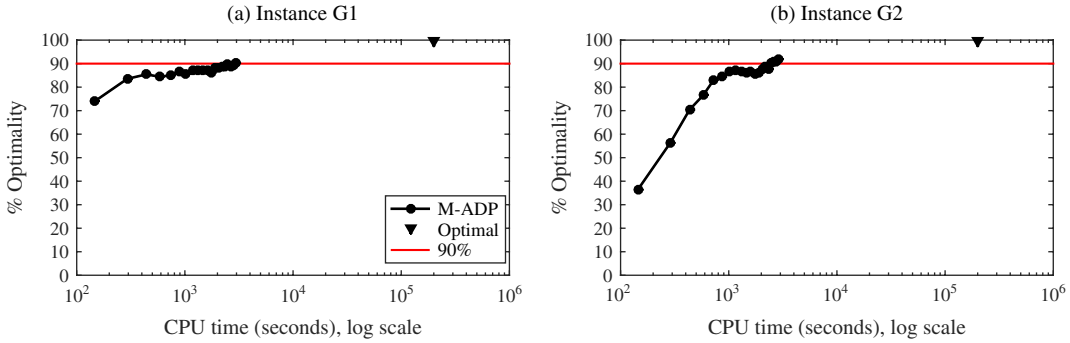


Figure 11. Computation times (seconds) of Monotone-ADP vs. backward dynamic programming.



(2002) as a heuristic for one dimensional state variables, that exploits the structure of the value function by performing a monotonicity preserving operation at each iteration to increase the information gained from each random observation. The algorithm can be applied in the context of three common formulations of Bellman's equation, the pre-decision version, post-decision version and the Q-function (or state-action) version. Under several technical assumptions, we prove the almost sure convergence of the estimates produced by Monotone-ADP to the optimal value function. The proof draws upon techniques used in Tsitsiklis (1994) and Nascimento and Powell (2009). However, in Nascimento and Powell (2009), where concavity was assumed, pure exploitation could be used, but only in one dimension. This paper requires a full exploration policy but exploits monotonicity in multiple dimensions, dramatically accelerating the rate of convergence. We then presented three example applications: regenerative optimal stopping, energy storage/allocation, and glycemic control for diabetes patients. Our empirical results show that in these problem domains, Monotone-ADP outperforms several popular ADP/RL algorithms (kernel-based reinforcement learning, approximate policy iteration, asynchronous value iteration, and Q-learning) by exploiting the additional structural information. Moreover, it can produce near-optimal policies using up to *two orders of magnitude* less computational resources than backward dynamic programming. In an application where the optimal solution cannot be computed exactly because of a large state space, we expect that utilizing monotonicity can bring significant advantages.

Supplemental Material

Supplemental material to this paper is available at http://dx.doi.org/10.1287/opre.2015.1425.

Acknowledgments

The authors are grateful to the area editor, associate editor, and three anonymous reviewers whose comments have helped to significantly improve earlier versions of this paper. This project was funded in part by the National ScienceFoundation [Grant ECCS-1127975], and the SAP Initiative for Energy Systems Research.

Appendix A. Proof of Lemma 1

Proof. The following arguments apply to all three definitions of H. (i) is true because of the monotonicity of the supremum (or maximum) and expectation operators. (ii) follows by the definition of H (for each of the three cases) and (iii) follows from the well-known fact that our finite horizon MDP has a unique solution. (iii) is easily deduced by applying the definition of H.

Appendix B. Proof of Proposition 2

Proof. Let $\bar{V}_t^{\Pi} = \Pi_M(S_t^n, z_t^n(S_t^n), \bar{V}_t^{n-1})$ and consider an arbitrary feasible solution $\tilde{V}_t \in \mathcal{V}_M(S_t^n, z_t^n(S_t^n))$ to (9). To prove the statement of the proposition, we need to argue that

$$\|\bar{V}_t^{\Pi} - \bar{V}_t^{n-1}\|_2 \leqslant \|\tilde{V}_t - \bar{V}_t^{n-1}\|_2.$$

We do so by verifying that for each state $s \in \mathcal{S}$,

$$|\bar{V}_t^{\Pi}(s) - \bar{V}_t^{n-1}(s)|^2 \leqslant |\tilde{V}_t(s) - \bar{V}_t^{n-1}(s)|^2. \tag{B1}$$

There are four cases:

1. If s and S_t^n are incomparable (neither $s \preceq S_t^n$ nor $S_t^n \preceq s$), then monotonicity does not apply and $\bar{V}_t^{\Pi}(s) = \bar{V}_t^{n-1}(s)$, trivially satisfying (B1).

