Performance Bounds for Lambda Policy Iteration and Application to the Game of Tetris

Bruno Scherrer

Performance Bounds for λ Policy Iteration and Application to the Game of Tetris

Bruno Scherrer BRUNO.SCHERRER@INRIA.FR
Maia Project-Team, INRIA Lorraine
615 rue du Jardin Botanique
54600 Villers-les-Nancy
FRANCE

Abstract

We consider the discrete-time infinite-horizon optimal control problem formalized by Markov Decision Processes (Puterman, 1994; Bertsekas and Tsitsiklis, 1996). We revisit the work of Bertsekas and Ioffe (1996), that introduced λ Policy Iteration, a family of algorithms parameterized by λ that generalizes the standard algorithms Value Iteration and Policy Iteration, and has some deep connections with the Temporal Differences algorithm TD(λ) described by Sutton and Barto (1998). We deepen the original theory developped by the authors by providing convergence rate bounds which generalize standard bounds for Value Iteration described for instance by Puterman (1994). Then, the main contribution of this paper is to develop the theory of this algorithm when it is used in an approximate form and show that this is sound. Doing so, we extend and unify the separate analyses developped by Munos for Approximate Value Iteration (Munos, 2007) and Approximate Policy Iteration (Munos, 2003). Eventually, we revisit the use of this algorithm in the training of a Tetris playing controller as originally done by Bertsekas and Ioffe (1996). We provide an original performance bound that can be applied to such an undiscounted control problem. Our empirical results are different from those of Bertsekas and Ioffe (which were originally qualified as "paradoxical" and "intriguing"), and much more conform to what one would expect from a learning experiment. We discuss the possible reason for such a difference.

Keywords: Stochastic Optimal Control, Reinforcement Learning, Markov Decision Processes, Analysis of Algorithms, Performance Bounds.

1. Introduction

We consider the discrete-time infinite-horizon optimal control problem formalized by Markov Decision Processes (Puterman, 1994; Bertsekas and Tsitsiklis, 1996). We revisit the λ Policy Iteration algorithm introduced by Bertsekas and Ioffe (1996) (also published in the reference textbook of Bertsekas and Tsitsiklis (1996)[1]), that (as the authors then stated) *"is primarily motivated by the case of large and complex problems where the use of approximation is essential"*. It is a family of algorithms parameterized by λ that generalizes the standard Dynamic Programming algorithms Value Iteration (which corresponds to the case $\lambda = 0$) and Policy Iteration (case $\lambda = 1$), and has some deep connections with the Temporal Dif-

1. The reference (Bertsekas and Ioffe, 1996) being historically anterior to (Bertsekas and Tsitsiklis, 1996), we only refer to the former in the rest of the paper.

ferences algorithm TD(λ) that are well known to the Reinforcement Learning community (Sutton and Barto, 1998; Bertsekas and Tsitsiklis, 1996).

In their original paper, Bertsekas and Ioffe (1996) show the convergence of λ Policy Iteration when it is run without error and provide its *asymptotic* convergence rate. The authors also describe a case study involving an instance of Approximate λ Policy Iteration, but neither their paper nor (to the best of our knowledge) any subsequent work studies the theoretical soundness of doing so. In this paper, we extend the theory on this algorithm in several ways. We derive its *non-asymptotic* convergence rate when it is run without error. More importantly, we develop the theory of λ Policy Iteration for its main purpose, that is — recall the above quote — when it is run in an approximate form, and prove that such an approach is sound: we show that the loss of using the greedy policy with respect to the current value estimate can be made arbitrarily small by controlling the error made during the iterations.

The rest of the paper is organized as follows. In Section 2, we introduce the framework of Markov Decision Processes and decribe the two standard algorithms, Value Iteration and Policy Iteration, along with some of their state-of-the-art analysis in exact and approximate form. Section 3 introduces λ Policy Iteration in an original way that makes its connection with Value Iteration and Policy Iteration obvious, and discusses its close connection with Reinforcement Learning. We recall the main results obtained by Bertsekas and Ioffe (1996) (convergence and asymptotic rate of convergence of the exact algorithm). At this point of the paper, we naturally describe how one expects that the properties of Value Iteration ($\lambda = 0$) and Policy Iteration ($\lambda = 1$) described in Section 2 may translate for general λ. The precise statements of our results are the topic of the next two Sections: Section 5 contains our new results on Exact λ Policy Iteration and Section 6 those on Approximate λ Policy Iteration[2]. Last but not least, Section 7 revisits the empirical part of the work of Bertsekas and Ioffe (1996), where an approximate version of λ Policy Iteration is used for training a Tetris controller.

Notations The analysis we describe in this article relies on a few notations, such as several norms and seminorms, that we need to define precisely before we can go further. Let X be a finite space. Let u denote a real-valued function on X, which can be seen as a vector of dimension $|X|$. Let $\mathbf{e}$ denote the vector of which all components are 1. The vector μ denotes a distribution on X. We consider the **weighted L_p norm**:

$$\|u\|_{p,\mu} := \left(\sum_x \mu(x)|u(x)|^p\right)^{1/p} = (\mu^T|u|^p)^{1/p}$$

where $|u|^p$ denotes the componentwise absolute value and exponentiation of u. We write $\|.\|_p$ the **unweighted L_p norm** (with uniform distribution μ). The max norm $\|.\|_\infty$ is:

$$\|u\|_\infty := \max_x |u(x)| = \lim_{p\to\infty} \|u\|_p .$$

We write $\text{span}_\infty[.]$ the **span seminorm** (as for instance defined by Puterman (1994)):

$$\underset{\infty}{\text{span}}[u] := \max_x u(x) - \min_x u(x).$$

2. Section 6 is probably the place where the reader familiar with Approximate Dynamic Programming would quickly want to jump.

It can be seen that

$$\operatorname*{span}_{\infty}[u] = 2\min_{a} \|u - a\mathbf{e}\|_{\infty}.$$

We propose to generalize the span seminorm definition for any p as follows:

$$\operatorname*{span}_{p,\mu}[u] := 2\min_{a} \|u - a\mathbf{e}\|_{p,\mu}$$

It is clear that it is a seminorm (it is non-negative, it satisfies the triangle inequality and $\operatorname{span}_*[au] = |a| \operatorname{span}_*[u]$). It is not a norm because it is zero for all constant functions.

The error bounds we derive in this paper are expressed in terms of some span seminorm. The following relations

$$\left\{ \begin{array}{lcll} \operatorname{span}_p[u] & \leq & 2\|u\|_p & \leq & 2\|u\|_\infty \\ \operatorname{span}_{p,\mu}[u] & \leq & 2\|u\|_{p,\mu} & \leq & 2\|u\|_\infty \\ \operatorname{span}_\infty[u] & \leq & 2\|u\|_\infty & & \end{array} \right. \tag{1}$$

show how to deduce error bounds involving the (more standard) L_p and max norms. Since the span seminorm can be zero for non zero (constant) vectors, there is no relation that would enable us to derive error bounds in span seminorm from a L_p or a max norm. Bounding an error with the span seminorm is in this sense stronger and this constitutes our motivation for using it.

2. Framework and Standard Algorithms

In this section, we begin by providing a short description of the framework of Markov Decision Processes we consider throughout the paper. We go on by describing the two main algorithms, Value Iteration and Policy Iteration, for solving the related problem.

2.1 Markov Decision Processes

We consider a discrete-time dynamic system whose state transition depends on a control. We assume that there is a **state space** X of finite size N. When at state $i \in \{1, .., N\}$, the control is chosen from a finite **control space** A. The control $a \in A$ specifies the **transition probability** $p_{ij}(a)$ to the next state j. At the k^{th} iteration, the system is given a reward $\gamma^k r(i, a, j)$ where r is the instantaneous **reward function**, and $0 < \gamma < 1$ is a discount factor. The tuple $\langle X, A, p, r, \gamma \rangle$ is called a **Markov Decision Process (MDP)** (Puterman, 1994; Bertsekas and Tsitsiklis, 1996).

We are interested in stationary deterministic policies, that is functions $\pi : X \to A$ which map states into controls[3]. Writing i_k the state at time k, the **value of policy** π at state i is defined as the *total expected discounted return* while following a policy π from i, that is

$$v^\pi(i) := \lim_{N\to\infty} E_\pi \left[\sum_{k=0}^{N-1} \gamma^k r(i_k, \pi(i_k), i_{k+1}) \middle| i_0 = i \right] \tag{2}$$

3. Restricting our attention to stationary deterministic policies is not a limitation. Indeed, for the optimality criterion to be defined soon, it can be shown that there exists at least one stationary deterministic policy which is optimal (Puterman, 1994).

where E_π denotes the expectation conditional on the fact that the actions are selected with the policy π (that is, for all k, i_{k+1} is reached from i_k with probability $p_{i_k i_{k+1}}(\pi(i_k))$). The **optimal value** starting from state i is defined as

$$v_*(i) := \max_\pi v^\pi(i).$$

We write P^π the $N \times N$ stochastic matrix whose elements are $p_{ij}(\pi(i))$ and r^π the vector whose components are $\sum_j p_{ij}(\pi(i))r(i,\pi(i),j)$. The value functions v^π and v_* can be seen as vectors on X. It is well known that v^π solves the following Bellman equation:

$$v^\pi = r^\pi + \gamma P^\pi v^\pi.$$

The value function v^π is a fixed point of the linear operator $\mathcal{T}^\pi v := r^\pi + \gamma P^\pi v$. As P^π is a stochastic matrix, its eigenvalues cannot be greater than 1, and consequently $I - \gamma P^\pi$ is invertible. This implies that

$$v^\pi = (I - \gamma P^\pi)^{-1} r^\pi = \sum_{i=0}^{\infty} (\gamma P^\pi)^i r^\pi. \tag{3}$$

It is also well known that v_* satisfies the following Bellman equation:

$$v_* = \max_\pi (r^\pi + \gamma P^\pi v_*) = \max_\pi \mathcal{T}^\pi v_*$$

where the max operator is componentwise. In other words, v_* is a fixed point of the nonlinear operator $\mathcal{T}_* v := \max_\pi \mathcal{T}^\pi v$. For any value vector v, we call a **greedy policy with respect to the value** v a policy π that satisfies:

$$\pi \in \arg\max_{\pi'} \mathcal{T}^{\pi'} v$$

or equivalently $\mathcal{T}^\pi v = \mathcal{T}_* v$. We write, with some abuse of notation[4] greedy(v) any policy that is greedy with respect to v. The notions of optimal value function and greedy policies are fundamental to optimal control because of the following property: any policy π_* that is greedy with respect to the optimal value is an **optimal policy** and its value v^{π_*} is equal to v_*.

The operators $\mathcal{T}^\pi$ and $\mathcal{T}_*$ can be shown to be γ-contraction mappings with respect to the max norm. In what follows we only write what this means for the Bellman operator $\mathcal{T}_*$ but the same holds for $\mathcal{T}^\pi$. Being a γ-contraction mapping for the max norm means that for all pairs of vectors (v, w),

$$\|\mathcal{T}_* v - \mathcal{T}_* w\|_\infty \leq \gamma \|v - w\|_\infty .$$

This ensures that the fixed point v_* of $\mathcal{T}$ exists and is unique. Furthermore, for any initial vector v_0,

$$\lim_{k \to \infty} (\mathcal{T}_*)^k v_0 = v_*. \tag{4}$$

Given an MDP, standard algorithmic solutions for computing an optimal value/policy (which dates back to the 1950s, see for instance (Puterman, 1994) and the references therein) are Value Iteration and Policy Iteration. The rest of this section describes both of these algorithms with some of the relevant properties for the subject of this paper.

4. There might be several policies that are greedy with respect to some value v.

Algorithm 1 `Value Iteration`

Input: An MDP, an initial value v_0
Output: An (approximately) optimal policy
  $k \leftarrow 0$
  repeat
    $v_{k+1} \leftarrow \mathcal{T}_* v_k + \epsilon_{k+1}$ // Update the value
    $k \leftarrow k + 1$
  until some stopping criterion
  Return greedy(v_k)

2.2 Value Iteration

The Value Iteration algorithms for computing the value of a policy π and the value of the optimal policy π_* rely on Equation 4. Algorithm 1 provides a description of Value Iteration for computing an optimal policy (replace $\mathcal{T}_*$ by $\mathcal{T}^\pi$ in it and one gets Value Iteration for computing the value of some policy π). In this description, we have introduced a term ϵ_k which stands for several possible sources of error at each iteration: this error might be the computer round off, the fact that we use an approximate architecture for representing v, a stochastic approximation of P^{π_k}, etc... or a combination of these. In what follows, when we talk about the *Exact* version of an algorithm, this means that $\epsilon_k = 0$ for all k.

Properties of Exact Value Iteration The contraction property induces some interesting properties for Exact Value Iteration. We have already mentioned that contraction implies the asymptotic convergence (Equation 4). It can also be inferred that there is at least a linear rate of convergence: for all reference iteration k_0, and for all $k \geq k_0$,

$$\|v_* - v_k\|_\infty \leq \gamma^{k-k_0} \|v_* - v_{k_0}\|_\infty .$$

Even more interestingly, it is possible to derive a performance bound, that is a bound of the difference between the real value of a policy produced by the algorithm and the value of the optimal policy π_* (Puterman, 1994). Let π_k denote the policy that is greedy with respect to v_{k-1}. Then, for all reference iteration k_0, and for all $k \geq k_0$,

$$\|v_* - v^{\pi_k}\|_\infty \leq \frac{2\gamma^{k-k_0}}{1-\gamma} \|\mathcal{T}_* v_{k_0} - v_{k_0}\|_\infty = \frac{2\gamma^{k-k_0}}{1-\gamma} \|v_{k_0+1} - v_{k_0}\|_\infty .$$

This fact is of considerable importance computationally since it provides a stopping criterion: taking $k = k_0 + 1$, we see that if $\|v_{k_0+1} - v_{k_0}\|_\infty \leq \frac{1-\gamma}{2\gamma}\epsilon$, then $\|v_* - v^{\pi_{k_0+1}}\|_\infty \leq \epsilon$.

It is somewhat less known that the Bellman operators $\mathcal{T}_*$ and $\mathcal{T}^\pi$ are also contraction mapping with respect to the span_∞ seminorm (Puterman, 1994). This means that there exists a variant of the above equation involving the span seminorm instead of the max norm. For instance, such a fact provides the following stopping criterion:

Proposition 1 (Stopping Cond. for Exact Value Iteration (Puterman, 1994)) *If at some iteration k_0, the difference between two subsequent iterations satisfies*

$$\operatorname*{span}_\infty [v_{k_0+1} - v_{k_0}] \leq \frac{1-\gamma}{\gamma}\epsilon,$$

then the greedy policy π_{k_0+1} with respect to v_{k_0} is ϵ-optimal: $\|v_* - v^{\pi_{k_0+1}}\|_\infty \leq \epsilon$.

This latter stopping criterion is finer since, from the relation between the span seminorm and the norm (Equation 1) it implies the former.

