https://pubsonline.informs.org/journal/opre

OPERATIONS RESEARCH
Articles in Advance, pp. 1–18
ISSN 0030-364X (print), ISSN 1526-5463 (online)

Crosscutting Areas

The Role of Lookahead and Approximate Policy Evaluation in Reinforcement Learning with Linear Value Function Approximation

Anna Winnicki,[a,b,*] Joseph Lubars,[c] Michael Livesay,[c] R. Srikant[a,b,d]

[a] Department of Electrical and Computer Engineering, University of Illinois Urbana-Champaign, Urbana, Illinois 61801; [b] Coordinated Science Laboratory, University of Illinois Urbana-Champaign, Urbana, Illinois 61801; [c] Sandia National Laboratories, Albuquerque, New Mexico 87123; [d] c3.ai Digital Transformation Institute, University of Illinois Urbana-Champaign, Urbana, Illinois 61801

*Corresponding author

Contact: annaw5@illinois.edu, https://orcid.org/0000-0001-9880-2340 (AW); lubars2@illinois.edu, https://orcid.org/0000-0001-9273-8456 (JL); mlivesa@sandia.gov, https://orcid.org/0000-0002-2594-3772 (ML); rsrikant@illinois.edu, https://orcid.org/0000-0003-1483-5204 (RS)

Received: July 12, 2022
Revised: September 11, 2023; February 13, 2024
Accepted: February 21, 2024
Published Online in Articles in Advance: May 30, 2024

Area of Review: Machine Learning and Data Science

https://doi.org/10.1287/opre.2022.0357

Abstract. Function approximation is widely used in reinforcement learning to handle the computational difficulties associated with very large state spaces. However, function approximation introduces errors that may lead to instabilities when using approximate dynamic programming techniques to obtain the optimal policy. Therefore, techniques such as lookahead for policy improvement and m-step rollout for policy evaluation are used in practice to improve the performance of approximate dynamic programming with function approximation. We quantitatively characterize the impact of lookahead and m-step rollout on the performance of approximate dynamic programming (DP) with function approximation. (i) Without a sufficient combination of lookahead and m-step rollout, approximate DP may not converge. (ii) Both lookahead and m-step rollout improve the convergence rate of approximate DP. (iii) Lookahead helps mitigate the effect of function approximation and the discount factor on the asymptotic performance of the algorithm. Our results are presented for two approximate DP methods: one that uses least-squares regression to perform function approximation and another that performs several steps of gradient descent of the least-squares objective in each iteration.

Funding: The research presented here was supported in part by a grant from Sandia National Labs and the NSF [Grants CCF 1934986, CCF 2207547, CNS 2106801], ONR [Grant N00014-19-1-2566], and ARO [Grant W911NF-19-1-0379].

Keywords: Markov decision processes • dynamic programming

1. Introduction

In many applications of reinforcement learning (RL), such as playing chess and Go, the underlying model is known, and so, the main challenge is in solving the associated dynamic programming problem in an efficient manner. Policy iteration (PI) and variants of PI (Bertsekas and Tsitsiklis 1996; Bertsekas 2011, 2019) that solve dynamic programming problems rely on computations that are infeasible because of the sizes of the state and action spaces in modern reinforcement learning problems. As a remedy to this "curse of dimensionality," several state-of-the-art algorithms (Mnih et al. 2016; Silver et al. 2017a, b) employ function approximation, lookahead for policy improvement, m-step rollout for policy evaluation, and gradient descent to compute the function approximation; see Section 2 for a definition of these terms.

In vanilla PI, one has to compute the value function associated with each state of a Markov decision process (MDP). This is clearly infeasible for large state spaces; therefore, a number of techniques are used to mitigate the computational intractability of PI. Our goal in this paper is to understand the role of multistep lookahead for policy improvement (i.e., repeatedly applying the Bellman operator multiple times) and m-step rollout (which is a technique to approximately evaluate a policy by rolling out the dynamic programming tree for a certain number of steps m; see Section 2 for definitions of these terms) on the accuracy of approximate PI techniques with linear value function approximation. The algorithms we study in this paper are closely related to least-squares policy iteration (LSPI) (Lagoudakis and Parr 2001, 2003; Buşoniu et al. 2012) and approximate PI; see Bertsekas and Tsitsiklis

(1996) and Bertsekas (2019). In the analysis of approximate PI, it is assumed that the policy evaluation and improvement steps have bounded errors, and using these, an error bound is obtained for the algorithm that repeatedly uses approximate policy evaluation and improvement. We remark that vanilla PI is a special case of approximate PI where there are no errors in policy evaluation and improvement. LSPI is an algorithm that builds on approximate PI where the policy evaluation step uses a least-squares algorithm to estimate the value function for the entire state space using the value function evaluated at a few states. However, the bounds presented in Lagoudakis and Parr (2003) as well as the related studies in Lagoudakis and Parr (2001) and Buşoniu et al. (2012) are simply a special case of the bounds for generic approximate PI (Bertsekas and Tsitsiklis 1996, Bertsekas 2019), and they do not explicitly take into account the details of the implementation of least squares-based policy evaluation. When such details are taken into account, it turns out that the roles of the depth of lookahead (H) and rollout (m) become important, and their impact on the error bounds on the performance of approximate value iteration has not been characterized in prior work.

The recent work in Efroni et al. (2019) considers a variant of PI that utilizes lookahead and approximate policy evaluation using an m-step rollout. As stated in the motivation in Efroni et al. (2019), it is well known that Monte Carlo tree search (MCTS) (Kocsis and Szepesvári 2006, Browne et al. 2012, Świechowski et al. 2023) works well in practice, even though the worst-case compute complexity can be exponential (Shah et al. 2020a); see Munos (2014) for some analysis of MCTS in MDPs, where the number of states that can be visited from a given state is bounded. It is important to note that many prior works use lookahead and that the use of tree search as an enhancement of training RL algorithms has become commonplace. For more on lookahead, see Hong et al. (2019).

Motivated by PI, the algorithm in Efroni et al. (2019) estimates the value function associated with a policy and aims to improve the policy at each step. Policy improvement is achieved by obtaining the "greedy" policy in the case of PI or a lookahead policy in the work of Efroni et al. (2019), which involves applying the Bellman operator several times to the current iterate before obtaining the greedy policy. The idea is that the application of the Bellman operator several times gives a more accurate estimate of the optimal value function. Then, similarly to PI, the algorithm in Efroni et al. (2019) aims to evaluate the new policy. The algorithm in Efroni et al. (2019) uses an m-step rollout to compute the value function associated with a policy (i.e., it applies the Bellman operator associated with the policy m times). The work of Efroni et al. (2019) establishes that a lookahead can significantly improve the rate of convergence if one uses the value function computed using lookahead in the approximate policy evaluation step. However, like the works of Bertsekas and Tsitsiklis (1996), Lagoudakis and Parr (2001, 2003), Buşoniu et al. (2012), and Bertsekas (2019), the work of Efroni et al. (2019) does not study the use of function approximation, which is critical to handling large state spaces, nor does it quantify the effects of varying m in the convergence of their algorithm. Our results show that the aforementioned results change drastically when least squares-based policy evaluation is incorporated. For a more detailed comparison of the works of Bertsekas and Tsitsiklis (1996), Lagoudakis and Parr (2001, 2003), Buşoniu et al. (2012), Bertsekas (2019), and Efroni et al. (2019) with our work, see Section 3.4. In this paper, we assume that policies are evaluated at a few states using an m-step rollout. The use of a partial rollout in our algorithm is similar to modified PI (Puterman and Shin 1978), which is also called optimistic PI (Bertsekas and Tsitsiklis 1996). We remark that vanilla PI is a special case of modified PI where $m = \infty$. However, motivated by Tsitsiklis and Roy (1994), we present an example that shows that the algorithm can diverge when function approximation is used. Therefore, our goal is to understand how to integrate linear value function approximation into the well-studied modified PI algorithm. To the best of our knowledge, none of the prior works consider the impact of using gradient descent to implement an approximate version of least-squares policy evaluation within approximate PI. Thus, our algorithm and analysis can be viewed as a detailed look at approximate PI and modified PI when linear function approximation, least-squares policy evaluation, and gradient descent are used to evaluate policies.

Our key contributions can be summarized as follows. We extend the analysis of approximate PI to allow for iteration-dependent policy evaluation and policy improvement errors. However, when we allow iteration-dependent errors, it is not clear that the accumulation of errors over multiple iterations can be bounded. We show that under least-squares function approximation as well as gradient descent-based function approximation, these errors can be bounded if lookahead is sufficiently large. Combining this with the counterexample motivated by Tsitsiklis and Roy (1994), we believe that our result is why lookahead is important in approximate policy iteration with function approximation. Since RL training can be viewed as a version of approximate PI, our results show the importance of lookahead in RL training and not just in implementing an RL agent. In particular, our paper contains the following results.

- We examine the impact of lookahead and m-step rollout on approximate PI with linear function approximation. As is common in practice, we assume that we evaluate an approximate value function only for some

states at each iteration. We obtain performance bounds for our algorithm under the assumption that the sum of the lookahead and the number of steps in the m-step rollout is sufficiently large. We demonstrate through an extension of a counterexample in Tsitsiklis and Roy (1994) that such a condition is necessary, in general, for convergence with function approximation, unlike the tabular setting in the prior works. See Section 3.2 for our counterexample.

- For ease of exposition, we first present the case where one solves a least-squares problem at each iteration to obtain the weights associated with the feature vectors in the function approximation of the value function in Section 3.4. Our performance bounds in this case generalize the bounds in Bertsekas and Tsitsiklis (1996), Lagoudakis and Parr (2001, 2003), Buşoniu et al. (2012), Bertsekas (2019), and Efroni et al. (2019) for approximate PI.
- We then consider a more practical and widely used scheme, where several steps of gradient descent are used to update the weights of the value function approximation at each iteration. Obtaining performance bounds for the gradient descent algorithm is more challenging, and these bounds can be found in Section 4.
- Our results show that the sufficient conditions on the hyperparameters (such as the amount of lookahead, rollout, and gradient descent parameters) of the algorithm required for convergence either do not depend on the size of the state space or depend only logarithmically on the size of the state space. Our results also illustrate the role of feature vectors in the amount of lookahead required.
- In addition to asymptotic performance bounds, we also provide finite-time guarantees for our algorithms. Our finite-time bounds show that our algorithm converges exponentially fast in the case of least squares as well as the case where a fixed number of gradient descent steps are performed in each iteration of the algorithm.
- We complement our theoretical results with experiments on the same grid world problem as in Efroni et al. (2019). These experiments are presented in Section 5.

1.1. Other Related Work

The role of lookahead and rollout in improving the performance of RL algorithms has also been studied in a large number of papers, including Efroni et al. (2018b, 2020), Deng et al. (2020), Moerland et al. (2020), Shah et al. (2020b), Springenberg et al. (2020), Tomar et al. (2020), and Winnicki and Srikant (2022, 2023). The works of Baxter et al. (1999), Veness et al. (2009), and Lanctot et al. (2014) explore the role of tree search in RL algorithms. However, to the best of our knowledge, the amount of lookahead and rollout needed as a function of the feature vectors has not been quantified in prior works.

The works of Bertsekas (2011, 2019) also study a variant of PI, wherein a greedy policy is evaluated approximately using feature vectors at each iteration. These papers also provide rates of convergence as well as a bound on the approximation error. However, our main goal is to understand the relations between function approximation and lookahead/rollout, which are not considered in these other works.

2. Preliminaries

We consider an MDP, which is defined to be a 5-tuple $(\mathcal{S}, \mathcal{A}, P, r, \alpha)$. The finite set of states of the MDP is $\mathcal{S}$. There exists a finite set of actions associated with the MDP $\mathcal{A}$. Let $P_{ij}(a)$ be the probability of transitioning from state i to state j when taking action $a \in \mathcal{A}$. We denote by s_k the state of the MDP and by a_k the corresponding action at time k. We associate with state s_k and action a_k a nondeterministic reward $r(s_k, a_k) \in [0,1]$ $\forall s_k \in \mathcal{S}, a_k \in \mathcal{A}$.

Our objective is to maximize the cumulative discounted reward with discount factor $\alpha \in (0,1)$. Toward this end, we seek to find a deterministic policy μ, which associates with each state $s \in \mathcal{S}$ an action $\mu(s) \in \mathcal{A}$. For every policy μ and every state $s \in \mathcal{S}$, we define $J^\mu(s)$ as follows:

$$J^\mu(s) := E\left[\sum_{i=0}^{\infty} \alpha^k r(s_k, \mu(s_k)) \middle| s_0 = s\right].$$

We define the optimal reward-to-go J^* as $J^*(s) := \max_\mu J^\mu(s)$. The objective is to find a policy μ that maximizes $J^\mu(s)$ for all $s \in \mathcal{S}$. Toward the objective, we associate with each policy μ a function $T_\mu : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$, where for $J \in \mathbb{R}^{|\mathcal{S}|}$, the sth component of $T_\mu J$ is

$$(T_\mu J)(s) = r(s, \mu(s)) + \alpha \sum_{j=1}^{|\mathcal{S}|} P_{sj}(\mu(s)) J(j),$$

for all $s \in S$. If function T_μ is applied m times to vector $J \in \mathbb{R}^{|\mathcal{S}|}$, then we say that we have performed an m-step rollout of the policy μ, and the result $T_\mu^m J$ of the rollout is called the return. It is well known that each time T_μ is applied to a vector J to obtain $T_\mu J$, the following holds:

$$\|T_\mu J - J^\mu\|_\infty \leq \alpha \|J - J^\mu\|_\infty,$$

where $\|\cdot\|_\infty$ refers to the supremum norm or the largest component of a vector. Thus, applying T_μ to obtain $T_\mu J$ gives a better estimate of the value function corresponding to policy μ than J. Furthermore, it is easy to see that the result of an m-step rollout of policy μ gives

the following:

$$\|T_\mu^m J - J^\mu\|_\infty \leq \alpha^m \|J - J^\mu\|_\infty,$$

and hence, increasing m yields better estimates of J^μ.