2. If $s = S_t^n$, then the definition of the feasible region gives $\bar{V}_t^{\Pi}(s) = \tilde{V}_t(s)$. Once again, (B1) is satisfied.

3. Consider the case where $s \succeq S_t^n$ and $s \neq S_t^n$. First, if monotonicity is not violated with the new observation $z_t^n(S_t^n)$, then Π_M does not alter the value at s. Therefore, $\bar{V}_t^{\Pi}(s) = \bar{V}_t^{n-1}(s)$ and (B1) holds. Now suppose monotonicity is violated, meaning that $\bar{V}_t^{n-1}(s) \leqslant z_t^n(S_t^n)$. After applying Π_M, we have that $\bar{V}_t^{\Pi}(s) = z_t^n(S_t^n)$. Since $\tilde{V}_t$ is in the feasible set, it must be the case that $\tilde{V}_t(s) \geqslant z_t^n(S_t^n)$ since $\tilde{V}_t(S_t^n) = z_t^n(S_t^n)$. Combining these relationships, it is clear that (B1) holds.

4. The case where $s \preceq S_t^n$ and $s \neq S_t^n$ is handled in an analogous way.

Since this holds for any feasible solution $\tilde{V}_t$, the proof is complete.

Appendix C. Proof of Lemma 3

Proof. The proof is by induction on k. We note that by definition and (3), U^0 and L^0 satisfy this property. By the definition of H and Assumption 1, it is easy to see that if U^k satisfies the property, then HU^k does as well. Thus, by the definition of U^{k+1} being the average of the two, we see that U^{k+1} also satisfies the monotonicity property.

Appendix D. Proof of Lemma 4

Proof. Consider $\mathcal{S}_t^-$ (the other case is symmetric). Since $\mathcal{S}$ is finite, there exists a state $\underline{s}$ such that there is no state $s' \in \mathcal{S}$ where $s' \preceq \underline{s}$. An increase from the projection operator must originate from a violation of monotonicity during an observation of a state s' where $s' \preceq \underline{s}$ and $s' \neq \underline{s}$, but such a state does not exist. Thus, $\underline{s} \in \mathcal{S}_t^-$.

Appendix E. Proof of Lemma 5

Proof. Define

$$A = \{s'' \colon s'' \preceq s,\ s'' \neq s\} \quad \text{and} \quad B = \bigcup_{s_l \in \mathcal{S}_L(s)} \{s'' \colon s'' \preceq s_l\}.$$

We argue that $A \subseteq B$. Choose $s_1 \in A$ and suppose for the sake of contradiction that $s_1 \notin B$. In particular, this means that $s_1 \notin \mathcal{S}_L(s)$ because $\mathcal{S}_L(s) \subseteq B$. By Definition 6, it follows that there must exist s_2 such that $s_1 \preceq s_2 \preceq s$ where $s_2 \neq s_1$ and $s_2 \neq s$. It now follows that $s_2 \notin \mathcal{S}_L(s)$ because if it were, then $s_1 \preceq s_2$ would imply that s_1 is an element of B. This argument can be repeated to produce other states $s_3, s_4, \ldots$, each *different* from the rest, such that

$$s_1 \preceq s_2 \preceq s_3 \preceq s_4 \preceq \cdots \preceq s, \tag{E1}$$

where each state s_k is not an element $\mathcal{S}_L(s)$. However, because $\mathcal{S}$ is a finite set, eventually we reach a point where we cannot produce another state to satisfy Definition 6 and we will have that the final state, call it s_K, is an element of $\mathcal{S}_L$. Here, we reach a contradiction because (E1) (specifically, the fact that $s_1 \preceq s_K$) implies that $s_1 \in B$. Thus, $s_1 \in B$ and we have shown that $A \subseteq B$.

Because the value of s was increased, a violation of monotonicity must have occurred during the observation of S_t^n, implying that $S_t^n \in A$. Therefore, $S_t^n \in B$, and we know that $S_t^n \preceq s'$ for some $s' \in \mathcal{S}_L(s)$. Since $\bar{V}_t^{n-1}$ is monotone over $\mathcal{S}$ and $\bar{V}_t^n(s) = z_t^n(S_t^n)$, we can write

$$\bar{V}_t^{n-1}(S_t^n) \leqslant \bar{V}_t^{n-1}(s') \leqslant \bar{V}_t^{n-1}(s) < z_t^n(S_t^n) = \bar{V}_t^n(s),$$

meaning that Π_M acts on s' and we have

$$\bar{V}_t^n(s') = z_t^n(S_t^n) = \bar{V}_t^n(s),$$

the desired result.