Properties of Approximate Value Iteration (AVI) When considering large Markov Decision Processes, one cannot usually implement an exact version of Value Iteration. In such a case $\epsilon_k \neq 0$. In general, the algorithm does not converge anymore but it is possible to study its asymptotic behaviour. The most well-known result is due to Bertsekas and Tsitsiklis (1996, pp. 332-333): if the approximation erros are uniformly bounded, the difference between the asymptotic performance of policies π_{k+1} greedy with respect to v_k satisfies

$$\limsup_{k\to\infty} \|v_* - v^{\pi_k}\|_\infty \leq \frac{2\gamma}{(1-\gamma)^2} \sup_{k\geq 0} \|\epsilon_k\|_\infty . \tag{5}$$

Munos (2003, 2007) has recently argued that, since most supervised learning algorithms (such as least square regression) that are used in practice for approximating each iterate of Value Iteration control some L_p norm, it would be more interesting to have an analogue of the above result where the approximation error ϵ_k is expressed in terms of the L_p norm. Munos (2007) actually showed how to do this. The idea is to analyze the *componentwise* asymptotic behaviour of Approximate Value Iteration, from which it is possible to derive L_p bounds for any p. Write $P_k = P^{\pi_k}$ the stochastic matrix corresponding to the policy π_k which is greedy with respect to v_{k-1}, P_* the stochastic matrix corresponding to the (unknown) optimal policy π_*. Munos (2007) showed the following lemma:

Lemma 2 (Asymptotic Componentwise Performance of AVI (Munos, 2007)) *For all $k > j \geq 0$, the following matrices*

$$\begin{aligned} Q_{kj} &:= (1-\gamma)(I-\gamma P_k)^{-1} P_k P_{k-1} ... P_{j+1} \\ Q'_{kj} &:= (1-\gamma)(I-\gamma P_k)^{-1} (P_*)^{k-j} \end{aligned}$$

are stochastic and the asymptotic performance of the policies generated by Approximate Value Iteration satisfies

$$\limsup_{k\to\infty} v_* - v^{\pi_k} \leq \limsup_{k\to\infty} \frac{1}{1-\gamma} \sum_{j=0}^{k-1} \gamma^{k-j} \left[Q_{kj} - Q'_{kj}\right] \epsilon_j .$$

From the above componentwise bound, it is possible[5] to derive the following L_p bounds.

Proposition 3 (Asymptotic Performance of AVI (1/2)) *Choose any p and any distribution μ. Consider the notations of Lemma 2. Then*

$$\mu_{kj} := \frac{1}{2}\left(Q_{jk} + Q'_{jk}\right)^T \mu$$

are distributions and the asymptotic performance of policies generated by Value Iteration satisfies

$$\limsup_{k\to\infty} \|v_* - v^{\pi_k}\|_{p,\mu} \leq \frac{2\gamma}{(1-\gamma)^2} \sup_{k\geq j\geq 0} \|\epsilon_k\|_{p,\mu_{kj}}$$

5. This result is not explicitely stated by Munos (2007), but using a technique of another of his articles (Munos, 2003), it can be derived from Lemma 2. The current paper generalizes this result (in Proposition 24 page 24).

As the above bounds rely on the partially unknown matrices Q_{kj} and Q'_{kj}, Munos (2003, 2007) introduced some assumption in terms of **concentration coefficient** to remove this dependency. Assume there exists a distribution ν and a real number $C(\nu)$ such that

$$C(\nu) := \max_{i,j,a} \frac{p_{ij}(a)}{\nu(j)}. \tag{6}$$

For instance, if one chooses the uniform law ν, then there always exists such a $C(\nu) \in (1, N)$ where N is the size of the state space (see (Munos, 2003, 2007) for more discussion on this coefficient). This allows to derive the following performance bounds on the max norm of the loss.

Proposition 4 (Asymptotic Performance of AVI (2/2) (Munos, 2007))
Let $C(\nu)$ be the concentration coefficient defined in Equation 6. The asymptotic performance of the policies generated by Approximate Value Iteration satisfies

$$\limsup_{k\to\infty} \|v_* - v^{\pi_k}\|_\infty \leq \frac{2\gamma \left(C(\nu)\right)^{1/p}}{(1-\gamma)^2} \sup_{k\geq 0} \|\epsilon_k\|_{p,\nu}.$$

The main difference between the bounds of Propositions 3 and 4 and that of Bertsekas and Tsitsiklis (Equation 5) is that the approximation error ϵ_k is measured by the weighted L_p norm. As $\lim_{p\to\infty} \|.\|_{p,\mu} \leq \|.\|_\infty$ and $\left(C(\nu)\right)^{1/p} \overset{p\to\infty}{\longrightarrow} 1$, Munos's results are strictly finer.

There is generally no guarantee that AVI converges. AVI converges for specific approximation architectures called *averagers* (Gordon, 1995) which include state aggregation (see (Van Roy, 2006) for a very fine approximation bound in this specific case). Also, convergence may just occur experimentally. Assume that the sequence $(v_k)_{k\geq 0}$ tends to some value v. Write π the corresponding greedy policy. Notice this implies that $(\epsilon_k)_{k\geq 0}$ tends to $v - \mathcal{T}_* v$, that is called the **Bellman residual**. The above bounds can be improved by a factor $\frac{1}{1-\gamma}$. We know from Williams and Baird (1993) that

$$\|v_* - v^\pi\|_\infty \leq \frac{2\gamma}{1-\gamma} \|v - \mathcal{T}_* v\|_\infty$$

and, with the same notations as above, Munos (2007) derived the analogous finer L_p bound:

Corollary 5 (Performance of AVI in case of convergence (Munos, 2007))
Let $C(\nu)$ be the concentration coefficient defined in Equation 6. Assume that $(v_k)_{k\geq 0}$ tends to some value v. Write π the corresponding greedy policy. Then

$$\|v_* - v^\pi\|_\infty \leq \frac{2\gamma \left(C(\nu)\right)^{1/p}}{1-\gamma} \|v - \mathcal{T}_* v\|_{p,\nu}.$$

Eventually, let us mention that Munos (2007) and Farahmand *et al.* (2010) consider some *finer* performance bounds (in weighted L_p norm) using some *finer* concentration coefficients. We won't discuss them in this paper and we recommend the interested reader to go through these references for further details.

Algorithm 2 `Policy Iteration`

Input: An MDP, an initial policy π_0
Output: An (approximately) optimal policy
  $k \leftarrow 0$
  repeat
    $v_k \leftarrow (I - \gamma P^{\pi_k})^{-1} r^{\pi_k} + \epsilon_k$ // Estimate the value of π_k
    $\pi_{k+1} \leftarrow \text{greedy}(v_k)$ // Update the policy
    $k \leftarrow k + 1$
  until some stopping criterion
  Return π_k

2.3 Policy Iteration

Policy Iteration is an alternative method for computing an optimal policy for an infinite-horizon discounted Markov Decision Process. This algorithm is based on the following property: if π is some policy, then any policy π' that is greedy with respect to the value of π, that is any π' satisfying $\pi' = \text{greedy}(v^\pi)$, is better than π in the sense that $v^{\pi'} \geq v^\pi$. Policy Iteration exploits this property in order to generate a sequence of policies with increasing values. It is described in Algorithm 2. Note that we use the analytical form of the value of a policy given by Equation 3. Also, as for Value Iteration, our description includes a potential error ϵ_k term each time the value of a policy is estimated.

Properties of Exact Policy Iteration When the state space and the control spaces are finite, Exact Policy Iteration converges to an optimal policy π_* in a finite number of iterations (Puterman, 1994; Bertsekas and Tsitsiklis, 1996). In infinite state spaces, if the function $v \mapsto P^{\text{greedy}(v)}$ is Lipschitz, then it can be shown that Policy Iteration has a quadratic convergence rate (Puterman, 1994). However, to our knowledge, and contrary to Value Iteration, finite-time stopping conditions such as that of Proposition 1 are not widely known for Policy Iteration, though they appear implicitely in some recent works on Approximate Policy Iteration (Antos *et al.*, 2007, 2008; Farahmand *et al.*, 2010; Lazaric *et al.*, 2010).

Properties of Approximate Policy Iteration (API) For problems of interest, one usually uses Policy Iteration in an approximate form, that is with $\epsilon_k \neq 0$. Results similar to those presented for Approximate Value Iteration exist for Approximate Policy Iteration. As soon as there is some error $\epsilon_k \neq 0$, the algorithm does not necessarily converge anymore but there is an analogue of Equation 5 which is also due to Bertsekas and Tsitsiklis (1996, Prop 6.2 p. 276): if the approximation errors are uniformly bounded, then the difference between the asymptotic performance of policies π_{k+1} greedy with respect to v_k and the optimal policy is

$$\limsup_{k\to\infty} \|v_* - v^{\pi_k}\|_\infty \leq \frac{2\gamma}{(1-\gamma)^2} \sup_{k\geq 0} \|\epsilon_k\|_\infty . \tag{7}$$

As for Value Iteration, Munos has extended this result so that one can get bounds involving the L_p norm. He also showed how to relate the performance analysis to the Bellman residual $v_k - \mathcal{T}^{\pi_k} v_k$ that says how much v_k approximates the real value of the policy π_k; this is for

instance interesting when the evaluation step of Approximate Policy Iteration involves the minimization of the norm of this Bellman residual (see (Munos, 2003)). It is important to note that this Bellman residual is different from the one we introduced in the previous section (we then considered $v_k - \mathcal{T}_* v_k = v_k - \mathcal{T}^{\pi_{k+1}} v_k$ where π_{k+1} is greedy with respect to v_k). To avoid confusion, and because it is related to some specific policy, we call $v_k - \mathcal{T}^{\pi_k} v_k$ the **Policy Bellman residual**. Munos started by deriving a componentwise analysis. Write $P_k = P^{\pi_k}$ the stochastic matrix corresponding to the policy π_k which is greedy with respect to v_{k-1}, P_* the stochastic matrix corresponding to the (unknown) optimal policy π_*.

Lemma 6 (Asymptotic Componentwise Performance of API (Munos, 2003)) *The following matrices*

$$\begin{aligned} R_k &:= (1-\gamma)^2 (I - \gamma P_*)^{-1} P_{k+1} (I - \gamma P_{k+1})^{-1} \\ R'_k &:= (1-\gamma)^2 (I - \gamma P_*)^{-1} \left[P_* + \gamma P_{k+1} (I - \gamma P_{k+1})^{-1} P_k \right] \\ R''_k &:= (1-\gamma)^2 (I - \gamma P_*)^{-1} P_* (I - \gamma P_k) \end{aligned}$$

are stochastic and the asymptotic performance of the policies generated by Approximate Policy Iteration satisfies

$$\begin{aligned} \limsup_{k\to\infty} \; v_* - v^{\pi_k} &\leq \frac{2\gamma}{(1-\gamma)^2} \limsup_{k\to\infty} \; \left[R_k - R'_k \right] \epsilon_k \\ \limsup_{k\to\infty} \; v_* - v^{\pi_k} &\leq \frac{2\gamma}{(1-\gamma)^2} \limsup_{k\to\infty} \; \left[R_k - R''_k \right] (v_k - \mathcal{T}^{\pi_k} v_k). \end{aligned}$$

As for Value Iteration, the above componentwise bound leads to the following L_p bounds.

Proposition 7 (Asymptotic Performance of API (1/2) (Munos, 2003)) *Choose any p and any distribution μ. Consider the notations of Lemma 6. For all $k \geq 0$,*

$$\mu_k := \frac{1}{2} \left(R_k + R'_k \right)^T \mu \quad \textit{and} \quad \mu'_k := \frac{1}{2} \left(R_k + R''_k \right)^T \mu$$

are distributions and the asymptotic performance of the policies generated by Approximate Policy Iteration satisfies

$$\begin{aligned} \limsup_{k\to\infty} \; \left\| v_* - v^{\pi_k} \right\|_{p,\mu} &\leq \frac{2\gamma}{(1-\gamma)^2} \limsup_{k\to\infty} \; \left\| \epsilon_k \right\|_{p,\mu_k} \\ \textit{and} \quad \limsup_{k\to\infty} \; \left\| v_* - v^{\pi_k} \right\|_{p,\mu} &\leq \frac{2\gamma}{(1-\gamma)^2} \limsup_{k\to\infty} \; \left\| v_k - \mathcal{T}^{\pi_k} v_k \right\|_{p,\mu'_k}. \end{aligned}$$

Using the concentration coefficient $C(\nu)$ introduced in the previous section (Equation 6), it is also possible to show[6] the following L_∞ / L_p bounds:

6. Similarly to footnote 5, this result is not explicitely stated by Munos (2003) but using techniques of another of his articles (Munos, 2007), it can be derived from Lemma 6. The current paper anyway generalizes this result (in Proposition 26 page 25).

Proposition 8 (Asymptotic Performance of API (2/2))
Let $C(\nu)$ be the concentration coefficient defined in Equation 6. The asymptotic performance of the policies generated by Approximate Policy Iteration satisfies

$$\begin{aligned}\limsup_{k\to\infty} \|v_* - v^{\pi_k}\|_\infty &\leq \frac{2\gamma\left(C(\nu)\right)^{1/p}}{(1-\gamma)^2} \limsup_{k\to\infty} \|\epsilon_k\|_{p,\nu} \\ \limsup_{k\to\infty} \|v_* - v^{\pi_k}\|_\infty &\leq \frac{2\gamma\left(C(\nu)\right)^{1/p}}{(1-\gamma)^2} \limsup_{k\to\infty} \|v_k - \mathcal{T}^{\pi_k} v_k\|_{p,\nu}.\end{aligned}$$

Again, the bounds of Propositions 7 and 8 with respect to the approximation error ϵ_k are finer than that of Bertsekas and Tsitsiklis (Equation 7). Compared to the similar result for Approximate Value Iteration (Propositions 3 and 4) where the bound depends on a *uniform* error bound ($\forall k, \|\epsilon_k\|_{p,\nu} \leq \epsilon$), the above bounds have the nice property that they only depend on *asymptotic* errors/residuals.

Finally, as for Approximate Value Iteration, a better bound (by a factor $\frac{1}{1-\gamma}$) might be obtained if the sequence of policies happens to converge. It can be shown (Munos, 2003, Remark 4 page 7) that:

Corollary 9 (Performance of API in case of convergence)
Let $C(\nu)$ be the concentration coefficient defined in Equation 6. If the sequence of policies (π_k) converges to some π, then

$$\begin{aligned}v_* - v^{\pi} &\leq \frac{2\gamma\left(C(\nu)\right)^{1/p}}{1-\gamma} \limsup_{k\to\infty} \|\epsilon_k\|_{p,\nu} \\ v_* - v^{\pi} &\leq \frac{2\gamma\left(C(\nu)\right)^{1/p}}{1-\gamma} \limsup_{k\to\infty} \|v_k - \mathcal{T}^{\pi_k} v_k\|_{p,\nu}.\end{aligned}$$

After this tour of results for Value and Policy Iteration, we now introduce the algorithm studied in this paper.

3. λ Policy Iteration

Though all the results we have emphasized so far are strongly related (and even sometimes identical, compare Equations 5 and 7), they were surprisingly proved independently. In this section, we describe the family of algorithms "λ Policy Iteration"[7] introduced by Bertsekas and Ioffe (1996) parameterized by a coefficient $\lambda \in (0, 1)$, that generalizes them both. When $\lambda = 0$, λ Policy Iteration reduces to Value Iteration while it reduces to Policy Iteration when $\lambda = 1$. We also recall the fact discussed by Bertsekas and Ioffe (1996) that λ Policy Iteration draws some connections with Temporal Difference algorithms (Sutton and Barto, 1998).