Similarly, we define the Bellman operator $T : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$ with the sth component of TJ being

$$(TJ)(s) = \max_{a \in \mathcal{A}} \left\{ r(s,a) + \alpha \sum_{j=1}^{|\mathcal{S}|} P_{sj}(a) J(j) \right\}. \tag{1}$$

The policy corresponding to the T operator is defined as the *greedy* policy. If operator T is applied H times to vector $J \in \mathbb{R}^{|\mathcal{S}|}$, we call the result—$T^H J$—the H-step "lookahead" corresponding to J. The greedy policy corresponding to $T^H J$ is called the H-step lookahead policy or the lookahead policy when H is understood. More precisely, given an estimate J of the value function, the lookahead policy is the policy μ such that $T_\mu(T^{H-1}J) = T(T^{H-1}J)$.

Similarly to T_μ, each time the Bellman operator is applied to a vector J to obtain TJ, the following holds:

$$\|TJ - J^*\|_\infty \leq \alpha \|J - J^*\|_\infty.$$

Thus, applying T to obtain TJ gives a better estimate of the value function than J.

The Bellman equations state that the vector J^μ is the unique solution to the linear equation

$$J^\mu = T_\mu J^\mu. \tag{2}$$

Additionally, we have that J^* is a solution to

$$J^* = TJ^*.$$

Note that every greedy policy w.r.t. J^* is optimal and vice versa (Bertsekas and Tsitsiklis 1996). More precisely, J^* is the value function corresponding to an optimal policy.

We will now state several useful properties of the operators T and T_μ. See Bertsekas and Tsitsiklis (1996) for more on these properties. Consider the vector $e \in \mathbb{R}^{|\mathcal{S}|}$ where $e(i) = 1 \ \forall i \in 1, 2, \ldots, |\mathcal{S}|$. We have

$$T(J + ce) = TJ + \alpha ce, \quad T_\mu(J + ce) = T_\mu J + \alpha ce. \tag{3}$$

Operators T and T_μ are also monotone:

$$J \leq J' \Rightarrow TJ \leq TJ', \quad T_\mu J \leq T_\mu J'. \tag{4}$$

Finally, in this paper, we repeatedly use the following induced ∞-norm of a matrix A :

$$\|A\|_\infty = \sup_{x \neq 0} \frac{\|Ax\|_\infty}{\|x\|_\infty}.$$

For reference, we include the notation in Table A.1 in Appendix A.

Algorithm 1 (Approximate PI with Lookahead)

Input: θ_0, m, H.

1: Let $k = 0$.
2: Let μ_{k+1} be such that $\|T^H J_k - T_{\mu_{k+1}} T^{H-1} J_k\|_\infty \leq \varepsilon_{LA}$.
3: Compute $\hat{J}^{\mu_{k+1}}$ such that $\hat{J}^{\mu_{k+1}}$ satisfies the following:

$$\|\hat{J}^{\mu_{k+1}} - J^{\mu_{k+1}}\|_\infty \leq \delta.$$

4: $J_{k+1} = \hat{J}^{\mu_{k+1}}$.
5: Set $k \leftarrow k + 1$. Go to (2).

3. Approximate PI with Linear Value Function Approximation

As mentioned in Section 1, the work of Efroni et al. (2019) extends the result of Bertsekas (2019) to incorporate the use of lookahead policies as opposed to one-step greedy policies as well as m-step returns. We outline the algorithm of Efroni et al. (2019) in Algorithm 1. We then wish to incorporate linear value function approximation into the analysis. We will outline the approximate PI algorithm with lookahead and linear value function approximation and compare it with Algorithm 1.

Algorithm 2 (Least-Squares Function Approximation Algorithm)

Input: J_0, m, H, feature vectors $\{\phi(i)\}_{i \in \mathcal{S}}, \phi(i) \in \mathbb{R}^d$, and subsets $\mathcal{D}_k \subseteq \mathcal{S}, k = 0, 1, \ldots$. Here, $\mathcal{D}_k$ is the set of states at which we evaluate the current policy at iteration k.

1: Let $k = 0$.
2: Let μ_{k+1} be such that $\|T^H J_k - T_{\mu_{k+1}} T^{H-1} J_k\|_\infty \leq \varepsilon_{LA}$.
3: Compute $\hat{J}^{\mu_{k+1}}(i) = T_{\mu_{k+1}}^m T^{H-1}(J_k)(i) + w_{k+1}(i)$ for $i \in \mathcal{D}_k$.
4: Choose θ_{k+1} to solve

$$\min_\theta \sum_{i \in D_k} ((\Phi\theta)(i) - \hat{J}^{\mu_{k+1}}(i))^2, \tag{5}$$

where Φ is a matrix whose rows are the feature vectors.
5: $J_{k+1} = \Phi\theta_{k+1}$.
6: Set $k \leftarrow k + 1$. Go to (2).

3.1. Approximate PI with Linear Value Function Approximation

Our main algorithm is described in Algorithm 2. We now explain our algorithm and the associated notation in detail. For more on the notations used, see Table A.1 in Appendix A. Because of the use of function approximation, our algorithm is an approximation to PI with lookahead. At each iteration index, say k, we have an estimate of the value function, which we denote by J_k. To obtain J_{k+1}, we perform a lookahead to improve the value function estimate at a certain number of states

(denoted by $\mathcal{D}_k$), which can vary with each iteration. For example, $\mathcal{D}_k$ could be chosen as the states visited when performing a tree search to approximate the lookahead process. During the lookahead process, we note that we will also obtain an H-step lookahead policy, which we denote by μ_{k+1}. As noted in Section 1, the computation of $T^{H-1}(J_k)(i)$ for $i \in \mathcal{D}_k$ in Step 3 of Algorithm 2 may be computationally infeasible; however, as mentioned in Efroni et al. (2019), techniques such as MCTS are employed in practice to approximately estimate $T^{H-1}(J_k)(i)$. In this paper, we model the fact that lookahead cannot be performed exactly because of the associated computational complexity by allowing an error in the lookahead process, which we denote by ε_{LA} in Step 2 of Algorithm 2. The use of ε_{LA} is similar to the work of Efroni et al. (2019).

We obtain estimates of $J^{\mu_{k+1}}(i)$ for $i \in \mathcal{D}_k$, which we call $\hat{J}^{\mu_{k+1}}(i)$. To obtain an estimate of $J^{\mu_{k+1}}(i)$, we perform an m-step rollout with policy μ_{k+1} and obtain a noisy version of $T^m_{\mu_{k+1}} T^{H-1} J_k(i)$ for $i \in \mathcal{D}_k$. We also model the approximation error in the rollout by adding noise (denoted by $w_{k+1}(i)$ in Step 3 of Algorithm 2) to the return (result of the rollout; see Section 2) computed at the end of this step. In order to estimate the value function for states not in $\mathcal{D}_k$, we associate with each state $i \in \mathcal{S}$ a feature vector $\phi(i) \in \mathbb{R}^d$, where typically, $d << |\mathcal{S}|$. The matrix composed of the feature vectors as rows is denoted by Φ. We use those estimates to find the best-fitting $\theta \in \mathbb{R}^d$: that is,

$$\min_{\theta} \sum_{i \in D_k} ((\Phi\theta)(i) - \hat{J}^{\mu_{k+1}}(i))^2.$$

The solution to the minimization problem is denoted by θ_{k+1}. The algorithm then uses θ_{k+1} to obtain $J_{k+1} = \Phi\theta_{k+1}$. The process then repeats. This step of our algorithm differs from the algorithm in Efroni et al. (2019) in that the algorithm in Efroni et al. (2019) does not assume any particular technique for computing the estimate of $J^{\mu_{k+1}}$. It merely assumes the existence of some δ such that the distance from the estimate of $\hat{J}^{\mu_{k+1}}$ to $J^{\mu_{k+1}}$ is less than δ. We will show that the results of Efroni et al. (2019) change drastically when linear function approximation is employed to estimate $J^{\mu_{k+1}}$. Additionally, note that to compute $\hat{J}^{\mu_{k+1}}(i)$, we obtain noisy estimates of $T^m_{\mu_{k+1}} T^{H-1} J_k(i)$ for $i \in \mathcal{D}_k$. Another alternative is to instead obtain noisy estimates of $T^m_{\mu_{k+1}} J_k(i)$ for $i \in \mathcal{D}_k$. It was shown in Efroni et al. (2019) that the former option is preferable. Thus, we have chosen to use this computation in our algorithm as well. However, we will show in Appendix D that the algorithm also has bounded error, which becomes small if m is chosen to be sufficiently large.

Remark 1. We note that $\mu_{k+1}(i)$ in Step 2 of Algorithm 2 does not have to be computed for all states $i \in \mathcal{S}$. The actions $\mu_{k+1}(i)$ have to be computed only for those $i \in \mathcal{S}$ that are encountered in the rollout step of the algorithm (Step 3 of Algorithm 2).

3.1.1. Computational Considerations. We would like to note that m-step return and H-step lookahead are not algorithms that we propose to improve computational tractability. They are algorithms that are used in practice, and our goal is to point out why they are important in RL training. We will now attempt to explain why each of these ideas is used in practice. In the case of chess, for example, Shannon estimated the number of states to be approximately 10^{120}. So, to implement the policy evaluation step exactly, one has to perform the inversion of matrix of size $10^{120} \times 10^{120}$ or perform a fixed-point iteration of an operator represented by a $10^{120} \times 10^{120}$ matrix. Compared with this, even m of the order of several hundred steps (or even much more) is much more computationally efficient. Regarding H-step lookahead, this could indeed be a computational bottleneck. As mentioned earlier, the worst-case complexity can be exponential as shown in Shah et al. (2020a). However, there are practical implementations of lookahead that are efficient and perform well in practice. See Winnicki and Srikant (2023) for more on efficient implementations of lookahead. Our goal is not to argue the computational efficiency of these ideas but to understand why these ideas are important to ensure boundedness of errors given the fact that computationally efficient approximate implementations already exist in practice. In particular, in contrast to prior works, in our paper we have shown that, without these ideas, the algorithms used in practice may even fail to converge.

To analyze Algorithm 2, we make the following assumption, which states that we explore a sufficient number of states during the policy evaluation phase at each iteration and that the noise is bounded.

Assumption 1. *For each* $k \geq 0$, $\text{rank}\,\{\phi(i)\}_{i \in D_k} = d$. *Additionally, assume that the noise* w_k *is bounded. For some* $\varepsilon_{PE} > 0$, *the noise in policy evaluation satisfies* $\|w_k\|_\infty \leq \varepsilon_{PE} \ \forall k$.

Using Assumption 1, J_{k+1} can be written as

$$J_{k+1} = \Phi\theta_{k+1} = \underbrace{\Phi(\Phi_{\mathcal{D}_k}^\top \Phi_{\mathcal{D}_k})^{-1} \Phi_{\mathcal{D}_k}^\top \mathcal{P}_k}_{=:\mathcal{M}_{k+1}} \hat{J}^{\mu_{k+1}}, \tag{6}$$

where $\Phi_{\mathcal{D}_k}$ is a matrix whose rows are the feature vectors of the states in $\mathcal{D}_k$ and $\mathcal{P}_k$ is a matrix of zeros and ones such that $\mathcal{P}_k \hat{J}^{\mu_{k+1}}$ is a vector whose elements are a subset of the elements of $\hat{J}^{\mu_{k+1}}$ corresponding to $\mathcal{D}_k$. Note that $\hat{J}^{\mu_{k+1}}(i)$ for $i \notin \mathcal{D}_k$ does not affect the algorithm, so we can define $\hat{J}^{\mu_{k+1}}(i) = T^m_{\mu_{k+1}} T^{H-1} J_k(i)$ for $i \notin \mathcal{D}_k$.

Written concisely, our algorithm is as follows:

$$J_{k+1} = \mathcal{M}_{k+1}(T^m_{\mu_{k+1}} T^{H-1} J_k + w_k), \tag{7}$$

where μ_{k+1} is defined in Step 2 of Algorithm 2. Because $w_k(i)$ for $i \notin \mathcal{D}_k$ does not affect the algorithm, we define $w_k(i) = 0$ for $i \notin \mathcal{D}_k$.

We now present a counterexample to show that applying linear value function approximation to approximate PI is not a straightforward application of the bounds in Bertsekas (2019) and Efroni et al. (2019). In the counterexample, we give an MDP, which uses an m-step return to evaluate greedy policies at several states of the state space and linear value function approximation to estimate the value functions corresponding to the greedy policy at the rest of the states. The iterates diverge, which shows that more work needs to be done to understand how to incorporate linear value function approximation into approximate PI.

3.2. Counterexample

Even though in practice, J^{μ_k} is what we are interested in, the values J_k computed as part of our algorithm should not go to ∞ as the algorithm uses the values of J_k to compute J^{μ_k}, so divergence of J_k can result in inaccurate computations of values of J^{μ_k}. Additionally, divergence of J_k would result in a numerically unstable algorithm, which is also undesirable. Here, we show that J_k can become unbounded.

The example we use is depicted in Figure 1. There are two policies, μ^a and μ^b, and the transitions are deterministic under the two policies. The rewards are deterministic and only depend on the states. The rewards associated with states are denoted by $r(x_1)$ and $r(x_2)$, with $r(x_1) > r(x_2)$. Thus, the optimal policy is μ^a. We assume scalar features $\phi(x_1) = 1$ and $\phi(x_2) = 2$.

We fix $H = 1$. The MDP follows policy μ^a when

$$J_k(x_1) > J_k(x_2) \Rightarrow \theta_k > 2\theta_k.$$

Thus, as long as $\theta_k > 0$, the lookahead policy will be μ_b.