Appendix F. Proof of Lemma 6

Proof. We first notice that the product inside the limit is nonnegative because the stepsizes $\alpha_t^n \leqslant 1$. Also the sequence is monotonic; therefore, the limit exists. Now,

$$\prod_{n=1}^{m}(1-\alpha_t^n(s)) = \exp\left[\sum_{n=1}^{m}\log(1-\alpha_t^n(s))\right] \leqslant \exp\left[-\sum_{n=1}^{m}\alpha_t^n(s)\right],$$

where the inequality follows from $\log(1-x) \leqslant -x$. Since

$$\sum_{n=1}^{\infty}\alpha_t^n(s) = \infty \quad \text{a.s.},$$

the result follows by appropriately taking limits.

Appendix G. Proof of Lemma 8

Proof. First, we state an inequality needed later in the proof. Since $n \geqslant \tilde{N}_t^k(s)$, we know that $n \geqslant N_{t+1}^k$ by the induction hypothesis for k. Now by the induction hypothesis for $t+1$, we know that $\bar{V}_{t+1}^n(s) \leqslant U_{t+1}^k(s)$ holds. Therefore, using (ii) of Lemma 1, we see that

$$(H\bar{V}^n)_t(s) \leqslant (HU^k)_t(s). \tag{G1}$$

To show the statement of the lemma, we induct forward on n.

Base case, $n = \tilde{N}_t^k(s)$. By the induction hypothesis for k (which we can safely use in this proof because of the placement of the lemma after the induction hypothesis), we have that $\bar{V}_t^{\tilde{N}_t^k(s)}(s) \leqslant U_t^k(s)$. Combined with

$$W_t^{\tilde{N}_t^k(s),\,\tilde{N}_t^k(s)}(s) = 0 \quad \text{and} \quad U_t^k(s) = X_t^{\tilde{N}_t^k(s)}(s),$$

we see that the statement of the lemma holds for the base case.

Induction hypothesis, n. Suppose the statement of the lemma holds for n.

Inductive step from n to $n+1$. Suppose $S_t^{n+1} = s$, meaning a direct update happened on iteration $n+1$. Thus, we have that

$$\begin{aligned}
\bar{V}_t^{n+1}(s) &= z_t^{n+1}(s)\\
&= (1-\alpha_t^n(s))\bar{V}_t^n(s) + \alpha_t^n(s)\hat{v}_t^{n+1}(s)\\
&= (1-\alpha_t^n(s))\bar{V}_t^n(s) + \alpha_t^n(s)[(H\bar{V}^n)_t(s) + w_t^{n+1}(s)]\\
&\leqslant (1-\alpha_t^n(s))(X_t^n(s) + W_t^{n,\tilde{N}_t^k(s)}(s))\\
&\quad + \alpha_t^n(s)[(H\bar{V}^n)_t(s) + w_t^{n+1}(s)] \qquad \text{(G2)}\\
&\leqslant (1-\alpha_t^n(s))(X_t^n(s) + W_t^{n,\tilde{N}_t^k(s)}(s))\\
&\quad + \alpha_t^n(s)[(HU^k)_t(s) + w_t^{n+1}(s)] \qquad \text{(G3)}\\
&= X_t^{n+1}(s) + W_t^{n+1,\tilde{N}_t^k(s)}(s).
\end{aligned}$$

where (G2) follows from the induction hypothesis for n, and (G3) follows from (G1). Now let us consider the second case, that $S_t^{n+1} \neq s$. This means that the stepsize $\alpha_t^{n+1}(s) = 0$ and thus,

$$\begin{aligned}
&X_t^{n+1}(s) = X_t^n(s),\\
&W_t^{n+1,\tilde{N}_t^k(s)}(s) = W_t^{n,\tilde{N}_t^k(s)}(s).
\end{aligned} \qquad \text{(G4)}$$

Because $s \in \mathcal{S}_t^-$ and $n+1 \geqslant \tilde{N}_t^k(s) \geqslant N^{\Pi}$ (induction hypothesis for k), we know that the projection operator did not increase the value of s on this iteration (though a decrease is possible). Hence,

$$\begin{aligned}
\bar{V}_t^{n+1}(s) &\leqslant \bar{V}_t^n(s) \leqslant X_t^n(s) + W_t^{n,\tilde{N}_t^k(s)}(s)\\
&\leqslant X_t^{n+1}(s) + W_t^{n+1,\tilde{N}_t^k(s)}(s),
\end{aligned}$$

by the induction hypothesis for n and (G4).