3.1 The Algorithm

We begin by giving some intuition about how one can make a connection between Value Iteration and Policy Iteration. For simplicity, let us temporarily forget about the error

7. It was also called "Temporal Difference-Based Policy Iteration" in the original paper, but we take the name λ Policy Iteration, as it was the name picked by most subsequent works.

term ϵ_k. At first sight, Value Iteration builds a sequence of value functions and Policy Iteration a sequence of policies. In fact, both algorithms can be seen as updating a sequence of value-policy pairs. With some little rewriting — by decomposing the (nonlinear) Bellman operator $\mathcal{T}_*$ into *(i)* the maximization step and *(ii)* the application of the (linear) Bellman operator — it can be seen that each iterate of Value Iteration is equivalent to the two following updates:

$$\left\{ \begin{array}{lcl} \pi_{k+1} & \leftarrow & \text{greedy}(v_k) \\ v_{k+1} & \leftarrow & \mathcal{T}^{\pi_{k+1}} v_k \end{array} \right. \Leftrightarrow \left\{ \begin{array}{lcl} \pi_{k+1} & \leftarrow & \text{greedy}(v_k) \\ v_{k+1} & \leftarrow & r^{\pi_{k+1}} + \gamma P^{\pi_{k+1}} v_k. \end{array} \right.$$

The left hande side of the above equation uses the operator $\mathcal{T}^{\pi_{k+1}}$ while the right hande side uses its definition. Similarly — by inverting in Algorithm 2 the order of *(i)* the estimation of the value of the current policy and *(ii)* the update of the policy, and by using the fact that the value of the policy π_{k+1} is the fixed point of $\mathcal{T}^{\pi_{k+1}}$ (Equation 4) — it can be argued that every iteration of Policy Iteration does the following:

$$\left\{ \begin{array}{lcl} \pi_{k+1} & \leftarrow & \text{greedy}(v_k) \\ v_{k+1} & \leftarrow & (\mathcal{T}^{\pi_{k+1}})^{\infty} v_k \end{array} \right. \Leftrightarrow \left\{ \begin{array}{lcl} \pi_{k+1} & \leftarrow & \text{greedy}(v_k) \\ v_{k+1} & \leftarrow & (I - \gamma P^{\pi_{k+1}})^{-1} r^{\pi_{k+1}}. \end{array} \right.$$

This rewriting makes both algorithms look close to each other. Both can be seen as having an estimate v_k of the value of policy π_k, from which they deduce a potentially better policy π_{k+1}. The corresponding value $v^{\pi_{k+1}}$ of this better policy may be regarded as a target which is tracked by the next estimate v_{k+1}. The difference is in the update that enables to go from v_k to v_{k+1}: while Policy Iteration directly *jumps to* the value of π_{k+1} (by applying the Bellman operator $\mathcal{T}^{\pi_{k+1}}$ an infinite number of times), Value Iteration only *makes one step* towards it (by applying $\mathcal{T}^{\pi_{k+1}}$ only once). From this common view of Value Iteration, it is natural to introduce the well-known Modified Policy Iteration algorithm (Puterman and Shin, 1978) which *makes n steps* at each update:

$$\left\{ \begin{array}{lcl} \pi_{k+1} & \leftarrow & \text{greedy}(v_k) \\ v_{k+1} & \leftarrow & (\mathcal{T}^{\pi_{k+1}})^{n} v_k \end{array} \right. \Leftrightarrow \left\{ \begin{array}{lcl} \pi_{k+1} & \leftarrow & \text{greedy}(v_k) \\ v_{k+1} & \leftarrow & \left[I + ... + (\gamma P^{\pi_{k+1}})^{n-1} \right] r^{\pi_{k+1}} + (\gamma P^{\pi_{k+1}})^{n} v_k. \end{array} \right.$$

The above common view is actually here interesting because it also leads to a natural introduction of λ Policy Iteration. λ Policy Iteration is doing a λ-*adjustable step* towards the value of π_{k+1}:

$$\left\{ \begin{array}{lcl} \pi_{k+1} & \leftarrow & \text{greedy}(v_k) \\ v_{k+1} & \leftarrow & (1-\lambda) \sum_{j=0}^{\infty} \lambda^j (\mathcal{T}^{\pi_{k+1}})^{j+1} v_k \end{array} \right.$$

$$\Leftrightarrow \left\{ \begin{array}{lcl} \pi_{k+1} & \leftarrow & \text{greedy}(v_k) \\ v_{k+1} & \leftarrow & (I - \lambda\gamma P^{\pi_{k+1}})^{-1} (r^{\pi_{k+1}} + (1-\lambda)\gamma P^{\pi_{k+1}} v_k) \end{array} \right.$$

Remark 10 *The equivalence between the left and the right representation of λ Policy Iteration needs here to be proved. For all $k \geq 0$ and all function v, Bertsekas and Ioffe (1996) introduce the following operator*[8]

$$M_k \mathbf{v} := (1-\lambda)\mathcal{T}^{\pi_{k+1}} v_k + \lambda \mathcal{T}^{\pi_{k+1}} \mathbf{v} \quad (8)$$

$$= r^{\pi_{k+1}} + (1-\lambda)\gamma P^{\pi_{k+1}} v_k + \lambda\gamma P^{\pi_{k+1}} \mathbf{v} \quad (9)$$

and prove that

8. The equivalence between Equations 8 and 9 follows trivially from the definition of $\mathcal{T}^{\pi_{k+1}}$.

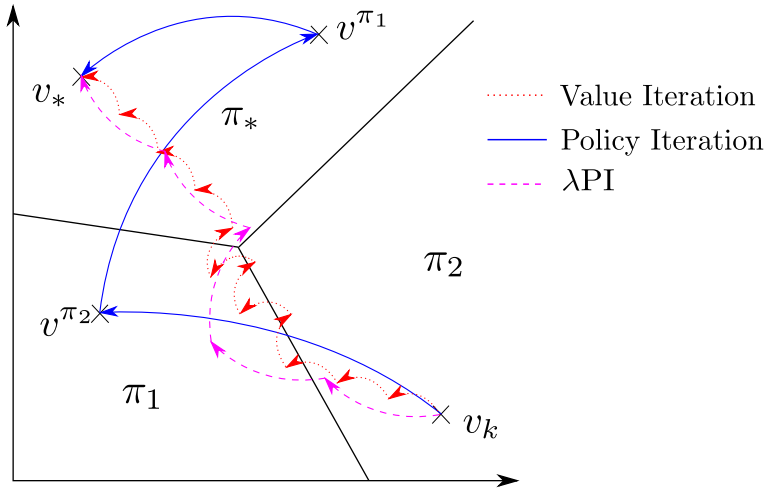

Figure 1: **Visualizing λ Policy Iteration on the greedy partition sketch:** Following (Bertsekas and Tsitsiklis, 1996, page 226), one can decompose the value space as a collection of polyhedra, such that each polyhedron corresponds to a region where one policy is greedy. This is called the *greedy partition*. In the above example, there are only 3 policies, π_1, π_2 and π_*. v_k is the initial value. greedy$(v_k) = \pi_2$, greedy$(v^{\pi_2}) = \pi_1$, and greedy$(v^{\pi_1}) = \pi_*$. Therefore (1-)Policy Iteration generates the sequence $((\pi_2, v^{\pi_2}), (\pi_1, v^{\pi_1}), (\pi_*, v^{\pi_*}))$. Value teration (or 0 Policy Iteration) starts by slowly updating v_k towards v^{π_2} until it crosses the boundary π_1/π_2, after which it tracks alternatively v^{π_1} and v^{π_2}, until it reaches the π_* part. In other words, Value Iteration makes small steps. λ Policy Iteration is intermediate between the two: it makes steps of which the steps length is related to λ. On the above sketch, λ Policy Iteration gets to the greedy partition of the optimal policy π_* in 3 steps, as did Policy Iteration.

- *M_k is a contraction mapping of modulus $\lambda\gamma$ for the max norm ;*
- *The next iterate v_{k+1} of λ Policy Iteration is the (unique) fixed point of M_k.*

The left representation of λ Policy Iteration is obtained by "unrolling" Equation 8 an infinite number of times, while the right one is obtained by using Equation 9 and solving the linear system $v_{k+1} = M_k v_{k+1}$.

Informally, the parameter λ (or n in the case of Modified Policy Iteration) can be seen as adjusting the size of the step for tracking the target $v^{\pi_{k+1}}$ (see Figure 1): the bigger the value, the longer the step. Formally, λ Policy Iteration (consider the above left hand side) consists in doing a geometric average of parameter λ of the different numbers of applications of the Bellman operator $(\mathcal{T}^{\pi_{k+1}})^j$ to v_k. The right hand side is here interesting because it clearly shows that λ Policy Iteration generalizes Value Iteration (when $\lambda = 0$) and Policy Iteration (when $\lambda = 1$). The operator M_k gives some insight on how one may concretely implement one iteration of λ Policy Iteration: it can for instance be done through a Value

Algorithm 3 λ `Policy Iteration`

Input: An MDP, $\lambda \in (0,1)$, an initial value v_0
Output: An (approximately) optimal policy
  $k \leftarrow 0$
  repeat
    $\pi_{k+1} \leftarrow \text{greedy}(v_k)$ // Update the policy
    $v_{k+1} \leftarrow \mathcal{T}_\lambda^{\pi_{k+1}} v_k + \epsilon_{k+1}$ // Update the estimate of the value of policy π_{k+1}
    $k \leftarrow k+1$
  until some convergence criterion
  Return greedy(v_k)

Iteration-like algorithm which applies M_k iteratively. Also, the fact that its contraction factor $\lambda\gamma$ can be tuned is of particular importance because finding the corresponding fixed point can be much easier than that of $\mathcal{T}^{\pi_{k+1}}$, which is only γ-contracting.

In order to describe the λ Policy Iteration algorithm, it is useful to introduce an operator that corresponds to computing the fixed point of M_k. For any value v and any policy π, define:

$$\begin{aligned} \mathcal{T}_\lambda^\pi v &:= v + (I - \lambda\gamma P^\pi)^{-1}(\mathcal{T}^\pi v - v) && (10)\\ &= (I - \lambda\gamma P^\pi)^{-1}(r^\pi + (1-\lambda)\gamma P^\pi v) && (11)\\ &= (I - \lambda\gamma P^\pi)^{-1}(\lambda r^\pi + (1-\lambda)\mathcal{T}^\pi v). && (12) \end{aligned}$$

Equation 11 indeed amounts to solve Equation 9 defining M_k. The other two formulations are equivalent up to some little linear algebra manipulations.

λ Policy Iteration is formally described in Algorithm 3. Once again, our description includes a potential error term each time the value is updated. Even with this error term, it is straightforward to see that the algorithm reduces to Value Iteration (Algorithm 1) when $\lambda = 0$ and to Policy Iteration[9] (Algorithm 2) when $\lambda = 1$.

Relation with Reinforcement Learning The definition of the operator $\mathcal{T}_\lambda^\pi$ given by Equation 11 is the form we have used for the introduction of λ Policy Iteration as an intermediate algorithm between Value Iteration and Policy Iteration. The equivalent form given by Equation 10 can be used to make a connection with the TD(λ) algorithms[10] (Sutton and Barto, 1998). Indeed, through Equation 10, the evaluation phase of λ Policy Iteration can be seen as an incremental additive procedure:

$$v_{k+1} \leftarrow v_k + \Delta_k$$

where

$$\Delta_k := (I - \lambda\gamma P^{\pi_{k+1}})^{-1}(\mathcal{T}^{\pi_{k+1}} v_k - v_k)$$

9. Policy Iteration starts with an initial policy while λ Policy Iteration starts with some initial value. To be precise, 1 Policy Iteration starting with v_0 is equivalent to Policy Iteration starting with the greedy policy with respect to v_0.
10. TD stands for Temporal Difference. As we have mentionned in Footnote 7, λ Policy Iteration was originally also called "Temporal Difference Based Policy Iteration" and the presentation of Bertsekas and Ioffe (1996) starts from the formulation of Equation 10 (which is close to TD(λ)), and afterwards makes the connection with Value Iteration and Policy Iteration.

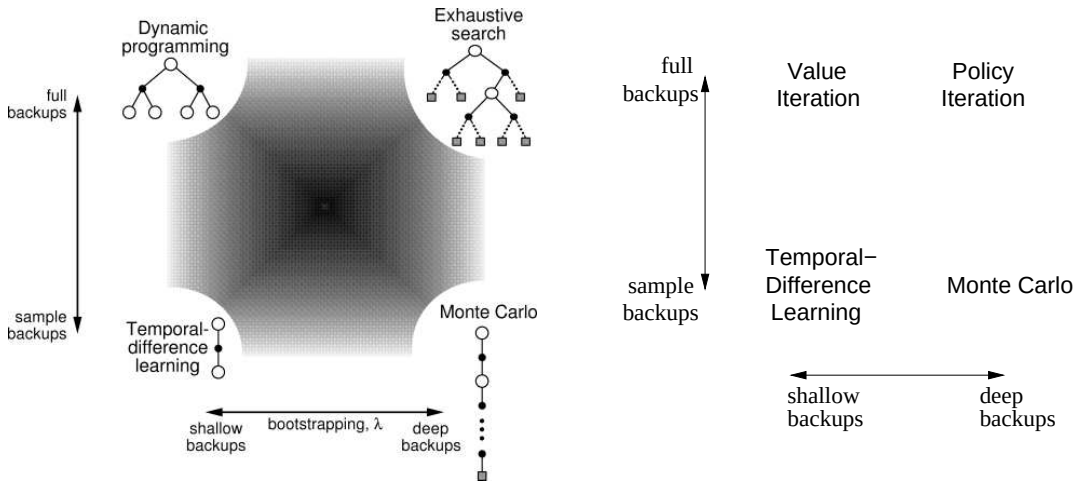

Figure 2: λ **Policy Iteration, a fundamental algorithm for Reinforcement Learning:** The left drawing, taken from chapter 10.1 the book of Sutton and Barto (1998), represents *"two of the most important dimensions"* of Reinforcement Learning methods. The vertical axis corresponds to whether one does full backup (exact computation of the expectations) or stochastic approximation (estimation through samples). The horizontal axis corresponds to the depth of the backups, and is (among other things) controlled by the parameter λ. On the right, is an original picture of λ Policy Iteration, along the same dimensions. It is interesting to notice that Sutton and Barto (1998) comment their drawing as follows: *"At three of the four corners of the space are the three primary methods for estimating values: DP, TD, and Monte Carlo"*. They do not recognize the fourth corner as one of the Reinforcement Learning *primary methods*. The natural representation of λ Policy Iteration actually suggests a modification of the sketch which is particularly meaningful: Policy Iteration, which consists in computing the value of the current policy, is the *deepest backup method*, and can be considered as the batch version of Monte Carlo.

is zero if and only if the value v_k is equal to the optimal value v_*. It can be shown (see Bertsekas and Ioffe (1996) for a proof or simply look at the equivalence between Equations 2 and 3 for an intuition) that the vector Δ_k has components given by:

$$\Delta_k(i) = \lim_{N\to\infty} E_{\pi_{k+1}}\left[\sum_{j=0}^{N-1}(\lambda\gamma)^j \delta_k(i_j, i_{j+1}) \middle| i_0 = i\right] \tag{13}$$

with

$$\delta_k(i,j) := r(i,\pi_{k+1}(i),j) + \gamma v(j) - v(i)$$

being the temporal difference associated to transition $i \to j$, as defined by Sutton and Barto (1998). When one uses a stochastic approximation of λ Policy Iteration, that is when the expectation $E_{\pi_{t+1}}$ is approximated by sampling, λ Policy Iteration reduces to the algorithm

TD(λ) which is described in chapter 7 of Sutton and Barto (1998). In particular, when $\lambda = 1$, the terms in the above sum collapse and become the exact discounted return:

$$\begin{aligned}\sum_{j=0}^{N-1} \gamma^j \delta_k(i_j, i_{j+1}) &= \sum_{j=0}^{N-1} \gamma^j \left[r(i_j, \pi_{k+1}(i_j), i_{j+1}) + \gamma v(i_{j+1}) - v(i_j) \right] \\ &= \sum_{j=0}^{N-1} \gamma^j r(i_j, \pi_{k+1}(i_j), i_{j+1}) + \gamma^N v(i_N)\end{aligned}$$

and the stochastic approximation matches the Monte-Carlo method. Also, Bertsekas and Ioffe (1996) show that Approximate TD(λ) with a linear feature architecture, as described in chapter 8.2 of Sutton and Barto (1998), corresponds to a natural Approximate version of λ Policy Iteration where the value is updated by least square fitting using a gradient-type iteration after each sample. Last but not least, the reader might notice that the "unified view" of Reinforcement Learning algorithms which is depicted in chapter 10.1 of Sutton and Barto (1998), and which is reproduced in Figure 2, is in fact a picture of λ Policy Iteration.