We will now show that θ_k increases at each iteration when $\frac{6}{5}\alpha^m > 1$. We assume that $\theta_0 > 0$ and $\mathcal{D}_k = \{1, 2\}$ $\forall k$. At iteration $k+1$, suppose $\mu_{k+1} = \mu^b$, and our $\hat{J}^{\mu_{k+1}}(i)$ for $i = 1, 2$ are as follows:

$$\hat{J}^{\mu_{k+1}}(1) = r(x_1) + \sum_{i=1}^{m-1} r(x_1)\alpha^i + 2\alpha^m\theta_k,$$

$$\hat{J}^{\mu_{k+1}}(2) = r(x_2) + \sum_{i=1}^{m-1} r(x_2)\alpha^i + 2\alpha^m\theta_k.$$

Thus, from Step 5 of Algorithm 2,

$$\theta_{k+1} = \arg\min_{\theta} \sum_{i=1}^{2} ((\Phi\theta)(i) - \hat{J}^{\mu_{k+1}}(i))^2$$

$$\Rightarrow \theta_{k+1} = \frac{\sum_{i=1}^{m-1} \alpha^i r(x_1)}{5} + \frac{2\sum_{i=1}^{m-1} \alpha^i r(x_2)}{5} + \frac{6\alpha^m\theta_k}{5}$$

$$\Rightarrow \theta_{k+1} > \frac{6}{5}\alpha^m\theta_k.$$

Thus, because $\theta_0 > 0$, when $\frac{6}{5}\alpha^m\theta_k$, θ_k goes to ∞.

It is worth noting that, even though J^{μ_k} is always bounded, the fact that J_k diverges means that the algorithm cannot be implemented in a numerically stable manner. The discussion can be summarized in the following claim.

Claim 1. *There exists an MDP with a linear feature vector representation for which modified PI diverges.*

An interpretation of this result is that modified policy iteration in the presence of linear function approximation is not a straightforward extension of modified policy iteration with convergence guarantees. In fact, the algorithm may diverge unlike modified policy iteration, which always converges. In Section 3.4, we introduce lookahead as a remedy to this divergence.

3.3. Approximate PI with Time-Dependent Policy Evaluation Error

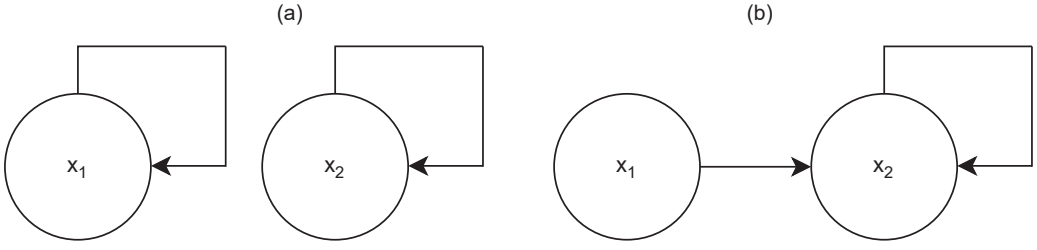
Algorithm 3 (Modified PI with Lookahead and Function Approximation)

Input: θ_0, m, H.

1: Let $k = 0$.

2: Let μ_{k+1} be such that $\|T^H J_k - T_{\mu_{k+1}} T^{H-1} J_k\|_\infty \leq \varepsilon_{LA}$.

Figure 1. An Example Illustrating the Necessity of the Condition in Theorem 1



Notes. (a) μ^a. (b) μ^b.

3: Compute θ_{k+1} such that $\hat{J}^{\mu_{k+1}} := \Phi\theta_{k+1}$ satisfies the following:

$$\|\hat{J}^{\mu_{k+1}} - J^{\mu_{k+1}}\|_\infty \leq \delta_k.$$

4: $J_{k+1} = \hat{J}^{\mu_{k+1}}$.
5: Set $k \leftarrow k+1$. Go to (2).

Before we present our main results, we first obtain bounds for modified PI with lookahead and time-varying bounds in the policy evaluation error. The algorithm we analyze in this section is described in Algorithm 3. The algorithm in Efroni et al. (2019) (Algorithm 1) is similar to Algorithm 3 except that at time k, the work of Efroni et al. (2019) assumes a constant bound in the policy evaluation error, δ, and in Algorithm 1, we assume that the policy evaluation error is upper bounded by time-dependent δ_k. Then, we assume that δ_k is of the following form: $\delta_k \leq \beta^k\delta_0 + \mu$ when $0 < \beta < 1$. The bounds are given in Proposition 1. In Section 3.4, we obtain values of β and μ corresponding to Algorithm 2, approximate PI with linear value function approximation, and lookahead. We further extend the results to incorporate the use of gradient descent in Section 4.

We now obtain a bound on the iterates in Algorithm 3 as follows.

Proposition 1.

$$\|J^{\mu_k} - J^*\|_\infty \leq \frac{\alpha^{k(H)}}{1-\alpha} + \sum_{\ell=0}^{k-1} \frac{\alpha^{(k-\ell-1)(H)} 2\alpha^H \delta_\ell}{1-\alpha} + \frac{\varepsilon_{LA}}{(1-\alpha)(1-\alpha^{H-1})}.$$

Furthermore, when

$$\delta_k \leq \beta^k\delta_0 + \mu \quad \text{for } 0 < \beta < 1, 0 < \mu. \tag{8}$$

Then,

$$\|J^{\mu_k} - J^*\|_\infty \leq \underbrace{\frac{\alpha^{k(H)}}{1-\alpha} + \frac{2\alpha^H}{1-\alpha} k \max(\alpha^{H-1}, \beta)^{k-1}\delta_0}_{=: \text{finite-time component}} + \underbrace{\frac{2\alpha^H\mu + \varepsilon_{LA}}{(1-\alpha)(1-\alpha^H)}}_{=: \text{asymptotic component}}.$$

Taking limits on both sides, when $0 < \beta < 1$, we have

$$\limsup_{k\to\infty} \|J^{\mu_k} - J^*\|_\infty \leq \frac{2\alpha^H\mu + \varepsilon_{LA}}{(1-\alpha)(1-\alpha^H)}.$$

3.4. Approximate PI with Linear Value Function Approximation and Lookahead

To apply Proposition 1 to Algorithm 2, we have to compute the parameters β and μ in the proposition. In Appendix C, we show that β and μ for Algorithm 2 are given by

$$\beta := \alpha^{m+H-1}\delta_{FV}$$
$$\mu := \frac{\tau}{1-\beta}, \tag{9}$$

where $\tau := \frac{\alpha^m + \alpha^{m+H-1}}{1-\alpha}\delta_{FV} + \delta_{app} + \delta_{FV}\varepsilon_{PE}$.

Using (9) along with Proposition 1, we now state Theorem 1, which characterizes the role of lookahead (H) and return (m) on the convergence of approximate PI with function approximation.

Theorem 1. *Suppose that m and H satisfy $m + H - 1 > \log(\delta_{FV})/\log(1/\alpha)$, where*

$$\delta_{FV} := \sup_k \|\mathcal{M}_k\|_\infty = \sup_k \|\Phi(\Phi_{\mathcal{D}_k}^\top \Phi_{\mathcal{D}_k})^{-1}\Phi_{\mathcal{D}_k}^\top \mathcal{P}_k\|_\infty.$$

Then, under Assumption 1, the following holds for Algorithm 2:

$$\|J^{\mu_k} - J^*\|_\infty \leq \underbrace{\frac{\alpha^{k(H)}}{1-\alpha} + \frac{2\alpha^H\|J^{\mu_0} - J_0\|_\infty}{1-\alpha} k \max(\alpha^H, \beta)^{k-1}}_{\text{finite-time component}} + \underbrace{\frac{2\alpha^H\frac{\tau}{1-\beta} + \varepsilon_{LA}}{(1-\alpha^H)(1-\alpha)}}_{\text{asymptotic component}}, \tag{10}$$

where

$$\tau := \frac{\alpha^m + \alpha^{m+H-1}}{1-\alpha}\delta_{FV} + \delta_{app} + \delta_{FV}\varepsilon_{PE},$$
$$\beta := \alpha^{m+H-1}\delta_{FV},$$

and

$$\delta_{app} := \sup_{k,\mu_k} \|\mathcal{M}_k J^{\mu_k} - J^{\mu_k}\|_\infty.$$

Remark 2. The tightness of the condition $m + H - 1 > \log(\delta_{FV})/\log(1/\alpha)$ can be observed in our counterexample in Section 3.2, where it can easily be shown that when $m + H - 1 \leq \log(\delta_{FV})/\log(1/\alpha)$, the algorithm diverges.

The proof of Theorem 1 follows easily from Proposition 1. In Appendix E, we give corresponding bounds on the iterates J_k in the algorithm. We now provide an interpretation of Theorem 1, including δ_{app} and δ_{FV}. δ_{app} represents the maximum error over k when feature vectors corresponding to the states in $\mathcal{D}_k$ are used to construct an estimate of J^{μ_k} based on $J^{\mu_k}(s)$ for $s \in \mathcal{D}_k$. In other words, δ_{app} is error because of function approximation. δ_{FV} is a function of the feature vectors. Although it is not straightforward to characterize δ_{FV} for different choices of

function approximation, δ_{FV} can be quantified for several choices of feature vectors. First, in the tabular setting (i.e., one-hot encoded feature vectors), when all states are visited at each iteration, $\delta_{FV} = 1$. Next, we consider the case of state aggregation in Bertsekas (2019, section 6.1). In this case, under Assumption 1, $\delta_{FV} = 1$. To show this, we provide details for the special case of two "representative" states (i.e., the case where the feature vectors are $[0,1]^\top$ and $[1,0]^\top$). The idea can be easily extended to cases with more than two representative states. In the case of two representative states, it can be shown that $(\Phi_{Dk}^\top \Phi_{Dk})^{-1} = \begin{bmatrix} 1/N_1 & 0 \\ 0 & 1/N_2 \end{bmatrix}$, where N_1 is the number of items in $\mathcal{D}_k$ belonging to the first representative state and N_2 is the number of items in $\mathcal{D}_k$ belonging to the second representative state. Note that because of Assumption 1, N_1 and N_2 are nonzero. Hence, the ith row of $\Phi[\Phi_{\mathcal{D}_k}^\top \Phi_{\mathcal{D}_k}]^{-1} = \phi(s_i)\frac{1}{N_i}$, where s_i is the state corresponding to the ith row.

It is straightforward to show that the jth column of $\Phi_{Dk}\mathcal{P}_k$ is equal to $\phi(s_j)\mathbb{1}_{j\in\mathcal{D}_k}$, where s_j is the state corresponding to the jth column.

Thus, we have that

$$[\Phi(\Phi_{\mathcal{D}_k}^\top \Phi_{\mathcal{D}_k})^{-1}\Phi_{\mathcal{D}_k}^\top \mathcal{P}_k]_{ij} = \frac{1}{N_i}\mathbb{1}_{j\in\mathcal{D}_k}.$$

Hence, every sum of row components of $\Phi(\Phi_{\mathcal{D}_k}^\top \Phi_{\mathcal{D}_k})^{-1}\Phi_{\mathcal{D}_k}^\top \mathcal{P}_k$ is equal to one, and thus, $\|\Phi(\Phi_{\mathcal{D}_k}^\top \Phi_{\mathcal{D}_k})^{-1}\Phi_{\mathcal{D}_k}^\top \mathcal{P}_k\|_\infty = 1$.

In general, it is hard to characterize δ_{FV}. However, when the terms of (10) are written out, the coefficients of δ_{FV} are α^{m+H-1}, $2\alpha^{m+2H-1}$, $2\frac{\alpha^{m+H}+\alpha^{m+2H-1}}{1-\alpha}$, and $2\alpha^H\delta_{FV}\varepsilon_{PE}$, where ε_{PE} is noise from the rollout. Thus, appropriately chosen m and H can offset the effect of δ_{FV}.

In light of our interpretations of δ_{app} and δ_{FV}, Theorem 1 can then be used to make the following observation; how close J^{μ_k} is to J^* depends on four factors—the representation power of the feature vectors and the feature vectors themselves ($\delta_{app}, \delta_{FV}$), the amount of lookahead (H), the extent of the rollout (m), and the approximation in the policy determination and policy evaluation steps (ε_{LA} and ε_{PE}). Additionally, Theorem 1 shows that although $\|J^{\mu_k} - J^*\|_\infty$ depends on the function approximation error (δ_{app}) and the feature vectors (δ_{FV}), the effect of these terms diminishes exponentially with increased H, with the exception of the tree search error (ε_{LA}). Further, it is easy to see that lookahead and rollout help mitigate the effect of feature vectors and their ability to represent the value functions.

3.4.1. Comparison with Prior Works.

We observe that the models studied in Lagoudakis and Parr (2003), Bertsekas (2019), and Efroni et al. (2019) are all special cases of model studied in Theorem 1.

- Specifically, if we set $\varepsilon_{PE} = 0$, $m = \infty$, and $H = 1$ and consider the tabular case, we get the models studied in Lagoudakis and Parr (2001) and Bertsekas (2019). We note that Lagoudakis and Parr (2001) is motivated by the linear value function approximation setting, but the errors because of function approximation are not explicitly modeled.
- On the other hand, if we set $\varepsilon_{PE} = 0$ and $m = \infty$ but allow an arbitrary H, we get the model in Efroni et al. (2019). Our work quantifies the effect of varying m on the convergence of Algorithm 1.
- The most important detail in our model that makes it different from the other models is the fact that we model the errors because of function approximation, which leads to convergence issues not noticed in the other papers.

In Bertsekas (2021), it is noted that in reinforcement learning, to play computer games or board games, it is not uncommon during training to get a relatively crude estimate of the value function, which is improved by lookahead and m-step return during actual game play. Our analysis would also apply to this situation; we have not explicitly differentiated between training and game play in our analysis.

4. Extension to Gradient Descent

Algorithm 4 (Gradient Descent Algorithm)

Input: θ_0, m, H, feature vectors $\{\phi(i)\}_{i\in\mathcal{S}}, \phi(i) \in \mathbb{R}^d$, and $\mathcal{D}_k$, which is the set of states for which we evaluate the current policy at iteration k.