References

Asamov T, Powell WB (2015) Regularized decomposition of high-dimensional multistage stochastic programs with Markov uncertainty. Working paper, Princeton University, Princeton, NJ.

Ayer M, Brunk HD, Ewing GM (1955) An empirical distribution function for sampling with incomplete information. *Ann. Math. Statist.* 26(4):641–647.

Barlow RE, Bartholomew DJ, Bremner JM, Brunk HD (1972) *Statistical Inference Under Order Restrictions: The Theory and Application of Isotonic Regression* (John Wiley & Sons, New York).

Bertsekas DP (2007) *Dynamic Programming and Optimal Control*, Vol. II, 4 ed. (Athena Scientific, Belmont, MA).

Bertsekas DP (2011) Approximate policy iteration: A survey and some new methods. *J. Control Theory Appl.* 9(3):310–335.

Bertsekas DP, Tsitsiklis JN (1996) *Neuro–Dynamic Programming* (Athena Scientific, Belmont, MA).

Birge JR (1985) Decomposition and partitioning methods for multistage stochastic linear programs. *Oper. Res.* 33(5):989–1007.

Breiman L (1992) *Probability* (SIAM, Philadelphia, PA).

Brunk HD (1955) Maximum likelihood estimates of monotone parameters. *Ann. Math. Statist.* 26(4):607–616.

Carmona R, Ludkovski M (2010) Valuation of energy storage: An optimal switching approach. *Quant. Finance* 10(4):359–374.

Dette H, Neumeyer N, Pilz KF (2006) A simple nonparametric estimator of a strictly monotone regression function. *Bernoulli* 12(3):469–490.

Ekström E (2004) Properties of American option prices. *Stochastic Processes Appl.* 114(2):265–278.

Feldstein MS, Rothschild M (1974) Towards an economic theory of replacement investment. *Econometrica* 42(3):393–424.

George AP, Powell WB (2006) Adaptive stepsizes for recursive estimation with applications in approximate dynamic programming. *Machine Learn.* 65(1):167–198.

Hsih KW (2010) Optimal dosing applied to glycemic control of type 2 diabetes. Senior thesis, Princeton University, Princeton, NJ.

Jiang DR, Powell WB (2015) Optimal hour-ahead bidding in the real-time electricity market with battery storage using approximate dynamic programming. *INFORMS J. Comput.* 27(3):525–543.

Kaplan G, Violante GL (2014) A model of the consumption response to fiscal stimulus payments. *Econometrica* 82(4):1199–1239.

Kim JH, Powell WB (2011) Optimal energy commitments with storage and intermittent supply. *Oper. Res.* 59(6):1347–1360.

Kleywegt AJ, Shapiro A, Homem-de Mello T (2002) The sample average approximation method for stochastic discrete optimization. *SIAM J. Optim.* 12(2):479–502.

Kurt M, Kharoufeh JP (2010) Monotone optimal replacement policies for a Markovian deteriorating system in a controllable environment. *Oper. Res. Lett.* 38(4):273–279.

Kurt M, Denton BT, Schaefer AJ, Shah ND, Smith SA (2011) The structure of optimal statin initiation policies for patients with type 2 diabetes. *IIE Trans. Healthcare Systems Engrg.* 1(1):49–65.

Luenberger DG (1998) *Investment Science* (Oxford University Press, New York).

Mammen E (1991) Estimating a smooth monotone regression function. *Ann. Statist.* 19(2):724–740.

Mason JE, Denton BT, Shah ND, Smith SA (2014) Optimizing the simultaneous management of blood pressure and cholesterol for type 2 diabetes patients. *Eur. J. Oper. Res.* 233(3):727–738.

Mason JE, England DA, Denton BT, Smith SA, Kurt M, Shah ND (2012) Optimizing statin treatment decisions for diabetes patients in the presence of uncertain future adherence. *Medical Decision Making* 32(1):154–166.

McCall JJ (1970) Economics of information and job search. *Quart. J. Econom.* 84(1):113–126.

Mukerjee H (1988) Monotone nonparametric regression. *Ann. Statist.* 16(2):741–750.

Müller A (1997) How does the value function of a Markov decision process depend on the transition probabilities? *Math. Oper. Res.* 22(4): 872–885.