3.2 Analysis of Exact λ Policy Iteration

To our knowledge, little has been done concerning the analysis of λ Policy Iteration: the only results available concern the Exact case (when $\epsilon_k = 0$). Define the following factor

$$\beta = \frac{(1-\lambda)\gamma}{1-\lambda\gamma}. \tag{14}$$

We have $0 \leq \beta \leq \gamma < 1$. If $\lambda = 0$ (Value Iteration) then $\beta = \gamma$, and if $\lambda = 1$ (Policy Iteration) then $\beta = 0$. In the original article introducing λ Policy Iteration, Bertsekas and Ioffe (1996) show the convergence and provide an asymptotic rate of convergence:

Proposition 11 (Convergence of Exact λPI (Bertsekas and Ioffe, 1996))
If the discount factor $\gamma < 1$, then v_k converges to v_*. Furthermore, after some index k_*, the rate of convergence is linear in β as defined in Equation 14, that is

$$\forall k \geq k_*, \quad \|v_{k+1} - v_*\| \leq \beta \|v_k - v_*\|.$$

By making λ close to 1, β can be arbitrarily close to 0 so the above rate of convergence might look overly impressive. This needs to be put into perspective: the index k_* is the index after which the policy π_k does not change anymore (and is equal to the optimal policy π_*). As we said when we introduced the algorithm, λ controls the speed at which one wants v_k to "track the target" $v^{\pi_{k+1}}$; when $\lambda = 1$, this is done in one step (and if $\pi_{k+1} = \pi_*$ then $v_{k+1} = v_*$).

4. Overview of our Results and Main Proof Ideas

Now that we have described the algorithms and some of their known properties, motivating the remaining of this paper is straightforward. λ Policy Iteration is conceptually nice since it generalizes the two most well-known algorithms for solving discounted infinite-horizon

Markov Decision Processes. The natural question that arises is whether one can generalizes the results we have described so far to λ Policy Iteration (uniformly for all λ). The answer is yes:

- we shall derive a componentwise analysis of Exact and Approximate λ Policy Iteration;
- we shall characterize the *non-asymptotic* convergence rate of Exact λ Policy Iteration (Proposition 11 only showed the *asymptotic* linear convergence) and shall generalize the stopping criterion described for Value Iteration (Proposition 1);
- we shall give bounds of the asymptotic error of Approximate λ Policy Iteration with respect to the *asymptotic* approximation error, Bellman residual, and Policy Bellman residual, that generalize Lemmas 2 and 6, and Propositions 3, 4, 7 and 8; our analysis actually implies that doing Approximate λ Policy Iteration is sound (when the approximation error tends to 0, the algorithm recovers the optimal solution)
- we shall provide specific (better) bounds for the case when the value or the policy converges, which generalizes Corollaries 5 and 9;
- interestingly, we shall provide all our results using the span seminorms we have introduced at the beginning of the paper, and using the relations between this span semi-norms and the standard L_p norms (Equation 1), it can be seen that our results are in this respect slightly stronger than all the previously described results.

Conceptually, we provide a unified vision (unified proofs, unified results) for all the mentionned algorithms.

4.1 On the Need for a New Proof Technique

In the literature, lines of analysis are different for Value Iteration and Policy Iteration. Analyses of Value Iteration are based on the fact that it computes the fixed point of the Bellman operator which is a γ-contraction mapping in max norm (see for instance (Bertsekas and Tsitsiklis, 1996)). Unfortunately, it can be shown that the operator by which Policy Iteration updates the value from one iteration to the next is in general not a contraction in max norm. In fact, this observation can be drawn for λ Policy Iteration as soon as it does not reduce to Value Iteration:

Proposition 12 *If $\lambda > 0$, there exists no norm for which the operator by which λ Policy Iteration updates the value from one iteration to the next is a contraction.*

Proof To see this, consider the deterministic MDP (shown in Figure 3) with two states $\{1, 2\}$ and two actions $\{change, stay\}$: $r_1 = 0$, $r_2 = 1$, $P_{change}(s_2|s_1) = P_{change}(s_1|s_2) = P_{stay}(s_1|s_1) = P_{stay}(s_2|s_2) = 1$. Consider the following two value functions $v = (\epsilon, 0)$ and $v' = (0, \epsilon)$ with $\epsilon > 0$. Their corresponding greedy policies are $\pi = (stay, change)$ and

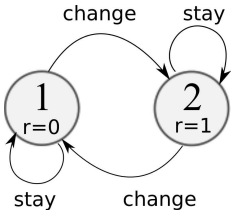

Figure 3: This simple deterministic MDP is used to show that λ Policy Iteration cannot be analysed in terms of contraction (see text for details).

$\pi' = (change, stay)$. Then, we can compute the next iterates of v and v' (using Equation 11):

$$\begin{aligned} r^{\pi} + (1-\lambda\gamma)P^{\pi}v &= \begin{pmatrix} (1-\lambda)\gamma\epsilon \\ 1+(1-\lambda)\gamma\epsilon \end{pmatrix}, \\ \mathcal{T}_{\lambda}^{\pi}v &= \begin{pmatrix} \frac{(1-\lambda)\gamma\epsilon}{1-\lambda\gamma} \\ 1+\frac{(1-\lambda)\gamma\epsilon}{1-\lambda\gamma} \end{pmatrix}, \\ r^{\pi'} + (1-\lambda\gamma)P^{\pi'}v' &= \begin{pmatrix} (1-\lambda)\gamma\epsilon \\ 1+(1-\lambda)\gamma\epsilon \end{pmatrix}, \\ \text{and } \mathcal{T}_{\lambda}^{\pi'}v' &= \begin{pmatrix} \frac{1+(1-\lambda)\gamma\epsilon}{1-\lambda\gamma} - 1 \\ \frac{1+(1-\lambda)\gamma\epsilon}{1-\lambda\gamma} \end{pmatrix}. \end{aligned}$$

Then

$$\mathcal{T}_{\lambda}^{\pi'}v' - \mathcal{T}_{\lambda}^{\pi}v = \begin{pmatrix} \frac{1}{1-\lambda\gamma} - 1 \\ \frac{1}{1-\lambda\gamma} - 1 \end{pmatrix}$$

while

$$v' - v = \begin{pmatrix} -\epsilon \\ \epsilon \end{pmatrix}.$$

As ϵ can be arbitrarily small, the norm of $\mathcal{T}_{\lambda}^{\pi}v - \mathcal{T}_{\lambda}^{\pi'}v'$ can be arbitrarily larger than norm of $v - v'$ when $\lambda > 0$. ■

Analyses of Policy Iteration usually rely on the fact that the sequence of values generated by Exact Policy Iteration is non-decreasing (see Bertsekas and Tsitsiklis (1996); Munos (2003)). Unfortunately, it can easily be seen that as soon as λ is smaller than 1, the value functions may decrease (it suffices to take a very high initial value). For non trivial values of λ, λ Policy Iteration is neither contracting nor non-decreasing, so we need a new proof technique.

4.2 An Overview on the Componentwise Analysis of λ Policy Iteration

The rest of this section provides an overview of our analysis. We show how to compute an upper bound of the loss for λ Policy Iteration in the general (possibly approximate) case. It is the basis for the derivation of componentwise bounds for Exact λ Policy Iteration

(Section 5) and Approximate λ Policy Iteration (Section 6). Consider λ Policy Iteration as described in Algorithm 3, and the sequences of value-policy-error triplets (v_k, π_k, ϵ_k) it generates. Most of our results come from a series of relations involving objects we now define:

- the **loss** of using policy π_k instead of the optimal policy:

$$l_k := v_* - v^{\pi_k};$$

- the **value** of the k^{th} iterate b.a. (before approximation):

$$w_k := v_k - \epsilon_k;$$

- the **distance** between the optimal value and the k^{th} value b.a.:

$$d_k := v_* - w_k = \mathcal{T}_\lambda^{\pi_k} v_{k-1};$$

- the **shift** between the k^{th} value b.a. and the value of the k^{th} policy:

$$s_k := w_k - v^{\pi_k};$$

- the **Bellman residual** of the k^{th} value:

$$b_k := \mathcal{T}_{k+1} v_k - v_k = \mathcal{T}_* v_k - v_k.$$

To lighten the notations, we now on write: $P_k := P^{\pi_k}$, $\mathcal{T}_k := \mathcal{T}^{\pi_k}$, $P_* := P^{\pi_*}$. We refer to the factor β as introduced by Bertsekas and Ioffe (Equation 14 page 15). Also, the following stochastic matrix plays a recurrent role in our analysis[11]:

$$A_k := (1 - \lambda\gamma)(I - \lambda\gamma P_k)^{-1} P_k. \tag{15}$$

We use the notation $\overline{x}$ for an upper bound of x and $\underline{x}$ for a lower bound.

Our analysis relies on a series of lemmas that we now state (for clarity, all the proofs are deferred to Appendix A).

Lemma 13 *The shift is related to the Bellman residual as follows:*

$$s_k = \beta(I - \gamma P_k)^{-1} A_k(-b_{k-1}).$$

Lemma 14 *The Bellman residual at iteration $k+1$ cannot be much lower than that at iteration k:*

$$b_{k+1} \geq \beta A_{k+1} b_k + x_{k+1}$$

where $x_k := (\gamma P_k - I)\epsilon_k$ only depends on the approximation error.

11. The fact that this is indeed a stochastic matrix is explained at the beginning of the Appendices.

As a consequence, a lower bound of the Bellman residual is[12]:

$$b_k \geq \sum_{j=k_0+1}^{k} \beta^{k-j} \left(A_k A_{k-1} ... A_{j+1}\right) x_j + \beta^{k-k_0} \left(A_k A_{k-1} ... A_{k_0+1}\right) b_{k_0} := \underline{b_k}$$

where k_0 is some arbitrary reference index. Using Lemma 13, the bound on the Bellman residual also provides an upper on the shift[13]:

$$s_k \leq \beta(I - \gamma P_k)^{-1} A_k(-\underline{b_{k-1}}) := \overline{s_k}$$

Lemma 15 *The distance at iteration $k+1$ cannot be much greater than that at iteration k:*

$$d_{k+1} \leq \gamma P_* d_k + y_k$$

where $y_k := \frac{\lambda\gamma}{1-\lambda\gamma} A_{k+1}(-\underline{b_k}) - \gamma P_* \epsilon_k$ depends on the lower bound of the Bellman residual and the approximation error.

Then, an upper bound of the distance is[14]:

$$d_k \leq \sum_{j=k_0}^{k-1} \gamma^{k-1-j} (P_*)^{k-1-j} y_j + \gamma^{k-k_0} (P_*)^{k-k_0} d_{k_0} = \overline{d_k}.$$

Eventually, as

$$l_k = d_k + s_k \leq \overline{d_k} + \overline{s_k},$$

the upper bounds on the distance and the shift enable us to derive the upper bound on the loss.

Remark 16 *The above derivation is a generalization of that of Munos (2003) for Approximate Policy Iteration. Note however that it is not a trivial generalization: when $\lambda = 1$, that is when both proofs coincide, $\beta = 0$ and Lemmas 13 and 14 have the following particularly simple form: $s_k = 0$ and $b_{k+1} \geq x_{k+1}$.*

The next two sections contain our main results, which take the form of performance bounds when using λ Policy Iteration. Section 5 gathers the results concerning Exact λ Policy Iteration, while Section 6 presents those concerning Approximate λ Policy Iteration.

5. Performance Bounds for Exact λ Policy Iteration

Consider Exact λ Policy Iteration for which we have $\epsilon_k = 0$ for all k. Let k_0 be some arbitrary index. By exploiting the recursive relations we have described in the previous section (this process is detailed in Appendix B), we can derive the following componentwise bounds for the loss:

12. We use the property here that if some vectors satisfy the componentwise inequality $x \leq y$, and if P is a stochastic matrix, then the componentwise inequality $Px \leq Py$ holds.
13. We use the fact that $(1-\gamma)(I-\gamma P_k)^{-1}$ is a stochastic matrix (see Footnote 11) and Footnote 12.
14. See Footnote 12.

Lemma 17 (Componentwise Rate of Convergence of Exact λPI) *For all $k > k_0$, the following matrices*

$$\begin{aligned} E_{kk_0} &:= (1-\gamma)(P_*)^{k-k_0}(I-\gamma P_*)^{-1}, \\ E'_{kk_0} &:= \left(\frac{1-\gamma}{\gamma^{k-k_0}}\right)\left(\frac{\lambda\gamma}{1-\lambda\gamma}\sum_{j=k_0}^{k-1}\gamma^{k-1-j}\beta^{j-k_0}(P_*)^{k-1-j}A_{j+1}A_j...A_{k_0+1}\right. \\ &\qquad \left. + \quad \beta^{k-k_0}(I-\gamma P_k)^{-1}A_kA_{k-1}...A_{k_0+1}\right), \end{aligned}$$

and $F_{kk_0} := (1-\gamma)P_*^{k-k_0} + \gamma E'_{kk_0}P_*$

are stochastic and the performance of the policies generated by λ Policy Iteration satisfies

$$v_* - v^{\pi_k} \leq \frac{\gamma^{k-k_0}}{1-\gamma}\left[F_{kk_0} - E'_{kk_0}\right](v_* - v_{k_0}), \tag{16}$$

$$v_* - v^{\pi_k} \leq \frac{\gamma^{k-k_0}}{1-\gamma}\left[E_{kk_0} - E'_{kk_0}\right](\mathcal{T}_*v_{k_0} - v_{k_0}), \quad \textit{and} \tag{17}$$

$$v_* - v^{\pi_k} \leq \gamma^{k-k_0}\left[(P_*)^{k-k_0}\left((v_* - v_{k_0}) - \min_s[v_*(s) - v_{k_0}(s)]e\right) + \|v_*(s) - v^{\pi_{k_0+1}}\|_\infty \mathbf{e}\right] \tag{18}$$

In order to derive (more interpretable) span seminorms bounds from the above componentwise bound, we rely on the following lemma, which for clarity of exposition is proved in Appendix F.

Lemma 18 *If for some non-negative vectors x and y, some constant $K \geq 0$, and some stochastic matrices X and X' we have*

$$x \leq K(X - X')y,$$

Then

$$\|x\|_\infty \leq K \operatorname*{span}_\infty[y].$$

With this, the componentwise bounds of Lemma 17 become:

Proposition 19 (Non-asymptotic bounds for Exact λ Policy Iteration) *For any $k > k_0$,*

$$\|v_* - v^{\pi_k}\|_\infty \leq \frac{\gamma^{k-k_0}}{1-\gamma}\operatorname*{span}_\infty[v_* - v_{k_0}], \tag{19}$$

$$\|v_* - v^{\pi_k}\|_\infty \leq \frac{\gamma^{k-k_0}}{1-\gamma}\operatorname*{span}_\infty[\mathcal{T}_*v_{k_0} - v_{k_0}], \tag{20}$$

$$\textit{and} \quad \|v_* - v^{\pi_k}\|_\infty \leq \gamma^{k-k_0}\left(\operatorname*{span}_\infty[v_* - v_{k_0}] + \|v_*(s) - v^{\pi_{k_0+1}}\|_\infty\right). \tag{21}$$

This *non-asympotic* bound supplements the *asymptotic* bound of Proposition 11 from Bertsekas and Ioffe (1996). Remarkably, these bounds do not depend on the value λ: whatever the value of λ, all algorithms have the same above rates. The bound of Equation 19 is expressed in terms of the distance between the value function and the optimal value function at some iteration k_0. The second inequality, Equation 20, can be used as a stopping criterion. Indeed, taking $k = k_0 + 1$ it implies the following stopping condition, which generalizes that of Proposition 1 about Value Iteration:

Proposition 20 (Stopping condition of Exact λ Policy Iteration)
If at some iteration k_0, the value v_{k_0} satisfies:

$$\operatorname*{span}_{\infty}\left[\mathcal{T}_* v_{k_0} - v_{k_0}\right] \leq \frac{1-\gamma}{\gamma}\epsilon$$

then the greedy policy π_{k_0+1} with respect to v_{k_0} is ϵ-optimal: $\left\|v_* - v^{\pi_{k_0+1}}\right\|_\infty < \epsilon$.