1: $k = 0, J_0 = \Phi\theta_0$.

2: Let μ_{k+1} be such that $\|T^H J_k - T_{\mu_{k+1}} T^{H-1} J_k\|_\infty \le \varepsilon_{LA}$.

3: Compute $\hat{J}^{\mu_{k+1}}(i) = T_{\mu_{k+1}}^m T^{H-1} J_k(i) + w_{k+1}(i)$ for $i \in \mathcal{D}_k$.

4: $\theta_{k+1,0} := \theta_k$. For $\ell = 1, 2, \ldots, \eta$, iteratively compute the following:

$$\theta_{k+1,\ell} = \theta_{k+1,\ell-1} - \gamma \nabla_\theta c(\theta; \hat{J}^{\mu_{k+1}})|_{\theta_{k+1,\ell-1}}, \tag{11}$$

where

$$c(\theta; \hat{J}^{\mu_{k+1}}) := \frac{1}{2}\sum_{i\in\mathcal{D}} ((\Phi\theta)(i) - \hat{J}^{\mu_{k+1}}(i))^2,$$

and Φ is a matrix whose rows are the feature vectors.

5: Define

$$\theta_{k+1} = \theta_{k+1,\eta},$$

and set

$$J_{k+1} = \Phi\theta_{k+1}.$$

6: Set $k \leftarrow k+1$. Go to (2).

Solving the least-squares problem in Algorithm 2 involves a matrix inversion, which can be computationally difficult. So, this step is often replaced by a few steps of gradient descent that are performed on the least-squares objective. Here, we assume that we

perform η steps of gradient descent with step size γ at each iteration k, where the gradient is the gradient of the least-squares objective in (5).

The gradient descent-based algorithm is presented in Algorithm 4. When γ is sufficiently small and η is sufficiently large, we have convergence to an asymptotic error, assuming that m and H are sufficiently large. When we increase η, our asymptotic error becomes smaller until it reaches the asymptotic error of the least-squares algorithm (i.e., when $\eta \to \infty$, we recover the asymptotic error of Algorithm 2).

To apply Proposition 1 to Algorithm 4, we have to first identify the parameters β and μ for this algorithm. We make the following assumption.

Assumption 2. *γ, m, η, and H satisfy*

$$\gamma < \frac{1}{d \inf_k \|\Phi_{\mathcal{D}_k}^\top \Phi_{\mathcal{D}_k}\|_\infty^2},$$

$$m + H > 1 + \log(2\delta_{FV})/\log(1/\alpha),$$

and

$$\eta > \log\left(\frac{3\sqrt{|S|}\|\Phi\|_\infty}{\sigma_{\min,\Phi}}\right)/\log(1/\alpha_{GD,\gamma}),$$

where $\alpha_{GD,\gamma} := \sup_k \max_i |1 - \gamma\lambda_i(\Phi_{\mathcal{D}_k}^\top \Phi_{\mathcal{D}_k})|$, in which λ_i denotes the ith-largest eigenvalue of a matrix and $\sigma_{\min,\Phi}$ is the smallest singular value in the singular value decomposition of Φ.

Under Assumption 2, we can obtain β and μ for Algorithm 4. In Appendix F, we show that β and μ are given by

$$\beta = \alpha^{m+H-1}\delta_{FV} + \frac{\sqrt{|S|}\|\Phi\|_\infty}{\sigma_{\min,\Phi}}\alpha_{GD,\gamma}^\eta(\alpha^{m+H-1}\delta_{FV} + 1),$$

$$\mu = \frac{\tau}{1-\beta}, \tag{12}$$

where $\tau := \left(1 + \frac{\sqrt{|S|}\|\Phi\|_\infty}{\sigma_{\min,\Phi}}\alpha_{GD,\gamma}^\eta\right)\left(\frac{\alpha^m + \alpha^{m+H-1}}{1-\alpha}\delta_{FV} + \delta_{app} + \delta_{FV}\varepsilon_{PE}\right) + \frac{\sqrt{|S|}\|\Phi\|_\infty}{(1-\alpha)\sigma_{\min,\Phi}}\alpha_{GD,\gamma}^\eta$.

Using (12) along with Proposition 1, we now state our theorem, which characterizes the error in using gradient descent in approximate PI with linear value function approximation and lookahead. We remark that any term undefined in Theorem 2 is assumed to have the same definition as in Theorem 1.

Theorem 2. *Suppose that m and H satisfy $m + H - 1 > \log(2\delta_{FV})/\log(1/\alpha)$, where*

$$\delta_{FV} := \sup_k \|\mathcal{M}_k\|_\infty = \sup_k \|\Phi(\Phi_{\mathcal{D}_k}^\top \Phi_{\mathcal{D}_k})^{-1}\Phi_{\mathcal{D}_k}^\top \mathcal{P}_k\|_\infty,$$

in which the norm is the induced matrix norm defined in Section 2. Then, under Assumptions 1 and 2, the following holds:

$$\|J^{\mu_k} - J^*\|_\infty \le \underbrace{\frac{\alpha^{kH}}{1-\alpha} + \frac{2\alpha^H\|J^{\mu_0} - J_0\|_\infty}{1-\alpha}k\max(\alpha^H, \beta)^{k-1}}_{\text{finite-time component}} + \underbrace{\frac{2\alpha^H\frac{\tau}{1-\beta} + \varepsilon_{LA}}{(1-\alpha^H)(1-\alpha)}}_{\text{asymptotic component}}, \tag{13}$$

where

$$\tau := \left(1 + \frac{\sqrt{|S|}\|\Phi\|_\infty}{\sigma_{\min,\Phi}}\alpha_{GD,\gamma}^\eta\right)\left(\frac{\alpha^m + \alpha^{m+H-1}}{1-\alpha}\delta_{FV} + \delta_{app} + \delta_{FV}\varepsilon_{PE}\right) + \frac{\sqrt{|S|}\|\Phi\|_\infty}{(1-\alpha)\sigma_{\min,\Phi}}\alpha_{GD,\gamma}^\eta,$$

$$\beta := \alpha^{m+H-1}\delta_{FV} + \frac{\sqrt{|S|}\|\Phi\|_\infty}{\sigma_{\min,\Phi}}\alpha_{GD,\gamma}^\eta(\alpha^{m+H-1}\delta_{FV} + 1),$$

and

$$\delta_{app} := \sup_{k,\mu_k} \|\mathcal{M}_k J^{\mu_k} - J^{\mu_k}\|_\infty.$$

Theorem 2 follows directly from Proposition 1 when β and μ are defined in (12).

Remark 3. Note that as $\eta \to \infty$, (i.e., the number of steps of gradient descent becomes very large), the error becomes the same as that of Algorithm 2.

Remark 4. Consider any ε such that $0 < \varepsilon < 1$. It is straightforward to see that when

$$m > \frac{\log\left(\left(\frac{8\delta_{FV}}{1-\alpha}\right)/\varepsilon\right)}{\log(1/\alpha)}$$

$$\eta > \frac{\log\left(\left(\frac{4\sqrt{|S|}\|\Phi\|_\infty}{\sigma_{\min,\Phi}(1-\alpha)}\right)/\varepsilon\right)}{\log(1/\alpha_{GD,\gamma})},$$

and

$$H > \frac{\log\left[\frac{\frac{32}{7}\alpha^H\left[\frac{5}{4}\left(\frac{1}{4} + \delta_{app} + \delta_{FV}\varepsilon_{PE}\right) + \frac{1}{4}\right]}{(1-\alpha)^2}/\varepsilon\right]}{\log(1/\alpha)},$$

ignoring the error because of lookahead, the asymptotic error will be less than or equal to ε. Notice that the parameters η, H, and m depend on $\log|\mathcal{S}|$ instead of $|\mathcal{S}|$ or $\sqrt{|\mathcal{S}|}$.

5. Numerical Results

We test our algorithms on a grid world problem using the same grid world problem as in Efroni et al. (2018a, 2019).

For our simulations, we assume a deterministic grid world problem played on an $N \times N$ grid. The states are the squares of the grid, and the actions are {'up', 'down', 'right', 'left', and 'stay'}, which move the agent in the prescribed direction, if possible. In each experiment, a goal state is chosen uniformly at random to have a reward of one, whereas each other state has a fixed reward drawn uniformly from $[-0.1, 0.1]$. Unless otherwise mentioned, for the duration of this section, $n = 25$ and $\alpha = 0.9$.

In order to perform linear function approximation, we prescribe a feature vector for each state. In this section, we focus on three particular choices.

1. Random feature vectors. Each entry of the matrix Φ is an independent $\mathcal{N}(0,1)$ random variable.

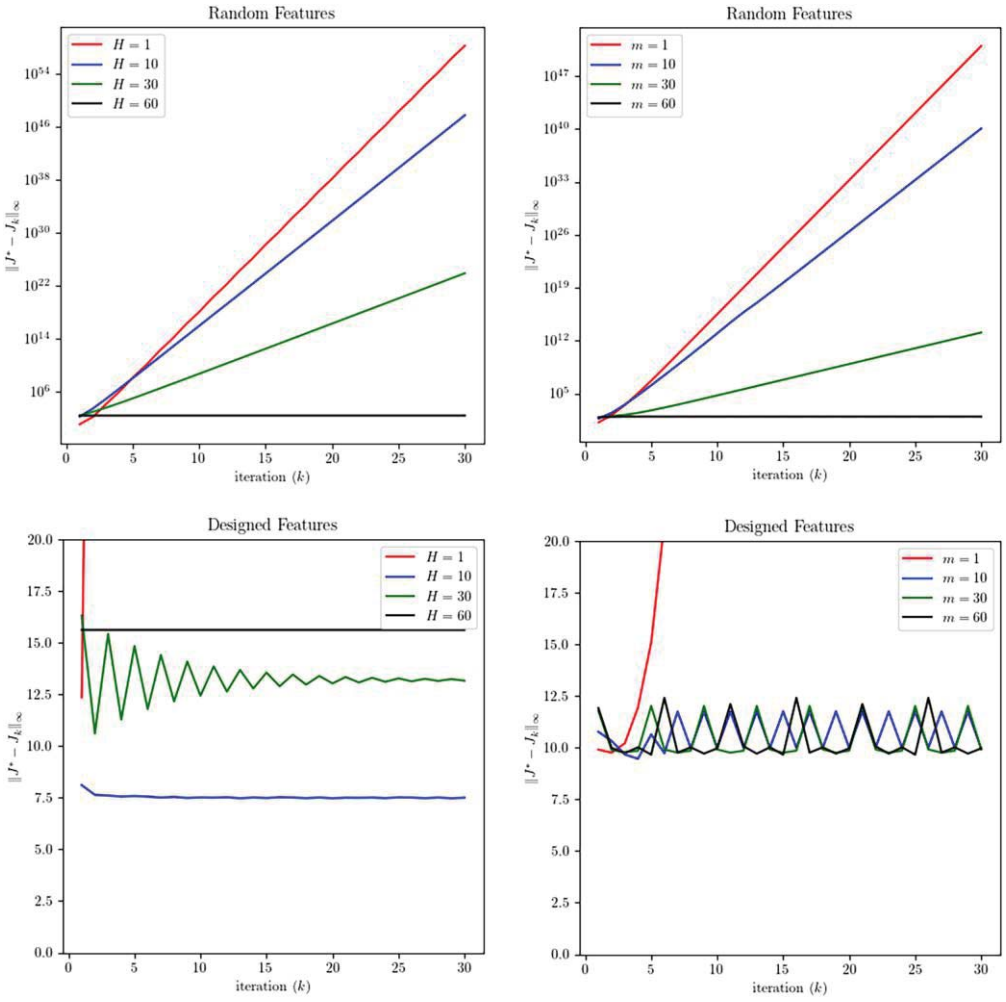
2. Designed feature vectors. The feature vector for a state with coordinates (x, y) is $[x, y, d, 1]^T$, where d is the number of steps required to reach the goal from state (x, y).

3. Indicator vectors. The feature vector for each state i is an N^2-dimensional indicator vector where only the ith entry is nonzero.

Recall that our theorems suggest that the amount of lookahead and return depends on the choice of the feature vectors. Our experiments support this observation as well. The amount of lookahead and m-step return required is high (often over 30) for random feature vectors, but we are able to significantly reduce the amount required by using the designed feature vectors, which better represent the states.

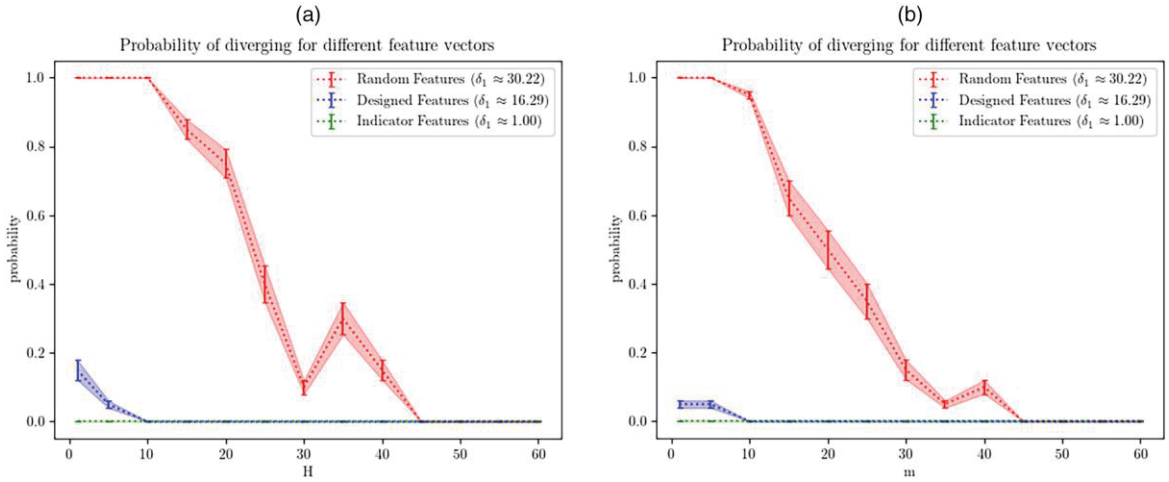
We test Algorithm 2 in each of our experiments using a starting state of $J_0 = \theta_0 = 0$. All plots in this section graph an average over 20 trials, where each trial has a fixed random choice of $\mathcal{D}_k$, the set of states used for

Figure 2. (Color online) Value of J_k as m and H Increase for Various Feature Vectors



Notes. (Upper panels) For random feature vectors, as m and H increase, the value J_k eventually stops diverging. (Lower panels) For designed feature vectors, smaller amounts of lookahead and m-step return are needed to prevent J_k from diverging.