Nadaraya EA (1964) On estimating regression. *Theory Probab. Appl.* 9(1):141–142.

Nascimento JM, Powell WB (2009) An optimal approximate dynamic programming algorithm for the lagged asset acquisition problem. *Math. Oper. Res.* 34(1):210–237.

Nascimento JM, Powell WB (2010) Dynamic programming models and algorithms for the mutual fund cash balance problem. *Management Sci.* 56(5):801–815.

Ormoneit D, Sen A (2002) Kernel-based reinforcement learning. *Machine Learn.* 49(2–3):161–178.

Papadaki KP, Powell WB (2002) Exploiting structure in adaptive dynamic programming algorithms for a stochastic batch service problem. *Eur. J. Oper. Res.* 142(1):108–127.

Papadaki KP, Powell WB (2003a) A discrete online monotone estimation algorithm. Working paper LSEOR 03.73, London School of Economics, London.

Papadaki KP, Powell WB (2003b) An adaptive dynamic programming algorithm for a stochastic multiproduct batch dispatch problem. *Naval Res. Logist.* 50(7):742–769.

Pereira MVF, Pinto LMVG (1991) Multi-stage stochastic optimization applied to energy planning. *Math. Programming* 52(1–3):359–375.

Pierskalla WP, Voelker JA (1976) A survey of maintenance models: The control and surveillance of deteriorating systems. *Naval Res. Logist. Quart.* 23(3):353–388.

Powell WB (2011) *Approximate Dynamic Programming: Solving the Curses of Dimensionality* 2nd ed. (John Wiley & Sons, Hoboken, NJ).

Powell WB, Ruszczyński A, Topaloglu H (2004) Learning algorithms for separable approximations of discrete stochastic optimization problems. *Math. Oper. Res.* 29(4):814–836.

Puterman ML (1994) *Markov Decision Processes: Discrete Stochastic Dynamic Programming* (Wiley, New York).

Ramsay JO (1998) Estimating smooth monotone functions. *J. Roy. Statist. Soc.: Ser. B (Statist. Methodology)* 60(2):365–375.

Ross SM (1983) *Introduction to Stochastic Dynamic Programming* (Academic Press, New York).

Rust J (1987) Optimal replacement of GMC bus engines: An empirical model of Harold Zurcher. *Econometrica: J. Econometric Soc.* 55(5):999–1033.

Salas D, Powell WB (2013) Benchmarking a scalable approximation dynamic programming algorithm for stochastic control of multidimensional energy storage problems. Working paper, Princeton University, Princeton, NJ.

Scott DW (2009) *Multivariate Density Estimation: Theory, Practice, and Visualization*, Vol. 383 (John Wiley & Sons, New York).

Scott WR, Powell WB, Moazeni S (2012) Least squares policy iteration with instrumental variables vs. direct policy search: Comparison against optimal benchmarks using energy storage. Working paper, Princeton University, Princeton, NJ.

Secomandi N (2010) Optimal commodity trading with a capacitated storage asset. *Management Sci.* 56(3):449–467.

Smith JE, McCardle KF (2002) Structural properties of stochastic dynamic programs. *Oper. Res.* 50(5):796–809.

Stockey N, Lucas Jr RE (1989) *Recursive Methods in Economic Dynamics* (Harvard University Press, Cambridge, MA).

Sutton RS, Barto AG (1998) *Reinforcement Learning: An Introduction* (MIT Press, Cambridge, MA).

Topaloglu H, Powell WB (2003) An algorithm for approximating piecewise linear concave functions from sample gradients. *Oper. Res. Lett.* 31(1):66–76.

Tsitsiklis JN (1994) Asynchronous stochastic approximation and *Q*-learning. *Machine Learn.* 16(3):185–202.

Watkins CJ, Dayan P (1992) *Q*-learning. *Machine Learn.* 8(3–4):279–292.

Daniel R. Jiang did his Ph.D. research in the Department of Operations Research and Financial Engineering at Princeton University. His research interests are in stochastic optimization, sequential decision making, and approximate dynamic programming, with applications in energy systems and markets.

Warren B. Powell is a professor in the Department of Operations Research and Financial Engineering at Princeton University, where he has taught since 1981. His research specializes in computational stochastic optimization, with applications in energy, transportation, finance, and health. He founded and directs CASTLE Labs and the Princeton Laboratory for Energy Systems Analysis (PENSA), focusing on the solution of stochastic optimization problems that arise in a wide range of applications.