The last inequality described in Equation 21 relies on the distance between the value function and the optimal value function and the value difference between the optimal policy and the first greedy policy; compared to the others, it has the advantage of not containing a $\frac{1}{1-\gamma}$ factor. To our knowledge, this bound is even new for the specific cases of Value Iteration and Policy Iteration.

6. Performance Bounds for Approximate λ Policy Iteration

We now turn to the (slightly more involved) results on Approximate λ Policy Iteration. We provide componentwise bounds of the loss $v_* - v^{\pi_k} \geq 0$ of using policy π_k instead of using the optimal policy, with respect to the approximation error ϵ_k, the Policy Bellman residual $\mathcal{T}_k v_k - v_k$ and the Bellman residual $\mathcal{T}_* v_k - v_k = \mathcal{T}_{k+1} v_k - v_k$. Recall the subtle difference between these two Bellman residuals: the Policy Bellman residual says how much v_k differs from the value of π_k while the Bellman residual says how much v_k differs from the value of the policies π_{k+1} and π_*.

The core of our analysis is based on the following lemma:

Lemma 21 (Componentwise Performance bounds for App. λ Policy Iteration)

For all $k > j \geq 0$, the following matrices

$$\begin{aligned}
B_{jk} &:= \frac{1-\gamma}{\gamma^{k-j}}\left[\frac{\lambda\gamma}{1-\lambda\gamma}\sum_{i=j}^{k-1}\gamma^{k-1-i}\beta^{i-j}(P_*)^{k-1-i}A_{i+1}A_i...A_{j+1}\right.\\
&\qquad\qquad\left. + \beta^{k-j}(I-\gamma P_k)^{-1}A_k A_{k-1}...A_{j+1}\right]\\
B'_{jk} &:= \gamma B_{jk}P_j + (1-\gamma)(P_*)^{k-j}\\
C_k &:= (1-\gamma)^2(I-\gamma P_*)^{-1}\left(P_*(I-\gamma P_k)^{-1}\right)\\
C'_k &:= (1-\gamma)^2(I-\gamma P_*)^{-1}\left(P_{k+1}(I-\gamma P_{k+1})^{-1}\right)\\
D &:= (1-\gamma)P_*(I-\gamma P_*)^{-1}\\
D'_k &:= (1-\gamma)P_k(I-\gamma P_k)^{-1}
\end{aligned}$$

are stochastic and

$$\forall k_0, \quad \limsup_{k\to\infty} \; v_* - v^{\pi_k} \;\leq\; \frac{1}{1-\gamma} \limsup_{k\to\infty} \; \sum_{j=k_0}^{k-1} \gamma^{k-j} \left[B_{jk} - B'_{jk}\right] \epsilon_j, \tag{22}$$

$$\limsup_{k\to\infty} \; v_* - v^{\pi_k} \;\leq\; \frac{\gamma}{(1-\gamma)^2} \limsup_{k\to\infty} \; [C_k - C'_k](\mathcal{T}_k v_k - v_k), \tag{23}$$

$$\textit{and} \;\; \forall k, \quad v_* - v^{\pi_k} \;\leq\; \frac{\gamma}{1-\gamma} \left[D - D'_k\right] (\mathcal{T}_* v_{k-1} - v_{k-1}). \tag{24}$$

The first relation (Equation 22) involves the errors (ϵ_k), is based on Lemmas 13-15 (presented in Section 4) and is proved in Appendix C. The two other inequalities (the asymptotic performance of Approximate λ Policy Iteration with respect to the Bellman residuals in Equations 23 and 24) are somewhat simpler and are proved independently in Appendix D.

Remark 22 (Relation with the previous bounds of Munos (2007, 2003)) *We can look at the relation between our bound for general λ and the bounds derived separately by Munos for Approximate Value Iteraton (Lemma 2) and Approximate Policy Iteraton (Lemma 6):*

- *Let us first consider the case where $\lambda = 0$. Then $\beta = \gamma$, $A_k = P_k$ and*

$$B_{jk} = (1-\gamma)(I - \gamma P_k)^{-1} P_k P_{k-1} ... P_{j+1}.$$

Then our bound implies that $\limsup_{k\to\infty} \; v_* - v^{\pi_k}$ is upper bounded by:

$$\limsup_{k\to\infty} \; \sum_{j=k_0}^{k-1} \gamma^{k-j} \Big[(I - \gamma P_k)^{-1} P_k P_{k-1} ... P_{j+1} - \left(\gamma (I - \gamma P_k)^{-1} P_k P_{k-1} ... P_j + (P_*)^{k-j}\right) \Big] \epsilon_j. \tag{25}$$

The bound derived by Munos for Approximate Value Iteration (Lemma 2 page 6) is

$$\begin{aligned}
&\limsup_{k\to\infty} \; (I - \gamma P_k)^{-1} \sum_{j=0}^{k-1} \gamma^{k-j} \left[P_k P_{k-1} ... P_{j+1} - (P_*)^{k-j} \right] \epsilon_j \\
=&\limsup_{k\to\infty} \; \sum_{j=0}^{k-1} \gamma^{k-j} \left[(I - \gamma P_k)^{-1} P_k P_{k-1} ... P_{j+1} - (I - \gamma P_k)^{-1} (P_*)^{k-j} \right] \epsilon_j \\
=&\limsup_{k\to\infty} \; \sum_{j=0}^{k-1} \gamma^{k-j} \Big[(I - \gamma P_k)^{-1} P_k P_{k-1} ... P_{j+1} - \left((I - \gamma P_k)^{-1} \gamma P_k (P_*)^{k-j} + (P_*)^{k-j} \right) \Big] \epsilon_j.
\end{aligned} \tag{26}$$

The above bounds are very close to each other: we can go from Equation 25 to Equation 26 by replacing $P_{k-1} ... P_j$ by $(P_*)^{k-j}$.

- *Now, when* $\lambda = 1$, $\beta = 0$, $A_k = (1-\gamma)(I-\gamma P_k)^{-1}P_k$ *and*

$$B_{jk} = (1-\gamma)(P_*)^{k-1-j}P_{j+1}(I-\gamma P_{j+1})^{-1}.$$

Our bound is

$$\limsup_{k\to\infty} \; v_* - v^{\pi_k} \leq \limsup_{k\to\infty} \; \sum_{j=k_0}^{k-1} \gamma^{k-1-j}(P_*)^{k-1-j}u_j$$

with

$$u_j := \left[\gamma P_{j+1}(I-\gamma P_{j+1})^{-1}(I-\gamma P_j) - \gamma P_*\right]\epsilon_j.$$

By definition of the supremum limit, for all $\epsilon > 0$, *there exists an index* k_1 *such that for all* $j \geq k_1$,

$$u_j \leq \limsup_{l\to\infty} \; u_l + \epsilon\mathbf{e}.$$

Then:

$$\begin{aligned}\limsup_{k\to\infty} \; \sum_{j=k_1}^{k-1} \gamma^{k-1-j}(P_*)^{k-1-j}u_j &\leq \limsup_{k\to\infty} \; \sum_{j=k_1}^{k-1} \gamma^{k-1-j}(P_*)^{k-1-j}\left(\limsup_{l\to\infty} \; u_l + \epsilon\mathbf{e}\right) \\ &= (I-\gamma P_*)^{-1}\left(\limsup_{l\to\infty} \; u_l + \epsilon\mathbf{e}\right).\end{aligned}$$

As this is true for all $\epsilon > 0$, *we eventually find the bound of Munos for Approximate Policy Iteration (Lemma 6 page 9).*

Thus, up to some little details, our componentwise analysis unifies those of Munos. It is not a surprise that we fall back on the result of Munos for Approximate Policy Iteration because, as already mentionned in Remark 16, the proof we developed in Section 4 and Appendix A is a generalization of his. If we don't exactly recover the componentwise analysis of Munos for Approximate Value Iteration, this is not really fundamental as it will not affect most of the results we derive.

The componentwise bounds on the performance of Approximate λ Policy Iteration can be translated into span seminorm bounds, using the following Lemma (proved in Appendix F):

Lemma 23 *Let* x_k, y_k *be vectors and* X_{jk}, X'_{jk} *stochastic matrices satisfying*

$$\forall k_0, \quad \limsup_{k\to\infty} \; |x_k| \leq K \limsup_{k\to\infty} \; \sum_{j=k_0}^{k-1} \xi_{k-j}(X_{kj} - X'_{kj})y_j,$$

where $(\xi_i)_{i\geq 1}$ *is a sequence of non-negative weights satisfying:*

$$\sum_{i=1}^{\infty} \xi_i = K' < \infty,$$

then, for all distribution μ,

$$\mu_{kj} := \frac{1}{2}(X_{kj} + X'_{kj})^T \mu$$

are distributions and

$$\limsup_{k\to\infty} \|x_k\|_{p,\mu} \leq KK' \lim_{k_0\to\infty} \left[\sup_{k\geq j\geq k_0} \operatorname*{span}_{p,\mu_{kj}} [y_j] \right].$$

Thus, using this Lemma and the fact that $\sum_{i=1}^{\infty} \gamma^i = \frac{\gamma}{1-\gamma}$, Lemma 21 can be turned into the following proposition that unifies and generalizes Proposition 3 (page 6) on Approximate Value Iteration and Proposition 7 (page 9) on Approximate Policy Iteration.

Proposition 24 (Span Seminorm Performance of Approximate λPI (1/2)) *With the notations of Lemma 21, for all p, $k > j \geq 0$ and all distribution μ,*

$$\begin{aligned}
\mu_{kj} &:= \frac{1}{2}\left(B_{jk} + B'_{jk}\right)^T \mu, \\
\mu'_{kj} &:= \frac{1}{2}\left(C_k + C'_k\right)^T \mu, \text{ and} \\
\mu''_{kj} &:= \frac{1}{2}\left(D + D'_k\right)^T \mu
\end{aligned}$$

are distributions and the performance of the policies generated by λ Policy Iteration satisfies:

$$\begin{aligned}
\limsup_{k\to\infty} \|v_* - v^{\pi_k}\|_{p,\mu} &\leq \frac{\gamma}{(1-\gamma)^2} \lim_{k_0\to\infty} \left[\sup_{k\geq j\geq k_0} \operatorname*{span}_{p,\mu_{kj}} [\epsilon_j] \right], \\
\limsup_{k\to\infty} \|v_* - v^{\pi_k}\|_{p,\mu} &\leq \frac{\gamma}{(1-\gamma)^2} \limsup_{k\to\infty} \operatorname*{span}_{p,\mu'_{kj}} [\mathcal{T}_k v_k - v_k], \\
\forall k, \quad \|v_* - v^{\pi_k}\|_{p,\mu} &\leq \frac{\gamma}{1-\gamma} \operatorname*{span}_{p,\mu''_{kj}} [\mathcal{T}_* v_{k-1} - v_{k-1}].
\end{aligned}$$

As already mentionned, a drawback of the above L_p bounds comes from the fact that the distributions involved on the right hand sides are unknown. To go round this issue, one may consider the **concentration coefficient** assumption introduced by Munos (2003, 2007) and already mentioned in Equation 6 page 7. For clarity, we recall its definition here. We assume there exists a distribution ν and a real number $C(\nu)$ such that

$$C(\nu) := \max_{i,j,a} \frac{p_{ij}(a)}{\nu(j)}.$$

Then, we have the following property:

Lemma 25 *Let X be an average of products of stochastic matrices of the MDP. For any distribution μ, and vector y and any p,*

$$\operatorname*{span}_{p,X^T\mu} [y] \leq (C(\nu))^{1/p} \operatorname*{span}_{p,\nu} [y].$$

Proof It can be seen from the definition of the concentration coefficient $C(\nu)$ that $\mu^T X \leq C(\nu)\nu^T$. Thus,

$$\begin{aligned}
\left(\underset{p,X^T\mu}{\text{span}}\,[y]\right)^p &= \min_a \left(\|y - a\mathbf{e}\|_{p,X^T\mu}\right)^p \\
&= \min_a \mu^T X |y - a\mathbf{e}|^{\mathbf{p}} \\
&\leq C(\nu) \min_a \nu^T |y - a\mathbf{e}|^{\mathbf{p}} \\
&= C(\nu) \min_a \left(\|y - a\mathbf{e}\|_{p,\nu}\right)^p \\
&= C(\nu) \left(\underset{p,\nu}{\text{span}}\,[y]\right)^p .
\end{aligned}$$

■

Using this Lemma, and the fact that for any p, $\|x\|_\infty = \max_\mu \|x\|_{p,\mu}$, the L_p bounds of Proposition 24 become

Proposition 26 (Span Seminorm Performance of Approximate λPI (2/2)) *Let $C(\nu)$ be the concentration coefficient defined in Equation 6. For all p and all k,*

$$\begin{aligned}
\limsup_{k\to\infty} \|v_* - v^{\pi_k}\|_\infty &\leq \frac{\gamma}{(1-\gamma)^2} [C(\nu)]^{1/p} \limsup_{j\to\infty} \underset{p,\nu}{\text{span}}\,[\epsilon_j] , \\
\limsup_{k\to\infty} \|v_* - v^{\pi_k}\|_\infty &\leq \frac{\gamma}{(1-\gamma)^2} [C(\nu)]^{1/p} \limsup_{k\to\infty} \underset{p,\nu}{\text{span}}\,[\mathcal{T}_k v_k - v_k] , \\
\textit{and } \forall k, \quad \|v_* - v^{\pi_k}\|_\infty &\leq \frac{\gamma}{1-\gamma} [C(\nu)]^{1/p} \underset{p,\nu}{\text{span}}\,[\mathcal{T}_* v_{k-1} - v_{k-1}] .
\end{aligned}$$

This results generalizes and unifies those derived for Approximate Value Iteration (Proposition 4 page 7) and Approximate Policy Iteration (Proposition 8 page 9).

When comparing the specific bounds of Munos for Approximate Value Iteration (Propositions 3 and 4) and Approximate Policy Iteration (Propositions 7 and 8), we wrote that the latter had the nice property that the bounds only depend on *asymptotic* errors/residuals (while the former depends on all errors). Our bounds for λ Policy Iteration have this nice property too. Considering the relations between the span seminorms and the other standard norms (Equation 1 page 3), we see that our results are not only more general, but also slightly finer than those of Munos.

When the policy or the value converges The performance bounds with respect to the approximation error can be improved if we know or observe that the value or the policy converges. Note that the former condition implies the latter (while the opposite is not true: the policy may converge while the value still oscillates). Indeed, we have the following Corollary.

Corollary 27 (Performance of Approximate λPI in case of convergence) *If the value converges to some v, then the approximation error converges to some ϵ, and the corresponding greedy policy π satisfies*

$$\|v_* - v^\pi\|_\infty \leq \frac{\gamma}{1-\gamma} [C(\nu)]^{1/p} \underset{p,\nu}{\text{span}}\,[\epsilon] .$$

If the policy converges to some π, then

$$\|v_* - v^\pi\|_\infty \leq \frac{\gamma(1-\lambda\gamma)}{(1-\gamma)^2} [C(\nu)]^{1/p} \limsup_{j\to\infty} \operatorname*{span}_{p,\nu} [\epsilon_j] .$$

These bounds, proved in Appendix E, unify and extend those presented for Approximate Value Iteration (Corollary 5 page 7) and Approximate Policy Iteration (Corollary 9 page 10), in the similar situation where the policy or the value converges. It is interesting to notice that in the weaker situation where only the policy converges, the constant decreases from $\frac{1}{(1-\gamma)^2}$ to $\frac{1}{1-\gamma}$ when λ varies from 0 to 1; in other words, the closer to Policy Iteration, the better the bound in that situation.