Figure 3. (Color online) We Plot the Probability That $\|J_k - J^*\|_\infty$ Diverges as a Function of H and m



Notes. For the first plot, $m = 3$, and for the second plot, $H = 3$. In both cases, the algorithm never diverges after $H + m$ is large enough, although a smaller amount of lookahead or m-step return is needed for the designed feature vectors. (a) Varying H. (b) Varying m.

policy evaluation. Error bars show the standard deviation of the mean.

5.1. The Effect of *m* and *H* on Convergence

In Figure 2, we showed how H and m affect convergence of the iterates J_k to J^*. When m and H are small, the value of J_k sometimes diverges. If the value diverges for even one trial, then the average over trials of $\|J_k - J^*\|_\infty$ also increases exponentially with k. However, if the average converges for all trials, then the plot is relatively flat. The m or H required for convergence depends on the parameter δ_{FV} defined in Theorem 1. Over 20 trials, the average values of δ_{FV} for each of our choices of feature vectors are 30.22, 16.29, and 1.0, respectively. As shown in our counterexample, in general, one needs $m + H - 1 > \log(\delta_{FV})/\log(1/\alpha)$ for convergence. However, in specific examples, it is possible for convergence to occur for smaller values of $m + H$. For example, in our grid word model, $\frac{\log(16.29)}{\log(1/0.9)} \approx 26.5$, but we will observe that such a large amount of $m + H$ is not required for convergence.

In Figure 2, it is difficult to see how H and m affect the probability of divergence as a function of the representative states chosen to be sampled. Therefore, we introduce Figure 3. These plots show the proportion of trials in which the distance $\|J_k - J^*\|_\infty$ exceeded 10^5 after 30 iterations of our algorithm. As expected, the algorithm never diverges for indicator vectors as our algorithm is then equivalent to the tabular setting. The designed feature vectors clearly require a much smaller amount of lookahead or m-step return, well below the amount predicted by the average δ_{FV} of 16.29. However, no matter the choice of feature vectors, we will eventually prevent our algorithm from diverging with a large-enough value of $H + m$.

5.2. Convergence to the Optimal Policy

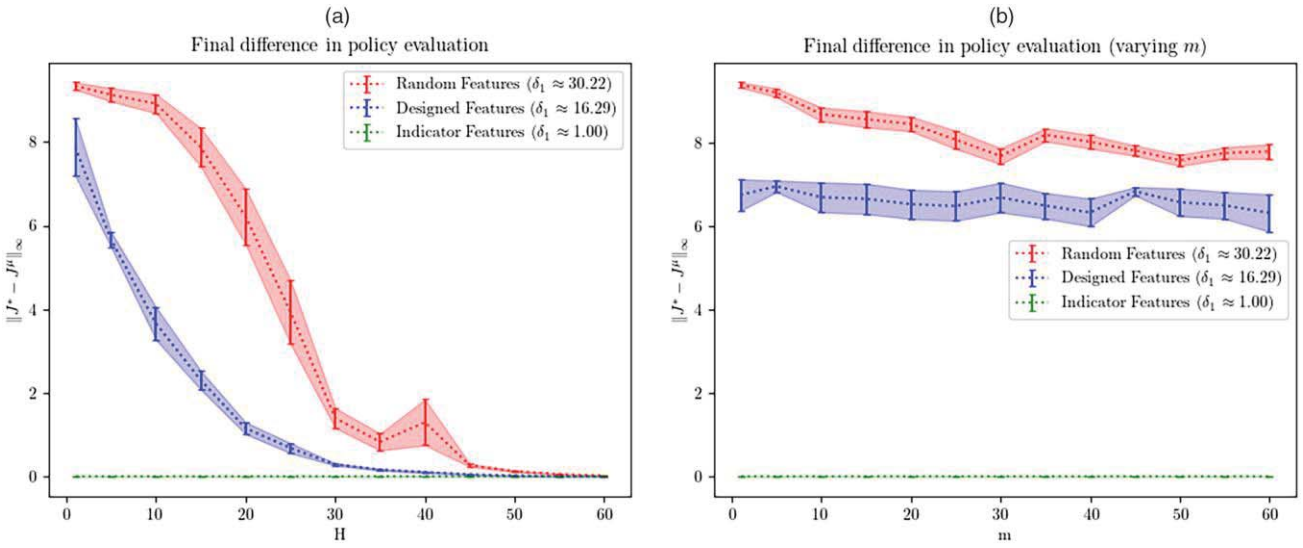
In Theorem 1, we show that as H increases, we converge to a policy μ_k that is closer to the optimal policy. In this section, we experimentally investigate the role of m and H on the final value of $\|J^{\mu_k} - J^*\|_\infty$. The results can be found in Figure 4. As predicted by theory, we do get closer to the optimal policy as H increases. However, increasing m does not help past a certain point, which is also consistent with the theory. Indeed, although μ_k is approaching the optimal policy μ^* as H increases, the iterates J_k are not converging to J^* because of error induced by function approximation. Increasing m improves the policy evaluation, but it cannot correct for this inherent error from approximating the value function. The figures also show the importance of good feature selection. In practice, this feature selection is done using neural networks, but analyzing this is beyond the scope of the paper. However, it should be noted that δ_{FV} somewhat captures this effect in our analysis.

Note that, in Figure 4, the plots corresponding to indicator feature functions converge very fast. This is because the indicator features correspond to no function approximation. Further, we note that m plays only a small role in controlling the error, whereas H plays a much larger role. This is consistent with the performance bounds in Theorem 1.

6. Conclusion

Practical RL algorithms that deal with large state spaces implement some form of approximate PI. In traditional

Figure 4. (Color online) We Plot the Final Value of $\|J^{\mu_k} - J^*\|_\infty$ After 30 Iterations



Notes. For the first plot, $m = 3$, and for the second plot, $H = 3$. As H increases, the final policy improves. With large-enough H, we obtain the optimal policy. However, past a certain point, increasing m is not helpful for finding a better policy. (a) Varying H. (b) Varying m.

analyses of approximate PI (for example, in Bertsekas 2019), it is assumed that there is an error in the policy evaluation step and an error in the policy improvement step. The work of Efroni et al. (2019) extends this analysis to incorporate lookahead policies, which mitigate the effects of function approximation. We provide a counterexample to show that incorporating linear value function into approximate PI is not straightforward as the iterates may diverge. In this paper, we seek to understand the role of linear value function approximation in the policy evaluation step and the associated changes that one has to make to the approximate PI algorithm (such as lookahead) to counteract the effect of function approximation. Our main conclusion is that lookahead mitigates the effects of function approximation, rollout, and the choice of specific feature vectors.

Possible directions for future work include the following.

- In game-playing applications, gradient descent is commonly used to estimate the value function, but temporal difference (TD) learning is used in other applications. It would be interesting to extend our results to the case of TD learning-based policy evaluation.
- Although neural networks are not linear function approximators, recent results on the neural tangent kernel (NTK) analysis of neural networks suggest that they can be approximated as linear combinations of basis functions (Jacot et al. 2018, Arora et al. 2019, Cao and Gu 2019, Du et al. 2019, Ji and Telgarsky 2019). Thus, to the extent that the NTK approximation is reasonable, our results can potentially shed light on why the combination of the representation capability of neural networks and tree search methods works well in practice, although further work is necessary to make this connection precise.

Acknowledgments

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration [Contract DE-NA0003525]. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the U.S. Government.

Appendix A. Notation

Table A.1. Notation

Notation	Definition
MDP	
α	Discount factor
J^*	Optimal value function
Indices	
k	Iteration index
ℓ	Gradient descent iteration index
θ_k	Provides estimate of optimal value function (i.e., $J_k = \Phi\theta_k$)
J_k	Estimate of optimal value function
d	Dimension of θ_k (i.e., $\theta_k \in \mathbb{R}^d$)
Policies	
μ	Policy
J^μ	Value function corresponding to a policy
μ^*	Optimal policy
μ_k	Policy at iteration k
Value operators/maps	
$T_\mu J$	Bellman operator for μ, $(T_\mu J)(s) = r(s,\mu(s)) + \alpha\sum_{j=1}^{\lvert\mathcal{S}\rvert} P_{sj}(\mu(s))J(j)\ \forall s \in \mathcal{S}$
TJ	Bellman optimality operator, $(TJ)(s) = \max_{a\in\mathcal{A}}\{r(s,a) + \alpha\sum_{j=1}^{\lvert\mathcal{S}\rvert} P_{sj}(a)J(j)\}\ \forall s \in \mathcal{S}$
Function approximation	
$\phi(s)$	Feature vector for state s
Φ	Matrix with rows that are the feature vectors
$\mathcal{D}_k$	States for which policy is evaluated at iteration k
$\Phi_{\mathcal{D}_k}$	Matrix with rows that are the feature vectors of the states in $\mathcal{D}_k$
$\mathcal{P}_k$	Matrix of zeros and ones such that $\mathcal{P}_k J$ is a vector with elements that are a subset of the elements of $\hat{J}^{\mu_k}$ corresponding to $\mathcal{D}_k$
$\mathcal{M}_k$	Projection matrix (i.e., given $\hat{J}^{\mu_k}(i), i \in \mathcal{D}_k$, $\Phi\theta_k = \mathcal{M}_k\hat{J}^{\mu_k}$, where $\theta_k = \arg\min_\theta \sum_{i\in\mathcal{D}_k}((\Phi\theta)(i) - J(i))^2$)
Error terms	
δ_{FV}	Feature vectors (i.e., $\delta_{FV} := \sup_k \lVert\mathcal{M}_k\rVert_\infty = \sup_k \lVert\Phi(\Phi_{\mathcal{D}_k}^\top\Phi_{\mathcal{D}_k})^{-1}\Phi_{\mathcal{D}_k}^\top\mathcal{P}_k\rVert_\infty$)
δ_{app}	Function approximation error (i.e., $\delta_{app} := \sup_{k,\mu_k} \lVert\mathcal{M}_k J^{\mu_k} - J^{\mu_k}\rVert_\infty$)
ε_{LA}	$\lVert T^H J_k - T_{\mu_{k+1}} T^{H-1} J_k\rVert_\infty \le \varepsilon_{LA}$
ε_{PE}	Noise in policy evaluation where $\lVert w_k\rVert_\infty \le \varepsilon_{PE}\ \forall k$
Algorithm 4	
η	Number of steps of gradient descent
γ	Gradient descent step size
$\alpha_{GD,\gamma}$	$\alpha_{GD,\gamma} := \sup_k \max_i \lvert 1 - \gamma\lambda_i(\Phi_{\mathcal{D}_k}^\top\Phi_{\mathcal{D}_k})\rvert$,
λ_i	ith-largest eigenvalue of a matrix
$\sigma_{\min,\Phi}$	Smallest singular value in the singular value decomposition of Φ

Appendix B. Proof of Proposition 1

The work of Efroni et al. (2019) shows that

$$\lVert J^{\mu_{k+1}} - J^*\rVert_\infty \le \alpha^H \lVert J^{\mu_k} - J^*\rVert_\infty + \frac{2\alpha^H\delta + \varepsilon_{LA}}{1-\alpha}. \tag{B.1}$$

Iterating over k,

$$\limsup_{k\to\infty} \lVert J^{\mu_k} - J^*\rVert_\infty \le \frac{2\alpha^H\delta + \varepsilon_{LA}}{(1-\alpha)(1-\alpha^H)},$$

which is a main result of Efroni et al. (2019). Suppose now that δ depends on k, and we call the sequence δ_k.

Starting from (B.1), we substitute δ_k for δ, and we get the following:

$$\lVert J^{\mu_{k+1}} - J^*\rVert_\infty \le \alpha^H \lVert J^{\mu_k} - J^*\rVert_\infty + \frac{2\alpha^H\delta_k + \varepsilon_{LA}}{1-\alpha}. \tag{B.2}$$

Iterating over k, we have

$$\begin{aligned}\lVert J^{\mu_k} - J^*\rVert_\infty &\le \alpha^{kH}\lVert J^{\mu_0} - J^*\rVert_\infty + \frac{2\alpha^H}{1-\alpha}\sum_{\ell=0}^{k-1}\frac{\alpha^{(k-\ell-1)(H-1)}2\alpha^H\delta_\ell + \varepsilon_{LA}}{1-\alpha}\\ &\le \frac{\alpha^{kH}}{1-\alpha} + \sum_{\ell=0}^{k-1}\frac{\alpha^{(k-\ell-1)(H)}2\alpha^H\delta_\ell + \varepsilon_{LA}}{1-\alpha}\\ &\le \frac{\alpha^{kH}}{1-\alpha} + \sum_{\ell=0}^{k-1}\frac{\alpha^{(k-\ell-1)(H)}2\alpha^H\delta_\ell}{1-\alpha} + \frac{\varepsilon_{LA}}{(1-\alpha)(1-\alpha^{H-1})}.\end{aligned} \tag{B.3}$$

Note that for the bound in (B.3) to be useful, we need for the δ_k sequence to exhibit some properties that ensure that the second term does not go to infinity as $k \to \infty$.

The bound in (B.3) can be further simplified if

$$\delta_k \le \beta^k \delta_0 + \mu \quad \text{for } 0 < \beta < 1, 0 < \mu. \tag{B.4}$$

Starting from (B.2), where $\delta_k = \beta^k \delta_0 + \mu$, we get the following:

$$\begin{aligned}
\|J^{\mu_k} - J^*\|_\infty &\le \frac{\alpha^{k(H)}}{1-\alpha} + \frac{2\alpha^H}{1-\alpha}\sum_{\ell=0}^{k-1}\alpha^{(k-\ell-1)(H)}[\beta^\ell\delta_0 + \mu] \\
&\quad + \frac{\varepsilon_{LA}}{(1-\alpha)(1-\alpha^H)} \\
&\le \frac{\alpha^{k(H)}}{1-\alpha} + \frac{2\alpha^H}{1-\alpha}\delta_0\sum_{\ell=0}^{k-1}\alpha^{(k-\ell-1)(H-1)}\beta^\ell \\
&\quad + \frac{2\alpha^H\mu + \varepsilon_{LA}}{(1-\alpha)(1-\alpha^H)} \\
&\le \frac{\alpha^{k(H)}}{1-\alpha} + \frac{2\alpha^H}{1-\alpha}\delta_0\sum_{\ell=0}^{k-1}\max(\alpha^{H-1},\beta)^{k-1} \\
&\quad + \frac{2\alpha^H\mu + \varepsilon_{LA}}{(1-\alpha)(1-\alpha^H)} \\
&= \frac{\alpha^{k(H)}}{1-\alpha} + \frac{2\alpha^H}{1-\alpha}k\max(\alpha^{H-1},\beta)^{k-1}\delta_0 \\
&\quad + \frac{2\alpha^H\mu + \varepsilon_{LA}}{(1-\alpha)(1-\alpha^H)}.
\end{aligned}$$

Taking limits on both sides, noting that $0 < \beta < 1$, we have

$$\limsup_{k\to\infty} \|J^{\mu_k} - J^*\|_\infty \le \frac{2\alpha^H\mu + \varepsilon_{LA}}{(1-\alpha)(1-\alpha^H)}.$$

Appendix C. Obtaining β and μ for Algorithm 1

Using Assumption 1, J_{k+1} can be written as

$$J_{k+1} = \Phi\theta_{k+1} = \underbrace{\Phi(\Phi_{\mathcal{D}_k}^\top\Phi_{\mathcal{D}_k})^{-1}\Phi_{\mathcal{D}_k}^\top\mathcal{P}_k}_{=:\mathcal{M}_{k+1}}\hat{J}^{\mu_{k+1}},$$

where $\Phi_{\mathcal{D}_k}$ is a matrix whose rows are the feature vectors of the states in $\mathcal{D}_k$ and $\mathcal{P}_k$ is a matrix of zeros and ones such that $\mathcal{P}_k\hat{J}^{\mu_{k+1}}$ is a vector whose elements are a subset of the elements of $\hat{J}^{\mu_{k+1}}$ corresponding to $\mathcal{D}_k$. Note that $\hat{J}^{\mu_{k+1}}(i)$ for $i \notin \mathcal{D}_k$ does not affect the algorithm, so we can define $\hat{J}^{\mu_{k+1}}(i) = T^m_{\mu_{k+1}}T^{H-1}J_k(i)$ for $i \notin \mathcal{D}_k$.

Written concisely, our algorithm is as follows:

$$J_{k+1} = \mathcal{M}_{k+1}(T^m_{\mu_{k+1}}T^{H-1}J_k + w_k), \tag{C.1}$$

where μ_{k+1} is defined in Step 2 of Algorithm 1. Because $w_k(i)$ for $i \notin \mathcal{D}_k$ does not affect the algorithm, we define $w_k(i) = 0$ for $i \notin \mathcal{D}_k$.

Using contraction properties of T_{μ_k} and T along with the triangle inequality, we obtain δ_k as follows:

$$\begin{aligned}
\|J_k - J^{\mu_k}\|_\infty &= \|\mathcal{M}_k(T^m_{\mu_k}T^{H-1}J_{k-1} + w_k) - J^{\mu_k}\|_\infty \\
&\le \|\mathcal{M}_kT^m_{\mu_k}T^{H-1}J_{k-1} - J^{\mu_k}\|_\infty + \|\mathcal{M}_kw_k\|_\infty \\
&\le \|\mathcal{M}_kT^m_{\mu_k}T^{H-1}J_{k-1} - J^{\mu_k}\|_\infty + \|\mathcal{M}_k\|_\infty\|w_k\|_\infty \\
&\le \|\mathcal{M}_kT^m_{\mu_k}T^{H-1}J_{k-1} - J^{\mu_k}\|_\infty + \delta_{FV}\varepsilon_{PE} \\
&= \|\mathcal{M}_kT^m_{\mu_k}T^{H-1}J_{k-1} - \mathcal{M}_kJ^{\mu_k} + \mathcal{M}_kJ^{\mu_k} - J^{\mu_k}\|_\infty \\
&\quad + \delta_{FV}\varepsilon_{PE} \\
&\le \|\mathcal{M}_kT^m_{\mu_k}T^{H-1}J_{k-1} - \mathcal{M}_kJ^{\mu_k}\|_\infty + \|\mathcal{M}_kJ^{\mu_k} - J^{\mu_k}\|_\infty \\
&\quad + \delta_{FV}\varepsilon_{PE} \\
&\le \|\mathcal{M}_k\|_\infty\|T^m_{\mu_k}T^{H-1}J_{k-1} - J^{\mu_k}\|_\infty + \|\mathcal{M}_kJ^{\mu_k} - J^{\mu_k}\|_\infty \\
&\quad + \delta_{FV}\varepsilon_{PE} \\
&\le \alpha^m\|\mathcal{M}_k\|_\infty\|T^{H-1}J_{k-1} - J^{\mu_k}\|_\infty + \sup_{k,\mu_k}\|\mathcal{M}_kJ^{\mu_k} - J^{\mu_k}\|_\infty \\
&\quad + \delta_{FV}\varepsilon_{PE} \\
&\le \alpha^m\|\mathcal{M}_k\|_\infty\|T^{H-1}J_{k-1} - J^* + J^* - J^{\mu_k}\|_\infty + \delta_{app} \\
&\quad + \delta_{FV}\varepsilon_{PE} \\
&\le \alpha^m\|\mathcal{M}_k\|_\infty\|T^{H-1}J_{k-1} - J^*\|_\infty + \alpha^m\|\mathcal{M}_k\|_\infty\|J^* - J^{\mu_k}\|_\infty \\
&\quad + \delta_{app} + \delta_{FV}\varepsilon_{PE} \\
&\le \alpha^{m+H-1}\|\mathcal{M}_k\|_\infty\|J_{k-1} - J^*\|_\infty + \frac{\alpha^m}{1-\alpha}\|\mathcal{M}_k\|_\infty + \delta_{app} \\
&\quad + \delta_{FV}\varepsilon_{PE} \\
&\le \alpha^{m+H-1}\|\mathcal{M}_k\|_\infty\|J_{k-1} - J^{\mu_{k-1}} + J^{\mu_{k-1}} - J^*\|_\infty \\
&\quad + \frac{\alpha^m}{1-\alpha}\|\mathcal{M}_k\|_\infty + \delta_{app} + \delta_{FV}\varepsilon_{PE} \\
&\le \alpha^{m+H-1}\|\mathcal{M}_k\|_\infty\|J_{k-1} - J^{\mu_{k-1}}\|_\infty \\
&\quad + \frac{\alpha^m + \alpha^{m+H-1}}{1-\alpha}\|\mathcal{M}_k\|_\infty + \delta_{app} + \delta_{FV}\varepsilon_{PE} \\
&\le \alpha^{m+H-1}\delta_{FV}\|J_{k-1} - J^{\mu_{k-1}}\|_\infty + \frac{\alpha^m + \alpha^{m+H-1}}{1-\alpha}\delta_{FV} \\
&\quad + \delta_{app} + \delta_{FV}\varepsilon_{PE}.
\end{aligned}$$

Now, we have

$$\underbrace{\|J_k - J^{\mu_k}\|_\infty}_{\delta_k} \le \underbrace{\alpha^{m+H-1}\delta_{FV}}_{=:\beta}\underbrace{\|J_{k-1} - J^{\mu_{k-1}}\|_\infty}_{\delta_{k-1}} + \underbrace{\frac{\alpha^m + \alpha^{m+H-1}}{1-\alpha}\delta_{FV} + \delta_{app} + \delta_{FV}\varepsilon_{PE}}_{=:\tau}.$$

Iterating,

$$\delta_k \le \beta^k \delta_0 + \sum_{i=0}^{k-1} \beta^i \tau \le \beta^k \delta_0 + \underbrace{\frac{\tau}{1-\beta}}_{=:\mu}. \tag{C.2}$$

Appendix D. A Modified Least-Squares Algorithm

Suppose Step 3 of Algorithm 2 is changed to $\hat{J}^{\mu_{k+1}}(i) = T^m_{\mu_{k+1}}(J_k)(i) + w_{k+1}(i)$ for $i \in \mathcal{D}_k$. Then, it is still possible to get bounds on the performance of the algorithm when m is sufficiently large. With this modification to the algorithm, we have the following.

Proposition D.1. *Suppose that m satisfies $m > \log(\delta_{FV})/\log(1/\alpha)$, where*

$$\delta_{FV} := \sup_k \|\mathcal{M}_k\|_\infty = \sup_k \|\Phi(\Phi^\top_{\mathcal{D}_k}\Phi_{\mathcal{D}_k})^{-1}\Phi^\top_{\mathcal{D}_k}\mathcal{P}_k\|_\infty.$$

Then, under Assumption 1, the following holds:

$$\|J^{\mu_k} - J^*\|_\infty \le \underbrace{\frac{\alpha^{k(H)}}{1-\alpha} + \frac{2\alpha^H\|J^{\mu_0} - J_0\|_\infty}{1-\alpha} k \max(\alpha^H, \beta')^{k-1}}_{\text{finite-time component}} + \underbrace{\frac{2\alpha^H \frac{\tau'}{1-\beta'} + \varepsilon_{LA}}{(1-\alpha^H)(1-\alpha)}}_{\text{asymptotic component}},$$

where

$$\tau' := \alpha^m \delta_{FV},$$

$$\beta' := \frac{\alpha^m \delta_{FV}}{1-\alpha} + \delta_{app} + \delta_{FV}\varepsilon_{PE},$$

and

$$\delta_{app} := \sup_{k,\mu_k} \|\mathcal{M}_k J^{\mu_k} - J^{\mu_k}\|_\infty.$$

Proof of Proposition D.1. The proof of Theorem 2 is similar to the proof of Theorem 1 and relies on contraction properties and the triangle inequality. We thus give the following iteration, which can be substituted in our proof of Theorem 1:

$$\begin{aligned}
&\|J_k - J^{\mu_k}\|_\infty \\
&= \|\mathcal{M}_k(T^m_{\mu_k}J_{k-1} + w_k) - J^{\mu_k}\|_\infty \\
&= \|\mathcal{M}_k(T^m_{\mu_k}J_{k-1} + w_k) - J^{\mu_k}\|_\infty \\
&\le \|\mathcal{M}_k T^m_{\mu_k}J_{k-1} - J^{\mu_k}\|_\infty + \|\mathcal{M}_k w_k\|_\infty \\
&\le \|\mathcal{M}_k T^m_{\mu_k}J_{k-1} - J^{\mu_k}\|_\infty + \|\mathcal{M}_k\|_\infty\|w_k\|_\infty \\
&\le \|\mathcal{M}_k T^m_{\mu_k}J_{k-1} - J^{\mu_k}\|_\infty + \delta_{FV}\varepsilon_{PE} \\
&= \|\mathcal{M}_k T^m_{\mu_k}J_{k-1} - \mathcal{M}_k J^{\mu_k} + \mathcal{M}_k J^{\mu_k} - J^{\mu_k}\|_\infty + \delta_{FV}\varepsilon_{PE} \\
&\le \|\mathcal{M}_k T^m_{\mu_k}J_{k-1} - \mathcal{M}_k J^{\mu_k}\|_\infty + \|\mathcal{M}_k J^{\mu_k} - J^{\mu_k}\|_\infty + \delta_{FV}\varepsilon_{PE} \\
&\le \sup_k \|\mathcal{M}_k\|_\infty \|T^m_{\mu_k}J_{k-1} - J^{\mu_k}\|_\infty + \sup_{k,\mu_k}\|\mathcal{M}_k J^{\mu_k} - J^{\mu_k}\|_\infty + \delta_{FV}\varepsilon_{PE} \\
&\le \alpha^m\delta_{FV}\|J_{k-1} - J^{\mu_k}\|_\infty + \delta_{app} + \delta_{FV}\varepsilon_{PE} \\
&= \alpha^m\delta_{FV}\|J_{k-1} - J^{\mu_{k-1}} + J^{\mu_{k-1}} - J^{\mu_k}\|_\infty + \delta_{app} + \delta_{FV}\varepsilon_{PE} \\
&\le \alpha^m\delta_{FV}\|J_{k-1} - J^{\mu_{k-1}}\|_\infty + \alpha^m\delta_{FV}\|J^{\mu_{k-1}} - J^{\mu_k}\|_\infty + \delta_{app} + \delta_{FV}\varepsilon_{PE} \\
&\le \alpha^m\delta_{FV}\|J_{k-1} - J^{\mu_{k-1}}\|_\infty + \frac{\alpha^m\delta_{FV}}{1-\alpha} + \delta_{app} + \delta_{FV}\varepsilon_{PE}.
\end{aligned}$$

Substituting

$$\beta' := \alpha^m\delta_{FV}$$

and

$$\tau' := \frac{\alpha^m\delta_{FV}}{1-\alpha} + \delta_{app} + \delta_{FV}\varepsilon_{PE},$$

in place of β and τ, respectively, in the proof of Theorem 1, we obtain Proposition D.1. □

Appendix E. Bounds on J_k in Algorithm 2

In the following proposition, we present a bound on the difference between J_k and J^*.