7. Application of λ Policy Iteration to the Game of Tetris

In the final part of this paper, we consider (and describe for the sake of keeping this paper self-contained) exactly the same application (Tetris) and implementation as Bertsekas and Ioffe (1996). Our motivation for doing so is twofold:

- from a theoretical point of view, we show how our analysis (made in the discounted case $\gamma < 1$) can be adapted to an undiscounted problem (where $\gamma = 1$) like Tetris;
- we obtain empirical results that are different (and much less intriguing) than those of the original study. This gives us the opportunity to describe what we think are the reasons for such a difference.

But before doing so, we begin by describing the Tetris domain.

7.1 The Game of Tetris and its Model as an MDP

Tetris is a popular video game created in 1985 by Alexey Pajitnov. The game is played on a 10×20 grid where pieces of different shapes fall from the top (see Figure 4). The player has to choose where each piece is added: he can move it horizontally and rotate it. When a row is filled, it is removed and all cells above it move one row downwards. The goal is to remove as many lines as possible before the game is over, that is when there is not enough space remaining on the top of the pile to put the current new piece.

Instead of mimicking the original game (precisely described by Fahey (2003)), Bertsekas and Ioffe (1996) have focused on the main problem, that is choosing where to drop each coming piece. The corresponding MDP model is straightforward: the state consists of the wall configuration and the shape of the current falling piece. An action is the horizontal translation and the rotation which are applied to the piece before it is dropped on the wall. The reward is the number of lines which are removed after we have dropped the piece. As one considers the maximization of the score (the total number of lines removed during a game), the natural choice for the discount factor is $\gamma = 1$.

In a bit more details, the dynamics of Tetris is made of two components: the place where one drops the current piece and the choice of a new piece. As the latter component is uncontrollable (a new piece is chosen with uniform probability), the value functions needs not to be computed for all wall-piece pairs configurations but only for all wall configurations

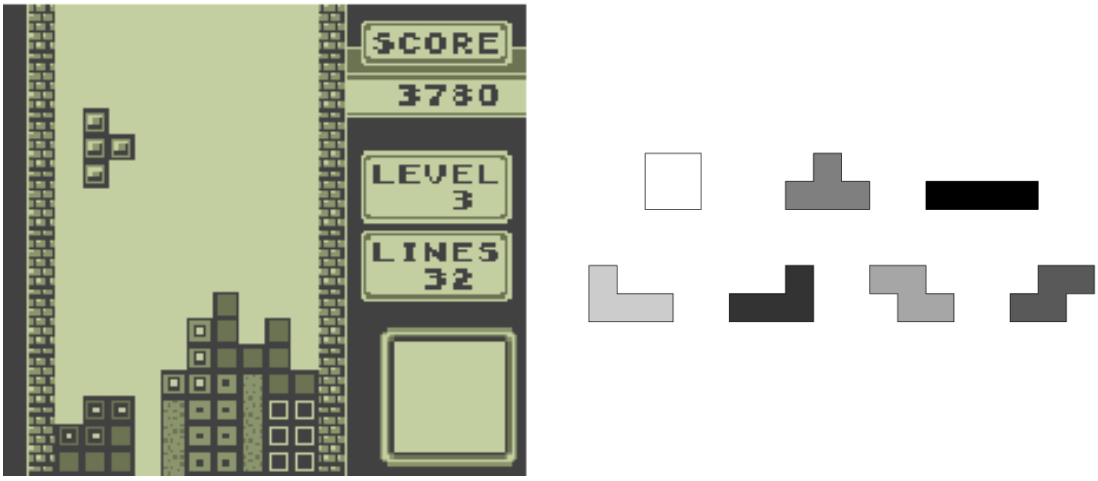

Figure 4: **Left:** a screenshot of a Tetris game. **Right:** the seven existing shapes.

(see for instance (Bertsekas and Ioffe, 1996)). Also considering that the first component of the dynamics is deterministic, the optimal value function satisfies a reduced form of the Bellman Equation

$$\forall s \in S, v_*(s) = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \max_{a \in A(p)} r(s,p,a) + v_*(succ(s,p,a)) \tag{27}$$

where S is the set of wall configurations, $\mathcal{P}$ is the set of pieces, $A(p)$ is the set of translation-rotation pairs that can be applied to a piece p, $r(s,p,a)$ and $succ(s,p,a)$ are respectively the number of lines removed and the (deterministic) next wall configuration if one puts a piece p on the wall s in translation-orientation a. The only function that satisfies the above Equation gives, for each wall configuration s, the average best score that can be achieved from s. If we know this function, a one step look-ahead strategy (that is a greedy policy) performs optimally.

Extension of the analysis for the undiscounted optimal control problem Tetris As just explained, the Tetris domain has a discount factor γ equal to 1, which makes it an *undiscounted MDP*. If this prevents us from applying directly most of the analysis we have made so far (since most of our bounds have a $(1-\gamma)$ term on the denominator), we briefly show in what follows how to adapt the analysis so that we recover a significant error analysis.

In undiscounted infinite horizon control problems, it is generally assumed that there exists a $N+1^{th}$ termination absorbing state 0. Once the system reaches this state, it remains there forever with no further reward, that is formally:

$$\forall a, p_{00}(a) = 1 \text{ and } r(0,a,0) = 0.$$

In the case of Tetris, the termination state corresponds to "game over", and the situation is particulary simple since Burgiel (1997) has shown that, whatever the strategy, some sequence of pieces (which necessarily occurs in finite time with probability 1) leads to

game-over whatever the decisions taken[15]. This implies in particular that there exists an integer $n_0 \leq N$ and a real number $\alpha < 1$ such that for all initial distributions μ, and actions $a_0, a_1, ..., a_{n_0-1}$,

$$P\left[i_{n_0} \neq 0 | i_0 \sim \mu, a_0, ..., a_{n_0-1}\right] \leq \alpha. \tag{28}$$

We can define the MDP model for Tetris only on the N non-terminal states, that is on $\{1, ...N\}$. In this situation, for any policy π, the matrix P_π is in general a *substochastic* matrix (a matrix of which the components are non-negative and for which the max norm is smaller than or equal to 1), and the above assumption means that for all set of n_0 policies $\pi_1, \pi_2, \cdots, \pi_{n_0}$,

$$\left\| P_{\pi_1} P_{\pi_2} \cdots P_{\pi_{n_0}} \right\|_\infty \leq \alpha.$$

The componentwise analysis of λ Policy Iteration is here identical to what we have done before, except that we have[16] $\gamma = 1$ and $\beta = 1$. The matrix A_k that appeared recurrently in our analysis has the following special form:

$$A_k := (1-\lambda)(I - \lambda P_k)^{-1} P_k.$$

and is a substochastic matrix. The first bound of the componentwise analysis of λ Policy Iteration (Lemma 21 page 21) can be shown to be generalized as follows (see Appendix G for details):

Lemma 28 (Componentwise Bounds in the Undiscounted Case)
Assume that there exists n_0 and α such that Equation 28 holds. Write $\eta := \frac{1-\lambda^{n_0}}{1-\lambda^{n_0}\alpha}$. For all i, write

$$\delta_i := \alpha^{\left\lfloor \frac{i}{n_0} \right\rfloor} \left[\left(\frac{1-\lambda^{n_0}}{1-\lambda} \right) \left(\frac{\lambda}{1-\lambda^{n_0}\alpha} \right) \left(\frac{1-\eta^i}{1-\eta} \right) + \frac{n_0 \eta^i}{1-\alpha} \right].$$

For all $j < k$, the following matrices

$$\begin{aligned} G_{jk} &:= \frac{1}{\delta_{k-j}} \left[\frac{\lambda}{1-\lambda} \sum_{i=j}^{k-1} (P_*)^{k-1-i} A_{i+1} A_i ... A_{j+1} + (I - P_k)^{-1} A_k A_{k-1} ... A_{j+1} \right] \\ \text{and } G'_{jk} &:= \frac{1}{\delta_{k-j}} G_{jk} P_j \end{aligned}$$

are substochastic and the performance of the policies generated by λ Policy Iteration satisfies

$$\forall k_0, \quad \limsup_{k\to\infty} v_* - v^{\pi_k} \leq \limsup_{k\to\infty} \sum_{j=k_0}^{k-1} \delta_{k-j} \left[G_{jk} - G'_{jk} \right] \epsilon_j. \tag{29}$$

15. In the literature, a stationary policy that reaches the terminal state in finite time with probability 1 is said to be *proper*. The usual assumptions in undiscounted infinite horizon control problems are: *(i)* there exists at least one proper policy and *(ii)* for every improper policy π, the corresponding value equals $-\infty$ for at least one state. A simpler situation is when all stationary policies are proper. The situation of Tetris is even simpler: all *non necessarily stationary* policies are proper.

16. For simplicity in our discussion, we consider $\lambda < 1$ to avoid the special case $\lambda = 1$ for which $\beta = 0$ (see Equation 14). The interested reader may however check that the results that we state are continuous in the neighborhood of $\lambda = 1$.

Remark 29 *By observing that $\eta \in (0,1)$, and that for all $x \in (0,1)$, $0 \leq \frac{1-x^{n_0}}{1-x} \leq n_0$, it can be seen that the coefficients δ_i are finite for all i. Furthermore, when $n_0 = 1$ (which matches the discounted case with $\alpha = \gamma$), one can observe that $\delta_i = \frac{\gamma^i}{1-\gamma}$ and that one recovers the result of Lemma 21.*

This lemma can then be exploited to show that λ Policy Iteration enjoys an L_p norm guarantee. Indeed, an analogue of Proposition 24 (whose proof is detailed in Appendix G) is the following proposition.

Proposition 30 (L_p norm Bound for in the Undiscounted Case) *Let $C(\nu)$ be the concentration coefficient defined in Equation 6. Let the notations of Lemma 28 hold. For all distribution μ on $(1, \cdots, N)$ and $k > j \geq 0$,*

$$\mu_{kj} := \frac{1}{2}\left(G_{jk} + G'_{jk}\right)^T \mu$$

are non-negative vectors and

$$\tilde{\mu}_{kj} := \frac{\mu_{kj}}{\|\mu_{kj}\|_1}$$

are distributions on $(1, \cdots, N)$. Then for all p, the loss of the policies generated by λ Policy Iteration satisfies

$$\limsup_{k\to\infty} \|v_* - v^{\pi_k}\|_{p,\mu} \leq K(\lambda, n_0)(C(\nu))^{1/p} \lim_{j\to\infty} \|\epsilon_j\|_{p,\tilde{\mu}_{kj}}$$

where

$$K(\lambda, n_0) := \lambda f(\lambda)\frac{f(1) - f(\eta)}{1-\eta} + f(1)f(\eta) - f(1),$$
$$\forall x < 1,\ f(x) := \frac{(1-x^{n_0})}{(1-x)(1-x^{n_0}\alpha)},\ \text{and by continuity } f(1) := \frac{n_0}{1-\alpha}.$$

Remark 31 *There are three differences with respect to the results we have presented for the discounted case.*

1. *The fact that we defined the model (and thus the algorithm) only on the non-terminal states $(1, \cdots, N)$ implies that there is no error incurred in the terminal state 0. Note, however, that this is not a strong assumption since the value of the terminal state is necessarily 0.*

2. *The right hand side depends on the L_p norm, and not the span L_p seminorm. This is due to the fact that the matrices G_{jk} and G'_{jk} defined above are in general* substochastic *matrices (and* not *stochastic matrices).*

3. *Eventually, the constant $K(\lambda, n_0)$ depends on λ. More precisely, it can be observed that:*
$$\lim_{\lambda\to 0} K(\lambda, n_0) = \lim_{\lambda\to 1} K(\lambda, n_0) = \frac{{n_0}^2}{(1-\alpha)^2} - \frac{n_0}{1-\alpha}$$
and that this is the minimal value of $\lambda \mapsto K(\lambda, n_0)$. Though we took particular care in deriving this bound, we leave for future work the question whether one could prove a

similar result with the constant $\frac{{n_0}^2}{(1-\alpha)^2} - \frac{n_0}{1-\alpha}$ *for any* $\lambda \in (0,1)$*. When* $n_0 = 1$ *(which matches the discounted case with* $\alpha = \gamma$*),* $K(\lambda, 1)$ *does not depend anymore on* λ *and we recover (without surprise) the bound of Proposition 24:*

$$\forall \lambda, K(\lambda, 1) = \frac{\alpha}{(1-\alpha)^2}.$$

Now that we are reassured about the fact that applying λ Policy Iteration approximately to Tetris is principled, we turn to the precise description of its actual implementation.

7.2 An Instance of Approximate λ Policy Iteration

For large scale problems, many Approximate Dynamic Programming algorithms are based on two complementary tricks:

- one uses samples to approximate the expectations such as that of Equation 13;
- one only looks for a linear approximation of the optimal value function:

$$v^\theta(s) = \theta(0) + \sum_{k=1}^{K} \theta(k)\Phi_k(s)$$

where $\theta = (\theta(0) \dots \theta(K))$ is the parameter vector and $\Phi_k(s)$ are some predefined feature functions on the state space. Thus, each value of θ characterizes a value function v^θ over the entire state space.

The instance of Approximate λ Policy Iteration of Bertsekas and Ioffe (1996) follows these ideas. More specifically, this algorithm is devoted to MDPs which have a termination state, that has 0 reward and is absorbing. For this algorithm to be run, one must further assume that all policies are proper, which means that all policies reach the termination state with probability one in finite time[17].

Similarly to Exact λ Policy Iteration, this Approximate λ Policy Iteration maintains a *compact* value-policy pair (θ_t, π_t). Given θ_t, π_{t+1} is the greedy policy with respect to v^{θ_t}, and can easily be computed exactly in any given state as the argmax in Equation 27. This policy π_{t+1} is used to simulate a batch of M trajectories: for each trajectory m, $(s_{m,0}, s_{m,1}, \dots, s_{m,N_m-1}, s_{m,N_m})$ denotes the sequence of states of the m^{th} trajectory, with s_{m,N_m} being the termination state. Then for approximating Equation 13, a reasonable

17. Bertsekas and Ioffe (1996) consider a weaker assumption for Exact λ Policy Iteration and its analysis, namely that there exists at least *one* propoer policy. However, this assumption is not sufficient for their Approximate algorithm, because this builds sample trajectories that need to reach a termination state. If the terminal state were not reachable in finite time, this algorithm may not terminate in finite time.

choice for θ_{t+1} is one that satisfies:

$$\begin{aligned}
v^{\theta_{t+1}}(s_{m,N_m}) &\simeq 0 \qquad (30)\\
v^{\theta_{t+1}}(s_{m,N_m-1}) &\simeq v^{\theta_t}(s_{m,N_m-1}) + \delta_t(s_{m,N_m-1}, s_{m,N_m})\\
v^{\theta_{t+1}}(s_{m,N_m-2}) &\simeq v^{\theta_t}(s_{m,N_m-2}) + \delta_t(s_{m,N_m-2}, s_{m,N_m-1}) + \gamma\lambda\delta_t(s_{m,N_m-1}, s_{m,N_m})\\
&\vdots \qquad \vdots\\
v^{\theta_{t+1}}(s_{m,k}) &\simeq v^{\theta_t}(s_{m,k}) + \sum_{s=k}^{N_{m-1}} (\gamma\lambda)^{s-k}\delta_t(s_{m,s}, s_{m,s+1})\\
&\vdots \qquad \vdots\\
v^{\theta_{t+1}}(s_{m,0}) &\simeq v^{\theta_t}(s_{m,0}) + \sum_{s=0}^{N_{m-1}} (\gamma\lambda)^{s}\delta_t(s_{m,s}, s_{m,s+1})
\end{aligned}$$

for all trajectories m, where

$$\delta_t(s_{m,N_m-1}, s_{m,N_m}) = r(s_{m,N_m-1}, \pi_{t+1}(s_{m,N_m-1}), s_{m,N_m}) - v^{\theta_t}(s_{m,N_m-1}) \qquad (31)$$

and for all $s < N_m - 1$

$$\delta_t(s_{m,s}, s_{m,s+1}) = r(s_{m,s}, \pi_{t+1}(s_{m,s}), s_{m,s+1}) + \gamma v^{\theta_t}(s_{m,s+1}) - v^{\theta_t}(s_{m,s})$$

are the temporal differences. Note that Equations 30 and 31 correspond to the terminal states for which there is no subsequent rewards. A standard and efficient solution to this problem consists in minimizing the least squares error, that is to choose θ_{t+1} as follows:

$$\theta_{t+1} = \arg\min_{\theta} \sum_{m=1}^{M}\sum_{k=0}^{N_m} \left(v^{\theta}(s_{m,k}) - v^{\theta_t}(s_{m,k}) - \sum_{j=k}^{N_{m-1}} (\gamma\lambda)^{j-k}\delta_t(s_{m,j}, s_{m,j+1}) \right)^2 .$$

This approximate version of λ Policy Iteration generalizes well-known algorithms. When $\lambda = 0$, the generic term becomes a sample of $[\mathcal{T}^{\pi_{k+1}} v](s_{m,k})$:

$$\begin{aligned}
v^{\theta_{t+1}}(s_{m,k}) &\simeq v^{\theta_t}(s_{m,k}) + \delta_t(s_{m,k}, s_{m,k+1})\\
&= r(s_{m,k}, \pi_{t+1}(s_{m,k}), s_{m,k+1}) + \gamma v^{\theta_t}(s_{m,k+1}). \qquad (32)
\end{aligned}$$

When $\lambda = 1$, the generic term becomes the sampled discounted return from $s_{m,k}$ until the end of the trajectory:

$$\begin{aligned}
v^{\theta_{t+1}}(s_{m,k}) &\simeq v^{\theta_t}(s_{m,k}) + \sum_{s=k}^{N_{m-1}} \gamma^{s-k}\delta_t(s_{m,s}, s_{m,s+1})\\
&= \sum_{s=k}^{N_{m-1}} \gamma^{s-k} r(s_{m,k}, \pi_{t+1}(s_{m,k}), s_{m,k+1}). \qquad (33)
\end{aligned}$$

In other words, for these limit values of λ, the algorithms correspond to approximate versions of Value Iteration and Policy Iteration as described by Bertsekas and Tsitsiklis (1996).

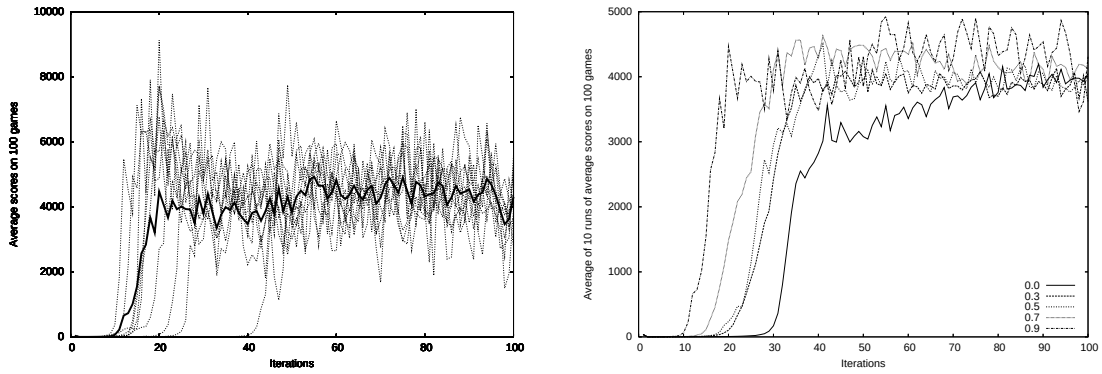

Figure 5: Average Score versus the number of iterations **Left**: 10 runs of λPI with $\lambda = 0.9$. Each point of each run is the average score computed with $M = 100$ games. The dark curve is a pointwise average of the 10 runs. **Right**: Pointwise average of 10 runs of λPI for different values of λ; the curve which appears to be the best ($\lambda = 0.9$) is the same as the bold curve of the left graph.

Also, as explained by Bertsekas and Ioffe (1996) and already mentioned in the introduction, the TD(λ) algorithm with linear features described by Sutton and Barto (1998, chapter 8.2) matches the algorithm we have just described when the above fitting problem is *approximated* using gradient iterations after each sample.

We follow the same protocol as originally proposed by Bertsekas and Ioffe (1996). Let $w = 10$ be the width of the board. We consider approximating the value function as a linear combination of $2w + 1 = 21$ feature functions:

$$V^\theta(s) = \theta(0) + \sum_{k=1}^{w} \theta(k) h_k + \sum_{k=1}^{w-1} \theta(k+w) \Delta h_k + \theta(2w) H + \theta(2w+1) L$$

where:

- for all $k \in \{1, 2, \cdots, w\}$, h_k is the *height* of the k^{th} column of the wall;
- for all $k \in \{1, 2, \cdots, w-1\}$, Δh_k is the *height difference* $|h_k - h_{k+1}|$ between columns k and $k+1$;
- H is the *maximum wall height* $\max_k h_k$;
- L is the number of *holes* (the number of empty cells covered by at least one full cell).

We started our experiments with the initial following vector: $r(2w) = -10$, $r(2w+1) = -1$ and $r(k) = 0$ for all $k < 2w$, so that the initial greedy policy scores in the low tens (Bertsekas and Ioffe (1996)). We used $M = 100$ training games for each policy update. As λPI is a stochastic algorithm, we ran each experiment 10 times. Figure 5 displays the learning curves. The left graph shows the 10 runs of λPI (each point is the average score computed with the $M = 100$ games) and the corresponding pointwise average for a single

value of λ, while the right graph shows such pointwise average curves for different values of λ: 0.0, 0.3, 0.5, 0.7 and 0.9. We chose to display on the left graph the runs corresponding to the value of $\lambda = 0.9$ that seemed to be the best on the right graph.

We can make the following observations.

- Though we initialized with not so bad a policy (the first value is around 30), the performance first drops to 0 and it *really* starts improving after a few iterations (typically arount ten). This is due to the fact that the initial value function is really bad: with the given parameters, the initial value is everywhere negative although it is clear that the optimal value function (the average best score) is everywhere positive. Further experiments showed that the overall behaviour of the algorithm was not affected by the weight initialization.

- The rise of performance globally happens sooner for larger values of λ, that is for values that makes the algorithm closer to Policy Iteration. This is not surprising as it complies with the fact that λ modulates the speed at which the value estimate tracks the real value of the current policy. However, the performance did not rise for $\lambda = 1$ (when it is equivalent to Approximate Policy Iteration), and this is probably due to the fact that the variance of the value update is too high.

- Quantitatively, the scores reach an overall level of 4000 lines per games for a big range of values of λ.

The empirical results we have just described qualitatively and quantitatively differ from the ones that were originally published in Bertsekas and Ioffe (1996), even though it is the exact same experimental setup. About their results, the authors wrote: "*An interesting and somewhat paradoxical observation is that a high performance is achieved after relatively few policy iterations, but the performance gradually drops significantly. We have no explanation for this intriguing phenomenon, which occurred with all of the successful methods that we tried*". As we explain now, we believe that the "intriguing" character of the results of Bertsekas and Ioffe (1996) might be related to a subtle implementation difference. Indeed, we can reproduce learning curves that are similar to those of Bertsekas and Ioffe (1996) with a little modification in our implementation of λPI, that removes the special treatments for the terminal states done through Equations 30 and 31. More precisely, if we replace them by the following Equations:

$$v^{\theta_{t+1}}(s_{m,N_m}) \simeq v^{\theta_t}(s_{m,N_m}) \tag{34}$$

$$\delta_t(s_{m,N_m-1}, s_{m,N_m}) = r(s_{m,N_m-1}, \pi_{t+1}(s_{m,N_m-1})) + \gamma v^{\theta_t}(s_{m,N_m}) - v^{\theta_t}(s_{m,N_m-1}) \tag{35}$$

that is if we replace the terminal value 0 by the value $V^{\theta_t}(s_{m,N_m})$ which is computed through the features of the terminal wall configuration s_{m,N_m}, then we get the performance shown in Figure 6. This figure shows the performance with respect to the iterations. We observe that the performance evolution qualitatively matches the performance curves published in Bertsekas and Ioffe (1996) and illustrates the above quotation describing the "intriguing phenomenon"[18].

18. The watchful reader may have noticed that the performance that we obtain is about twice that of Bertsekas and Ioffe (1996). A close inspection of the Tetris domain description in (Bertsekas and Ioffe,

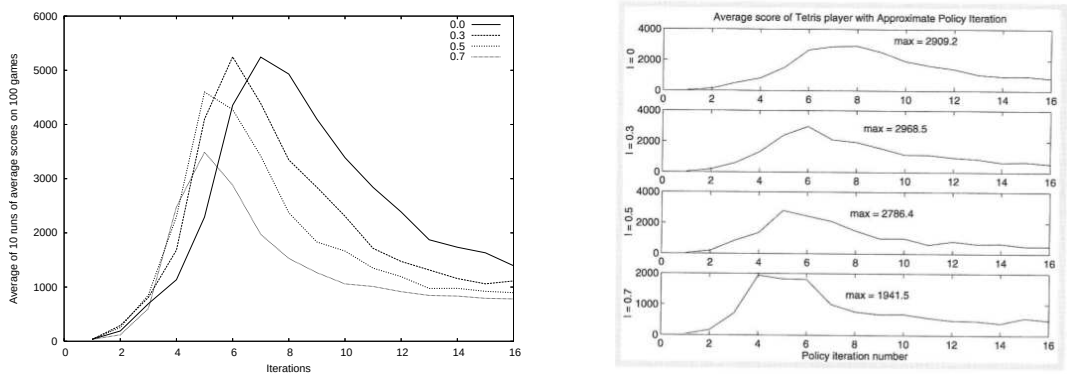

Figure 6: **Left**: Average score versus the number of iterations of λPI, *modified* so that it resembles the results of Bertsekas and Ioffe (1996) (see text for details). **Right**: A scan of the learning curves obtained by Bertsekas and Ioffe (1996).

In such a modified form, the approximate λPI algorithm makes much less sense: in particular, it is not true anymore that it reduces to approximate Value Iteration and approximate Policy Iteration when $\lambda = 0$ and $\lambda = 1$ respectively: Equations 34 and 35 induce a bias so that we cannot recover the identities of Equations 32 and 33. A closer examination of these experiments showed that the weights (θ_k) were diverging. This is not a surprise, since the use of Equations 34 and 35 violates the condition expressed in Remark 31-1 that there should be no error in the terminal state.

8. Conclusion and Future Work

We have considered the λ Policy Iteration algorithm introduced by Bertsekas and Ioffe (1996) that generalizes the standard algorithms Value Iteration and Policy Iteration. We have reviewed some results by Puterman (1994), Bertsekas and Tsitsiklis (1996) and Munos (2003, 2007), concerning the rate of convergence and some performance bounds of these standard algorithms in the exact and approximate cases. We have extended these results to λ Policy Iteration and derived convergence rates and performance bound for this algorithm, some of which are to our knowledge even new in the cases where λ Policy Iteration reduces to Value Iteration and Policy Iteration, notably the bound 21 (page 20). Not only does our analysis generalize previous results, but it improves them in two ways:

- as suggested by the results of Puterman (1994) in the exact case, the use of the span seminorm has enabled us to derive tighter bounds;

1996) shows that the authors consider the game of Tetris on a 10×19 board instead of our 10×20 setting, and as argued in a recent review on Tetris (Thiéry and Scherrer, 2009), this small difference is sufficient for explaining such a big performance difference.

- our analysis of Approximate λ Policy Iteration relates the asymptotic performance of the algorithm to the *asymptotic* errors/residuals instead of a uniform bound of the errors/residuals and this might be of practical interest[19].

More generally, we believe that an important contribution of this paper is of conceptual nature: we provide a unified vision of some of the main Approximate Dynamic Programming algorithms and their analyses; in particular, we hope that the new proof technique that is detailed in the appendices — especially the different objects that are defined in our proof overview in Section 4.2 — shall be useful for further studies.

As we mentionned earlier, Munos (2007) introduced some concentration coefficients that are finer than the one we used throughout the paper. In the same spirit, Farahmand *et al.* (2010) recently revisited the error propagation of Munos (2007, 2003) and improved (among other things) the constant in the bound related to these concentration coefficients. A natural track would be to adapt these refinements to the analysis of λ Policy Iteration. This does not look completely trivial since the componentwise analysis we derived for λ Policy Iteration is significanlty more intricate than the ones we find in the specific limit cases $\lambda = 0$ (Value Iteration) and $\lambda = 1$ (Policy Iteration).

Another potential direction would be to study the implications of the choice of the parameter λ, as for instance is done by Singh and Dayan (1998) for the value estimation problem. On this matter, the original analysis by Bertsekas and Ioffe (1996) shows how one can concretely implement Exact λ Policy Iteration. Each iteration requires the computation of the fixed point of the β-contracting operator M_k (see Equation 9 page 11). We plan to study the tradeoff between the ease for computing this fixed point (the smaller β the faster) and the time for λ Policy Iteration to converge to the optimal policy (the bigger β the faster). In parallel, the reader might have noticed that most of the bounds we have provided do not depend on λ. An interesting question is whether the finer concentration coefficients of Munos (2007) and Farahmand *et al.* (2010) we have just discussed may help keeping track of the influence of λ on the performance of the exact or approximate algorithm. In general, it would be interesting if we could tie a choice of λ to some intrisic characterics of the MDP, like for instance its smoothness.

Last but not least, we should insist on the fact that the implementation that we have described in Section 7.2, and which is borrowed from Bertsekas and Ioffe (1996), is just one possible instance of λ Policy Iteration. In the case of linear approximation architectures, Thiéry and Scherrer (2010) have proposed an implementation of λ Policy Iteration that is based on LSPI (Lagoudakis and Parr, 2003), in which the fixed point of M_k is approximated using LSTD(0) (Bradtke and Barto, 1996). Recently, Bertsekas (2011) proposed to compute this very fixed point with a variation of LSPE(λ') (Bertsekas and Ioffe, 1996; Nedić and Bertsekas, 2003) for some λ' potentially different from λ. Because of their very close structure, any existing implementation of Approximate Policy Iteration may probably be turned into some implementation of λ Policy Iteration. Proposing such implementations and assessing their relative merits constitutes interesting future research. This may in particular be done through some finite sample analysis, as had recently been done for Approximate

19. Recently and independently, Farahmand *et al.* (2010) derived bounds that have a similar flavour: they highlight the fact that the errors that have the more weight on the performance bounds are the latest.

Value Iteration and Policy Iteration implementations (Antos *et al.*, 2007, 2008; Munos and Szepesvári, 2008; Lazaric *et al.*, 2010).

References

Antos, A., Szepesvári, C., and Munos, R. (2007). Value-iteration based fitted policy iteration: Learning with a single trajectory. In *ADPRL 2007*, pages 330–337. IEEE.

Antos, A., Szepesvari, C., and Munos, R. (2008). Learning near-optimal policies with Bellman-residual minimization based fitted policy iteration and a single sample path. *Machine Learning Journal*, **71**, 89–129.

Bertsekas, D. (2011). Lambda Policy Iteration: A Review and A New Implementation. Technical Report LIDS-2874, MIT.

Bertsekas, D. and Ioffe, S. (1996). Temporal differences-based policy iteration and applications in neuro-dynamic programming. Technical Report LIDS-P-2349, MIT.

Bertsekas, D. and Tsitsiklis, J. (1996). *Neurodynamic Programming*. Athena Scientific.

Bradtke, S. J. and Barto, A. G. (1996). Linear Least-Squares algorithms for temporal difference learning. *Machine Learning*, **22**(1-3), 33–57.