Proposition E.1. When $\alpha^{m+H-1}\delta_{FV} < 1$,

$$\limsup_{k\to\infty} \|J_k - J^*\|_\infty \le \frac{(1+\delta_{FV}\alpha^m)\left[\frac{2\alpha^H\frac{\tau}{1-\beta}+\varepsilon_{LA}}{(1-\alpha^H)(1-\alpha)}\right] + \delta_{app} + \delta_{FV}\varepsilon_{LA}}{1-\delta_{FV}\alpha^{m+H-1}},$$

where β and τ are defined in Theorem 1.

The proof is as follows.

Proof of Proposition E.1.

$$\begin{aligned}
&\|J_{k+1} - J^*\|_\infty \\
&= \|J_{k+1} - J^{\mu_{k+1}} + J^{\mu_{k+1}} - J^*\|_\infty \\
&\le \|J_{k+1} - J^{\mu_{k+1}}\|_\infty + \|J^{\mu_{k+1}} - J^*\|_\infty \\
&\le \|\mathcal{M}_{k+1}T^m_{\mu_{k+1}}T^{H-1}J_k - J^{\mu_{k+1}}\|_\infty + \delta_{FV}\varepsilon_{LA} \\
&\quad + \|J^{\mu_{k+1}} - J^*\|_\infty + \delta_{FV}\varepsilon_{LA} \\
&= \|\mathcal{M}_{k+1}T^m_{\mu_{k+1}}T^{H-1}J_k - \mathcal{M}_{k+1}J^{\mu_{k+1}} + \mathcal{M}_{k+1}J^{\mu_{k+1}} - J^{\mu_{k+1}}\|_\infty \\
&\quad + \|J^{\mu_{k+1}} - J^*\|_\infty + \delta_{FV}\varepsilon_{LA} \\
&\le \|\mathcal{M}_{k+1}T^m_{\mu_{k+1}}T^{H-1}J_k - \mathcal{M}_{k+1}J^{\mu_{k+1}}\|_\infty + \|\mathcal{M}_{k+1}J^{\mu_{k+1}} - J^{\mu_{k+1}}\|_\infty \\
&\quad + \|J^{\mu_{k+1}} - J^*\|_\infty + \delta_{FV}\varepsilon_{LA} \\
&\le \|\mathcal{M}_{k+1}\|_\infty\|T^m_{\mu_{k+1}}T^{H-1}J_k - J^{\mu_{k+1}}\|_\infty + \|\mathcal{M}_{k+1}J^{\mu_{k+1}} - J^{\mu_{k+1}}\|_\infty \\
&\quad + \|J^{\mu_{k+1}} - J^*\|_\infty + \delta_{FV}\varepsilon_{LA} \\
&\le \delta_{FV}\alpha^m\|T^{H-1}J_k - J^{\mu_{k+1}}\|_\infty + \delta_{app} \\
&\quad + \|J^{\mu_{k+1}} - J^*\|_\infty + \delta_{FV}\varepsilon_{LA} \\
&= \delta_{FV}\alpha^m\|T^{H-1}J_k - J^* + J^* - J^{\mu_{k+1}}\|_\infty + \delta_{app} + \|J^{\mu_{k+1}} - J^*\|_\infty \\
&\quad + \delta_{FV}\varepsilon_{LA} \\
&\le \delta_{FV}\alpha^m\|T^{H-1}J_k - J^*\|_\infty + \delta_{FV}\alpha^m\|J^* - J^{\mu_{k+1}}\|_\infty + \delta_{app} \\
&\quad + \|J^{\mu_{k+1}} - J^*\|_\infty + \delta_{FV}\varepsilon_{LA} \\
&\le \delta_{FV}\alpha^{m+H-1}\|J_k - J^*\|_\infty + \delta_{FV}\alpha^m\|J^* - J^{\mu_{k+1}}\|_\infty + \delta_{app} \\
&\quad + \|J^{\mu_{k+1}} - J^*\|_\infty + \delta_{FV}\varepsilon_{LA} \\
&= \delta_{FV}\alpha^{m+H-1}\|J_k - J^*\|_\infty + (1+\delta_{FV}\alpha^m)\|J^* - J^{\mu_{k+1}}\|_\infty + \delta_{app} \\
&\quad + \delta_{FV}\varepsilon_{LA}.
\end{aligned}$$

From Theorem 1, we have that

$$\limsup_{k\to\infty} \|J^{\mu_k} - J^*\|_\infty \le \frac{2\alpha^{H\frac{\tau}{1-\beta}} + \varepsilon_{LA}}{(1-\alpha^H)(1-\alpha)}.$$

Thus, for every $\varepsilon' > 0$, there exists a $k(\varepsilon')$ such that for all $k > k(\varepsilon')$,

$$\|J^{\mu_k} - J^*\|_\infty \le \frac{2\alpha^{H\frac{\tau}{1-\beta}} + \varepsilon_{LA}}{(1-\alpha^H)(1-\alpha)} + \varepsilon'.$$

Thus, for all $k > k(\varepsilon')$, we have

$$\begin{aligned}\|J_{k+1} - J^*\|_\infty &\le \delta_{FV}\alpha^{m+H-1}\|J_k - J^*\|_\infty \\ &\quad + (1+\delta_{FV}\alpha^m)\left[\frac{2\alpha^{H\frac{\tau}{1-\beta}} + \varepsilon_{LA}}{(1-\alpha^H)(1-\alpha)} + \varepsilon'\right] \\ &\quad + \delta_{app} + \delta_{FV}\varepsilon_{LA}.\end{aligned}$$

Iterating over k gives us

$$\begin{aligned}&\limsup_{k\to\infty} \|J_k - J^*\|_\infty \\ &\le \frac{(1+\delta_{FV}\alpha^m)\left[\frac{2\alpha^{H\frac{\tau}{1-\beta}} + \varepsilon_{LA}}{(1-\alpha^H)(1-\alpha)} + \varepsilon'\right] + \delta_{app} + \delta_{FV}\varepsilon_{LA}}{1 - \delta_{FV}\alpha^{m+H-1}}.\end{aligned}$$

Because this holds for all ε',

$$\begin{aligned}&\limsup_{k\to\infty} \|J_k - J^*\|_\infty \\ &\le \frac{(1+\delta_{FV}\alpha^m)\left[\frac{2\alpha^{H\frac{\tau}{1-\beta}} + \varepsilon_{LA}}{(1-\alpha^H)(1-\alpha)}\right] + \delta_{app} + \delta_{FV}\varepsilon_{LA}}{1 - \delta_{FV}\alpha^{m+H-1}}. \quad \square\end{aligned}$$

Appendix F. Obtaining β and μ for Algorithm 4

In order to derive β and μ for Algorithm 4, we define $\tilde{\theta}^{\mu_k}$ for any policy μ_k:

$$\tilde{\theta}^{\mu_k} := \arg\min_\theta \frac{1}{2}\|\Phi_{\mathcal{D}_k}\theta - \mathcal{P}_k(T^m_{\mu_k}T^{H-1}J_{k-1} + w_k)\|_2^2.$$

Note that

$$\Phi\tilde{\theta}^{\mu_k} = \mathcal{M}_k(T^m_{\mu_k}T^{H-1}J_{k-1} + w_k), \tag{F.1}$$

where $\mathcal{M}_k$ is defined in (6). Thus, $\tilde{\theta}^{\mu_k}$ represents the function approximation of the estimate of J^{μ_k} obtained from the m-step return.

First, because θ_k is obtained by taking η steps of gradient descent toward $\tilde{\theta}^{\mu_k}$ beginning from θ_{k-1}, we show that the following holds:

$$\|\theta_k - \tilde{\theta}^{\mu_k}\|_2 \le \alpha^\eta_{GD,\gamma}\|\theta_{k-1} - \tilde{\theta}^{\mu_k}\|_2,$$

where $\alpha_{GD,\gamma} := \sup_k \max_i |1 - \gamma\lambda_i(\Phi^\top_{\mathcal{D}_k}\Phi_{\mathcal{D}_k})|$, in which λ_i denotes the ith-largest eigenvalue of a matrix.

We note that because

$$\begin{aligned}0 < \lambda_i(\Phi^\top_{\mathcal{D}_k}\Phi_{\mathcal{D}_k}) &\le \|\Phi^\top_{\mathcal{D}_k}\Phi_{\mathcal{D}_k}\|_2^2 \le d\|\Phi^\top_{\mathcal{D}_k}\Phi_{\mathcal{D}_k}\|_\infty^2 \\ &\le d \sup_k \|\Phi^\top_{\mathcal{D}_k}\Phi_{\mathcal{D}_k}\|_\infty^2,\end{aligned}$$

$\alpha_{GD,\gamma} < 1$ when $\gamma < \frac{1}{d\sup_k\|\Phi^\top_{\mathcal{D}_k}\Phi_{\mathcal{D}_k}\|_\infty^2}$, which follows from Assumption 2.

Recall that the iterates in Equation (11) can be written as follows:

$$\begin{aligned}\theta_{k,\ell} &= \theta_{k,\ell-1} - \gamma\nabla_\theta c(\theta; \hat{J}^{\mu_k})|_{\theta_{k,\ell-1}} \\ &= \theta_{k,\ell-1} - \gamma(\Phi^\top_{\mathcal{D}_{k-1}}\Phi_{\mathcal{D}_{k-1}}\theta_{k,\ell-1} \\ &\quad - \Phi^\top_{\mathcal{D}_{k-1}}\mathcal{P}_{k-1}(T^m_{\mu_k}T^{H-1}J_k + w_{k-1})).\end{aligned}$$

Because

$$\begin{aligned}0 = \nabla_\theta c(\theta; \hat{J}^{\mu_k})|_{\tilde{\theta}^{\mu_k}} &= \Phi^\top_{\mathcal{D}_{k-1}}\Phi_{\mathcal{D}_{k-1}}\tilde{\theta}^{\mu_k} \\ &\quad - \Phi^\top_{\mathcal{D}_{k-1}}\mathcal{P}_{k-1}(T^m_{\mu_k}T^{H-1}J_k + w_{k-1}),\end{aligned}$$

we have the following:

$$\begin{aligned}\theta_{k,\ell} &= \theta_{k,\ell-1} - \gamma(\Phi^\top_{\mathcal{D}_{k-1}}\Phi_{\mathcal{D}_{k-1}}\theta_{k,\ell-1} - \Phi^\top_{\mathcal{D}_{k-1}}\Phi_{\mathcal{D}_{k-1}}\tilde{\theta}^{\mu_k} \\ &\quad - \Phi^\top_{\mathcal{D}_{k-1}}\mathcal{P}_{k-1}(T^m_{\mu_k}T^{H-1}J_k + w_{k-1}) \\ &\quad + \Phi^\top_{\mathcal{D}_{k-1}}\mathcal{P}_{k-1}(T^m_{\mu_k}T^{H-1}J_k + w_{k-1})) \\ &= \theta_{k,\ell-1} - \gamma\Phi^\top_{\mathcal{D}_{k-1}}\Phi_{\mathcal{D}_{k-1}}(\theta_{k,\ell-1} - \tilde{\theta}^{\mu_k}).\end{aligned}$$

Subtracting $\tilde{\theta}^{\mu_k}$ from both sides gives

$$\begin{aligned}\theta_{k,\ell} - \tilde{\theta}^{\mu_k} &= \theta_{k,\ell-1} - \tilde{\theta}^{\mu_k} - \gamma\Phi^\top_{\mathcal{D}_{k-1}}\Phi_{\mathcal{D}_{k-1}}(\theta_{k,\ell-1} - \tilde{\theta}^{\mu_k}) \\ &= (I - \gamma\Phi^\top_{\mathcal{D}_{k-1}}\Phi_{\mathcal{D}_{k-1}})(\theta_{k,\ell-1} - \tilde{\theta}^{\mu_k}).\end{aligned}$$

Thus,

$$\begin{aligned}\|\theta_{k,\ell} - \tilde{\theta}^{\mu_k}\|_2 &= \|(I - \gamma\Phi^\top_{\mathcal{D}_{k-1}}\Phi_{\mathcal{D}_{k-1}})(\theta_{k,\ell-1} - \tilde{\theta}^{\mu_k})\|_2 \\ &\le \|I - \gamma\Phi^\top_{\mathcal{D}_{k-1}}\Phi_{\mathcal{D}_{k-1}}\|_2\|\theta_{k,\ell-1} - \tilde{\theta}^{\mu_k}\|_2 \\ &\le \max_i |\lambda_i(I - \gamma\Phi^\top_{\mathcal{D}_{k-1}}\Phi_{\mathcal{D}_{k-1}})|\|\theta_{k,\ell-1} - \tilde{\theta}^{\mu_k}\|_2 \\ &\le \max_i |1 - \gamma\lambda_i(\Phi^\top_{\mathcal{D}_{k-1}}\Phi_{\mathcal{D}_{k-1}})|\|\theta_{k,\ell-1} - \tilde{\theta}^{\mu_k}\|_2 \\ &\le \underbrace{\sup_k \max_i |1 - \gamma\lambda_i(\Phi^\top_{\mathcal{D}_k}\Phi_{\mathcal{D}_k})|}_{=:\alpha_{GD,\gamma}}\|\theta_{k,\ell-1} - \tilde{\theta}^{\mu_k}\|_2,\end{aligned}$$

where λ_i denotes the ith-largest eigenvalue of a matrix.

Iterating over k, the following holds:

$$\begin{aligned}\|\theta_k - \tilde{\theta}^{\mu_k}\|_2 &= \|\theta_{k,\eta} - \tilde{\theta}^{\mu_k}\|_2 \\ &\le \alpha^\eta_{GD,\gamma}\|\theta_{k,0} - \tilde{\theta}^{\mu_k}\|_2 \\ &= \alpha^\eta_{GD,\gamma}\|\theta_{k-1} - \tilde{\theta}^{\mu_k}\|_2.\end{aligned} \tag{F.2}$$

Using (F.2) as well as equivalence and submultiplicative properties of matrix norms, we have the following:

$$\begin{aligned}\frac{1}{\|\Phi\|_\infty}\|\Phi\theta_k - \Phi\tilde{\theta}^{\mu_k}\|_\infty &\le \|\theta_k - \tilde{\theta}^{\mu_k}\|_\infty \\ &\le \|\theta_k - \tilde{\theta}^{\mu_k}\|_2 \\ &\le \alpha^\eta_{GD,\gamma}\|\theta_{k-1} - \tilde{\theta}^{\mu_k}\|_2 \\ &\le \frac{1}{\sigma_{\min,\Phi}}\alpha^\eta_{GD,\gamma}\|\Phi\theta_{k-1} - \Phi\tilde{\theta}^{\mu_k}\|_2 \\ &\le \frac{\sqrt{|S|}}{\sigma_{\min,\Phi}}\alpha^\eta_{GD,\gamma}\|\Phi\theta_{k-1} - \Phi\tilde{\theta}^{\mu_k}\|_\infty \\ \Rightarrow \|J_k - \Phi\tilde{\theta}^{\mu_k}\|_\infty &\le \frac{\sqrt{|S|}\|\Phi\|_\infty}{\sigma_{\min,\Phi}}\alpha^\eta_{GD,\gamma}\|J_{k-1} - \Phi\tilde{\theta}^{\mu_k}\|_\infty,\end{aligned}$$

where $\sigma_{\min,\Phi}$ is the smallest singular value in the singular value decomposition of Φ and the last line follows from the fact that $J_k := \Phi\theta_k$.

This implies the following:

$$\begin{aligned}\|J^{\mu_k} - J_k\|_\infty &\le \|\Phi\tilde{\theta}^{\mu_k} - J^{\mu_k}\|_\infty + \frac{\sqrt{|S|}\|\Phi\|_\infty}{\sigma_{\min,\Phi}}\alpha^{\eta}_{GD,\gamma}\|J_{k-1} - \Phi\tilde{\theta}^{\mu_k}\|_\infty \\ &= \|\mathcal{M}_k(T^m_{\mu_k}T^{H-1}J_{k-1} + w_k) - J^{\mu_k}\|_\infty \\ &\quad + \frac{\sqrt{|S|}\|\Phi\|_\infty}{\sigma_{\min,\Phi}}\alpha^{\eta}_{GD,\gamma}\|J_{k-1} - \Phi\tilde{\theta}^{\mu_k}\|_\infty, \end{aligned} \tag{F.3}$$

where the equality follows from (F.1).

Now, we bound $\|J_{k-1} - \Phi\tilde{\theta}^{\mu_k}\|_\infty$ as follows:

$$\begin{aligned}\|J_{k-1} - \Phi\tilde{\theta}^{\mu_k}\|_\infty &\le \|J_{k-1} - J^{\mu_{k-1}}\|_\infty + \|J^{\mu_{k-1}} - J^{\mu_k}\|_\infty \\ &\quad + \|J^{\mu_k} - \Phi\tilde{\theta}^{\mu_k}\|_\infty \\ &\le \|J_{k-1} - J^{\mu_{k-1}}\|_\infty + \frac{1}{1-\alpha} + \|J^{\mu_k} - \Phi\tilde{\theta}^{\mu_k}\|_\infty \\ &\le \|J_{k-1} - J^{\mu_{k-1}}\|_\infty + \frac{1}{1-\alpha} \\ &\quad + \|J^{\mu_k} - \mathcal{M}_k(T^m_{\mu_k}T^{H-1}J_{k-1} + w_k)\|_\infty, \end{aligned} \tag{F.4}$$

where the last line follows from (F.1). We use our upper bound on $\|\mathcal{M}_k(T^m_{\mu_k}T^{H-1}J_{k-1} + w_k) - J^{\mu_k}\|_\infty$ introduced in Appendix C to put together with (F.3) and (F.4), and we get the following:

$$\underbrace{\|J^{\mu_k} - J_k\|_\infty}_{\delta_k} \le \beta\underbrace{\|J_{k-1} - J^{\mu_{k-1}}\|_\infty}_{\delta_{k-1}} + \tau,$$

$$\beta := \alpha^{m+H-1}\delta_{FV} + \frac{\sqrt{|S|}\|\Phi\|_\infty}{\sigma_{\min,\Phi}}\alpha^{\eta}_{GD,\gamma}(\alpha^{m+H-1}\delta_{FV} + 1),$$

and

$$\begin{aligned}\tau := &\left(1 + \frac{\sqrt{|S|}\|\Phi\|_\infty}{\sigma_{\min,\Phi}}\alpha^{\eta}_{GD,\gamma}\right)\left(\frac{\alpha^m + \alpha^{m+H-1}}{1-\alpha}\delta_{FV} + \delta_{app} + \delta_{FV}\varepsilon_{PE}\right) \\ &+ \frac{\sqrt{|S|}\|\Phi\|_\infty}{(1-\alpha)\sigma_{\min,\Phi}}\alpha^{\eta}_{GD,\gamma}.\end{aligned}$$

Thus, we get

$$\mu = \frac{\tau}{1-\beta}$$

when $0 < \beta < 1$, which follows from the assumptions in Proposition 1 and Assumption 2.

References

Arora S, Du S, Hu W, Li Z, Wang R (2019) Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. *36th Internat. Conf. Machine Learning, ICML 2019* (International Machine Learning Society), 477–502.

Baxter J, Tridgell A, Weaver L (1999) TDleaf(lambda): Combining temporal difference learning with game-tree search. *Clinical Orthopaedics Related Res.*

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

Bertsekas DP (2019) *Reinforcement Learning and Optimal Control* (Athena Scientific, Belmont, MA).

Bertsekas D (2021) Lessons from alphazero for optimal, model predictive, and adaptive control. Working paper, MIT, Cambridge, MA.

Bertsekas D, Tsitsiklis J (1996) *Neuro-Dynamic Programming* (Athena Scientific, Belmont, MA).

Browne C, Powley E, Whitehouse D, Lucas S, Cowling P, Rohlfshagen P, Tavener S, Perez Liebana D, Samothrakis S, Colton S (2012) A survey of Monte Carlo tree search methods. *IEEE Trans. Comput. Intelligence AI Games* 4(1):1–43.

Buşoniu L, Lazaric A, Ghavamzadeh M, Munos R, Babuška R, Schutter BD (2012) Least-squares methods for policy iteration. Wiering M, van Otterlo M, eds. *Reinforcement Learning. Adaptation, Learning, and Optimization*, vol. 12 (Springer, Berlin), 75–109.

Cao Y, Gu Q (2019) Generalization bounds of stochastic gradient descent for wide and deep neural networks. *Adv. Neural Inform. Processing Systems* 32:10836–10846.

Deng H, Yin S, Deng X, Li S (2020) Value-based algorithms optimization with discounted multiple-step learning method in deep reinforcement learning. *2020 IEEE 22nd Internat. Conf. High Performance Comput. Comm. IEEE 18th Internat. Conf. Smart City IEEE 6th Internat. Conf. Data Sci. Systems (HPCC/SmartCity/DSS)* (IEEE, Piscataway, NJ), 979–984.

Du SS, Zhai X, Poczos B, Singh A (2019) Gradient descent provably optimizes over-parameterized neural networks. *Internat. Conf. Learn. Representations* (OpenReview.net).

Efroni Y, Ghavamzadeh M, Mannor S (2020) Online planning with lookahead policies. *Adv. Neural Inform. Processing Systems* 33.

Efroni Y, Dalal G, Scherrer B, Mannor S (2018a) Beyond the one step greedy approach in reinforcement learning. Preprint, submitted July 30, https://arxiv.org/abs/1802.03654.

Efroni Y, Dalal G, Scherrer B, Mannor S (2018b) Multiple-step greedy policies in online and approximate reinforcement learning. *NIPS'18: Proc. 32nd Internat. Conf. Neural Inform. Processing Systems* (Curran Associates Inc., Red Hook, NY), 5244–5253.

Efroni Y, Dalal G, Scherrer B, Mannor S (2019) How to combine tree-search methods in reinforcement learning. *Thirty-Third AAAI Conf. Artificial Intelligence* (AAAI-19) (AAAI Press, Palo Alto, CA), 3494–3501.

Hong ZW, Pajarinen J, Peters J (2019) Model-based lookahead reinforcement learning. Preprint, submitted August 15, https://arxiv.org/pdf/1908.06012.

Jacot A, Gabriel F, Hongler C (2018) Neural tangent kernel: Convergence and generalization in neural networks. Preprint, submitted June 20, https://arxiv.org/abs/1806.07572.

Ji Z, Telgarsky M (2019) Polylogarithmic width suffices for gradient descent to achieve arbitrarily small test error with shallow ReLu networks. *Internat. Conf. Learn. Representations* (OpenReview.net).

Kocsis L, Szepesvári C (2006) Bandit based Monte-Carlo planning. Fürnkranz J, Scheffer T, Spiliopoulou M, eds. *Machine Learning: ECML 2006*, Lecture Notes in Computer Science, vol. 4212 (Springer, Berlin), 282–293.

Lagoudakis MG, Parr R (2001) Model-free least-squares policy iteration. *Adv. Neural Inform. Processing Systems*, vol. 14 (MIT Press, Cambridge, MA).

Lagoudakis MG, Parr R (2003) Least-squares policy iteration. *J. Machine Learn. Res.* 4:1107–1149.

Lanctot M, Winands MHM, Pepels T, Sturtevant NR (2014) Monte Carlo tree search with heuristic evaluations using implicit minimax backups. *2014 IEEE Conf. Comput. Intelligence Games* (IEEE, Piscataway, NJ), 1–8.

Mnih V, Badia AP, Mirza M, Graves A, Lillicrap TP, Harley T, Silver D, Kavukcuoglu K (2016) Asynchronous methods for deep reinforcement learning. Preprint, submitted June 16, https://arxiv.org/abs/1602.01783.

Moerland TM, Broekens J, Jonker CM (2020) A framework for reinforcement learning and planning. Working paper, AAAI Press, Palo Alto, CA.

Munos R (2014) From bandits to Monte-Carlo tree search: The optimistic principle applied to optimization and planning. *Foundations Trends Machine Learn.* 7(1):1–129.

Puterman M, Shin MC (1978) Modified policy iteration algorithms for discounted Markov decision problems. *Management Sci.* 24(11): 1127–1137.

Shah D, Xie Q, Xu Z (2020a) Non-asymptotic analysis of Monte Carlo tree search. *ACM SIGMETRICS Performance Evaluation Review*, vol. 48 (ACM, New York), 31–32.

Shah D, Somani V, Xie Q, Xu Z (2020b) On reinforcement learning for turn-based zero-sum Markov games. Preprint, submitted February 25, https://arxiv.org/abs/2002.10620.

Silver D, Hubert T, Schrittwieser J, Antonoglou I, Lai M, Guez A, Lanctot M, et al. (2017a) Mastering chess and shogi by self-play with a general reinforcement learning algorithm. Preprint, submitted December 5, https://arxiv.org/abs/1712.01815.

Silver D, Schrittwieser J, Simonyan K, Antonoglou I, Huang A, Guez A, Hubert T, et al. (2017b) Mastering the game of go without human knowledge. *Nature* 550(7676):354–359.

Springenberg JT, Heess N, Mankowitz D, Merel J, Byravan A, Abdolmaleki A, Kay J, et al. (2020) Local search for policy iteration in continuous control. Preprint, submitted October 12, https://arxiv.org/abs/2010.05545.

Świechowski M, Godlewski K, Sawicki B, Mańdziuk J (2023) Monte Carlo tree search: A review of recent modifications and applications. *Artificial Intelligence Rev.* 56(3):2497–2562.

Tomar M, Efroni Y, Ghavamzadeh M (2020) Multi-step greedy reinforcement learning algorithms. *Internat. Conf. Machine Learning* (PMLR, New York), 9504–9513.

Tsitsiklis JN, Roy BV (1994) Feature-based methods for large scale dynamic programming. *Machine Learn.* 22(1):59–94.

Veness J, Silver D, Blair A, Uther W (2009) Bootstrapping from game tree search. Bengio Y, Schuurmans D, Lafferty J, Williams C, Culotta A, eds. *Adv. Neural Inform. Processing Systems*, vol. 22 (Curran Associates, Inc., Red Hook, NY), 1937–1945.

Winnicki A, Srikant R (2022) Reinforcement learning with unbiased policy evaluation and linear function approximation. *2022 IEEE 61st Conf. Decision Control (CDC)* (IEEE, Piscataway, NJ), 801–806.

Winnicki A, Srikant R (2023) A new policy iteration algorithm for reinforcement learning in zero-sum Markov games. Preprint, submitted March 17, https://arxiv.org/abs/2303.09716.

Anna Winnicki is a PhD candidate in electrical and computer engineering at the University of Illinois Urbana-Champaign. Her research interests lie in stochastic control, reinforcement learning, and electricity markets. She is a finalist in the 2023 INFORMS Applied Probability Society Best Student Paper Competition.

Joseph Lubars is a technical staff member at Sandia National Laboratories, where his research is focused on modeling of complex systems.

Michael Livesay is a technical staff member at Sandia National Laboratories, where he does research in formal methods, automated control, and modeling in complex systems.

R. Srikant is with the University of Illinois Urbana-Champaign, where he is the Co-Director of the c3.ai Digital Transformation Institute, a Grainger Distinguished Chair in Engineering, and a professor in the Department of Electrical and Computer Engineering and the Coordinated Science Laboratory. His research interests include applied probability, machine learning, stochastic control, and communication networks. Dr. Srikant is the recipient of the 2015 INFOCOM Achievement Award, the 2019 IEEE Koji Kobayashi Computers and Communications Award, the 2021 ACM SIGMETRICS Achievement Award, the 2015 INFOCOM Best Paper Award, the 2017 Applied Probability Society Best Publication Award, and the 2017 WiOpt Best Paper Award.