Burgiel, H. (1997). How to Lose at Tetris. *Mathematical Gazette*, **81**, 194–200.

Fahey, C. P. (2003). Tetris AI, Computer plays Tetris. `http://colinfahey.com/tetris/tetris_en.html`.

Farahmand, A., Munos, R., and Szepesvári, C. (2010). Error Propagation for Approximate Policy and Value Iteration. In *NIPS*.

Gordon, G. (1995). Stable function approximation in dynamic programming. In A. Prieditis and S. Russell, editors, *ICML*, pages 261–268, San Francisco, CA. Morgan Kaufmann.

Lagoudakis, M. and Parr, R. (2003). Least-Squares Policy Iteration. *Journal of Machine Learning Research*, **4**, 1107–1149.

Lazaric, A., Ghavamzadeh, M., and Munos, R. (2010). Analysis of a Classification-based Policy Iteration Algorithm. In *ICML*, pages 607–614.

Munos, R. (2003). Error Bounds for Approximate Policy Iteration. In *ICML*, pages 560–567.

Munos, R. (2007). Performance bounds in Lp norm for approximate value iteration. *SIAM J. Control and Optimization*.

Munos, R. and Szepesvári, C. (2008). Finite-Time Bounds for Fitted Value Iteration. *Journal of Machine Learning Research*, **9**, 815–857.

Nedić, A. and Bertsekas, D. P. (2003). Least Squares Policy Evaluation Algorithms with Linear Function Approximation. *DEDS*, **13**, 79–110.

Puterman, M. (1994). *Markov Decision Processes*. Wiley, New York.

Puterman, M. and Shin, M. (1978). Modified policy iteration algorithms for discounted Markov decision problems. *Management Science*, **24**(11).

Singh, S. and Dayan, P. (1998). Analytical Mean Squared Error Curves for Temporal Difference Learning. *Machine Learning Journal*, **32**(1), 5–40.

Sutton, R. and Barto, A. (1998). *Reinforcement Learning, An introduction*. BradFord Book. The MIT Press.

Thiéry, C. and Scherrer, B. (2009). Improvements on Learning Tetris with Cross Entropy. *International Computer Games Association Journal*, **32**.

Thiéry, C. and Scherrer, B. (2010). Least-Squares λ Policy Iteration: Bias-Variance Trade-off in Control Problems. In *ICML*, Haifa, Israël.

Van Roy, B. (2006). Performance Loss Bounds for Approximate Value Iteration with State Aggregation. *Mathematics of Operation Research*, **31**(2), 234–244.

Williams, R. and Baird, L. (1993). Tight Performance Bounds on Greedy Policies Based on Imperfect Value Functions. Technical Report NU-CCS-93-14, Northeastern University.

Appendices

The following Appendices contains all the proofs concerning the analysis of λ Policy Iteration. We write $P_k = P^{\pi_k}$ the stochastic matrix corresponding to the policy π_k which is greedy with respect to v_{k-1}, P_* the stochastic matrix corresponding to the optimal policy π_*. Similarly we write $\mathcal{T}_k$ and $\mathcal{T}_*$ the associated Bellman operators.

The proof techniques we have developped are inspired by those of Munos in the articles (Munos, 2003, 2007). Most of the inequalities appear from the definition of the greedy operator:

$$\pi = \text{greedy}(v) \Leftrightarrow \forall \pi', \mathcal{T}^{\pi'} v \leq \mathcal{T}^{\pi} v.$$

We often use the property that an average of stochastic matrices is also a stochastic matrix. A recurrent instance of this property is: if P is some stochastic matrix, then the geometric average

$$(1-\alpha)\sum_{i=0}^{\infty}(\alpha P)^i = (1-\alpha)(I-\alpha P)^{-1}$$

with $0 \leq \alpha < 1$ is also a stochastic matrix. We use the property that if some vectors x and y are such that $x \leq y$, then $Px \leq Py$ for any stochastic matrix P. Eventually, we will

use the following equivalent forms of the operator $\mathcal{T}_\lambda^\pi$ (three of them were introduced in page 13): for any value v and any policy π, we have

$$\begin{aligned}
\mathcal{T}_\lambda^\pi v &:= v + (I - \lambda\gamma P^\pi)^{-1}(\mathcal{T}^\pi v - v) && (36)\\
&= (I - \lambda\gamma P^\pi)^{-1}(\mathcal{T}^\pi v - \lambda\gamma P^\pi v) && (37)\\
&= (I - \lambda\gamma P^\pi)^{-1}(r^\pi + (1-\lambda)\gamma P^\pi v) && (38)\\
&= (I - \lambda\gamma P^\pi)^{-1}(\lambda r^\pi + (1-\lambda)\mathcal{T}^\pi v). && (39)
\end{aligned}$$

Appendix A. Proofs of Lemmas 13-15 (core lemmas of the error propagation)

In this section, we prove the series of Lemmas that are at the heart of our analysis of the error propagation of λ Policy Iteration.

A.1 Proof of Lemma 13 (a relation between the shift and the Bellman residual)

Using the definition of $w_k = \mathcal{T}_\lambda^{\pi_k} v_{k-1}$ and the formulation of Equation 38, we can see that we have:

$$\begin{aligned}
(I - \gamma P_k)s_k &= (I - \gamma P_k)(w_k - v^{\pi_k})\\
&= (I - \gamma P_k)w_k - r_k\\
&= (I - \lambda\gamma P_k + \lambda\gamma P_k - \gamma P_k)w_k - r_k\\
&= (I - \lambda\gamma P_k)w_k + (\lambda\gamma P_k - \gamma P_k)w_k - r_k\\
&= r_k + (1-\lambda)\gamma P_k v_{k-1} + (\lambda - 1)\gamma P_k w_k - r_k\\
&= (1-\lambda)\gamma P_k(v_{k-1} - w_k)\\
&= (1-\lambda)\gamma P_k(I - \lambda\gamma P_k)^{-1}(v_{k-1} - \mathcal{T}_k v_{k-1})\\
&= (1-\lambda)\gamma P_k(I - \lambda\gamma P_k)^{-1}(-b_{k-1}).
\end{aligned}$$

Therefore

$$s_k = \beta(I - \gamma P_k)^{-1} A_k(-b_{k-1})$$

with

$$A_k := (1 - \lambda\gamma)P_k(I - \lambda\gamma P_k)^{-1}.$$

Suppose that we have a lower bound of the Bellman residual: $b_{k-1} \geq \underline{b_{k-1}}$ (we shall derive one soon). Since $(I - \gamma P_k)^{-1} A_k$ only has non-negative elements then

$$s_k \leq \beta(I - \gamma P_k)^{-1} A_k(-\underline{b_{k-1}}) := \overline{s_k}.$$

A.2 Proof of Lemma 14 (a lower bound of the Bellman residual)

From the definition of the algorithm, and using the fact that $\mathcal{T}_k v^{\pi_k} = v^{\pi_k}$, we see that:

$$\begin{aligned} b_k &= \mathcal{T}_{k+1} v_k - v_k \\ &= \mathcal{T}_{k+1} v_k - \mathcal{T}_k v_k + \mathcal{T}_k v_k - v_k \\ &\geq \mathcal{T}_k v_k - v_k \\ &= \mathcal{T}_k v_k - \mathcal{T}_k v^{\pi_k} + v^{\pi_k} - v_k \\ &= \gamma P_k (v_k - v^{\pi_k}) + v^{\pi_k} - v_k \\ &= (\gamma P_k - I)(s_k + \epsilon_k). \\ &= \beta A_k b_{k-1} + (\gamma P_k - I)\epsilon_k \end{aligned} \tag{40}$$

where we eventually used the relation between s_k and b_k (Lemma 13). In other words:

$$b_{k+1} \geq \beta A_{k+1} b_k + x_{k+1}$$

with

$$x_k := (\gamma P_k - I)\epsilon_k.$$

Since A_k is a stochastic matrix and $\beta \geq 0$, we get by induction:

$$b_k \geq \sum_{j=k_0+1}^{k} \beta^{k-j} \left(A_k A_{k-1} ... A_{j+1}\right) x_j + \beta^{k-k_0} \left(A_k A_{k-1} ... A_{k_0+1}\right) b_{k_0} := \underline{b_k}.$$

A.3 Proof of Lemma 15 (an upper bound of the distance)

Given that $\mathcal{T}_* v_* = v_*$, we have

$$\begin{aligned} v_* &= v_* + (I - \lambda\gamma P_{k+1})^{-1}(\mathcal{T}_* v_* - v_*) \\ &= (I - \lambda\gamma P_{k+1})^{-1}(\mathcal{T}_* v_* - \lambda\gamma P_{k+1} v_*). \end{aligned}$$

Using the definition of $w_{k+1} = \mathcal{T}_\lambda^{\pi_{k+1}} v_k$ and the formulation of Equation 37, one can see that the distance satisfies:

$$\begin{aligned} d_{k+1} &= v_* - w_{k+1} \\ &= (I - \lambda\gamma P_{k+1})^{-1}[(\mathcal{T}_* v_* - \lambda\gamma P_{k+1} v_*) - (\mathcal{T}_{k+1} v_k - \lambda\gamma P_{k+1} v_k)] \\ &= (I - \lambda\gamma P_{k+1})^{-1}[\mathcal{T}_* v_* - \mathcal{T}_{k+1} v_k + \lambda\gamma P_{k+1}(v_k - v_*)] \\ &= \lambda\gamma P_{k+1} d_{k+1} + \mathcal{T}_* v_* - \mathcal{T}_{k+1} v_k + \lambda\gamma P_{k+1}(v_k - v_*) \\ &= \lambda\gamma P_{k+1} d_{k+1} + \mathcal{T}_* v_* - \mathcal{T}_{k+1} v_k + \lambda\gamma P_{k+1}(w_k + \epsilon_k - v_*) \\ &= \lambda\gamma P_{k+1} d_{k+1} + \mathcal{T}_* v_* - \mathcal{T}_{k+1} v_k + \lambda\gamma P_{k+1}(\epsilon_k - d_k) \\ &= \mathcal{T}_* v_* - \mathcal{T}_{k+1} v_k + \lambda\gamma P_{k+1}(\epsilon_k + d_{k+1} - d_k). \end{aligned}$$

Since π^{k+1} is greedy with respect to v_k, we have $\mathcal{T}_{k+1} v_k \geq \mathcal{T}_* v_k$ and therefore:

$$\begin{aligned} \mathcal{T}_* v_* - \mathcal{T}_{k+1} v_k &= \mathcal{T}_* v_* - \mathcal{T}_* v_k + \mathcal{T}_* v_k - \mathcal{T}_{k+1} v_k \\ &\leq \mathcal{T}_* v_* - \mathcal{T}_* v_k \\ &= \gamma P_* (v_* - v_k) \\ &= \gamma P_* (v_* - (w_k + \epsilon_k)) \\ &= \gamma P_* d_k - \gamma P_* \epsilon_k. \end{aligned}$$

As a consequence, the distance satisfies:

$$d_{k+1} \leq \gamma P_* d_k + \lambda\gamma P_{k+1}(\epsilon_k + d_{k+1} - d_k) - \gamma P_* \epsilon_k.$$

Noticing that:

$$\begin{aligned}
\epsilon_k + d_{k+1} - d_k &= \epsilon_k + w_k - w_{k+1} \\
&= v_k - w_{k+1} \\
&= -(I - \lambda\gamma P_{k+1})^{-1}(\mathcal{T}_{k+1} v_k - v_k) \\
&= (I - \lambda\gamma P_{k+1})^{-1}(-b_k) \\
&\leq (I - \lambda\gamma P_{k+1})^{-1}(-\underline{b_k}),
\end{aligned}$$

we get:

$$d_{k+1} \leq \gamma P_* d_k + y_k$$

where

$$y_k := \frac{\lambda\gamma}{1-\lambda\gamma} A_{k+1}(-\underline{b_k}) - \gamma P_* \epsilon_k.$$

Since P_* is a stochastic matrix and $\gamma \geq 0$, we have by induction:

$$d_k \leq \sum_{j=k_0}^{k-1} \gamma^{k-1-j} (P_*)^{k-1-j} y_j + \gamma^{k-k_0} (P_*)^{k-k_0} d_{k_0} = \overline{d_k}.$$

Appendix B. Proofs of Lemma 17 (performance of Exact λ Policy Iteration)

We here derive the convergence rate bounds for Exact λ Policy Iteration (as expressed in Lemma 17 page 20). We rely on the loss bound analysis of Appendix A with $\epsilon_k = 0$. In this specific case, we know that the loss $l_k \leq \overline{d_k} + \overline{s_k}$ where

$$\begin{aligned}
-\underline{b_k} &= \beta^{k-k_0} A_k A_{k-1} ... A_{k_0+1}(-b_{k_0}), \\
\overline{d_k} &= \frac{\lambda\gamma}{1-\lambda\gamma} \sum_{j=k_0}^{k-1} \gamma^{k-1-j} (P_*)^{k-1-j} A_{j+1}(-\underline{b_j}) + \gamma^{k-k_0}(P_*)^{k-k_0} d_{k_0}, \\
\text{and } \overline{s_k} &= \beta(I - \gamma P_k)^{-1} A_k (-\underline{b_{k-1}}).
\end{aligned}$$

Introducing the following stochastic matrices:

$$\begin{aligned}
X_{i,j,k} &:= (P_*)^{k-1-i} A_{i+1} A_i ... A_{j+1} \\
\text{and } Y_{j,k} &:= (1-\gamma)(I - \gamma P_k)^{-1} A_k A_{k-1} ... A_{j+1},
\end{aligned}$$

we have

$$\overline{d_k} = \frac{\lambda\gamma}{1-\lambda\gamma} \sum_{j=k_0}^{k-1} \gamma^{k-1-j} \beta^{j-k_0} X_{j,k_0,k}(-b_{k_0}) + \gamma^{k-k_0}(P_*)^{k-k_0} d_{k_0}$$

and

$$\overline{s_k} = \frac{\beta^{k-k_0}}{1-\gamma} Y_{k_0,k}(-b_{k_0}).$$

Therefore the loss satisfies:

$$\begin{aligned} l_k &\leq \overline{d_k} + \overline{s_k} \\ &\leq \left(\frac{\gamma^{k-k_0}}{1-\gamma}\right) E'_{kk_0}(-b_{k_0}) + \gamma^{k-k_0}(P_*)^{k-k_0} d_{k_0} \end{aligned} \tag{41}$$

with

$$E'_{kk_0} := \left(\frac{1-\gamma}{\gamma^{k-k_0}}\right)\left(\frac{\lambda\gamma}{1-\lambda\gamma}\sum_{j=k_0}^{k-1}\gamma^{k-1-j}\beta^{j-k_0}X_{j,k_0,k} + \frac{\beta^{k-k_0}}{1-\gamma}Y_{k_0,k}\right).$$

To end the proof, we simply need to prove the following lemma:

Lemma 32 *E'_{kk_0} is a stochastic matrix.*

Proof It is clear from the definition of $X_{i,j,k}$ and $Y_{j,k}$ that normalizing E'_{kk_0} gives stochastic matrices. So we just need to check that their max norm is 1.

$$\begin{aligned} \|E'_{kk_0}\|_\infty &= \frac{1-\gamma}{\gamma^{k-k_0}}\left(\frac{\lambda\gamma}{1-\lambda\gamma}\sum_{j=k_0}^{k-1}\gamma^{k-1-j}\beta^{j-k_0} + \frac{\beta^{k-k_0}}{1-\gamma}\right) \\ &= \frac{1-\gamma}{\gamma^{k-k_0}}\left(\frac{\lambda\gamma}{1-\lambda\gamma}\frac{\gamma^{k-k_0}-\beta^{k-k_0}}{\gamma-\beta} + \frac{\beta^{k-k_0}}{1-\gamma}\right) \\ &= \frac{1-\gamma}{\gamma^{k-k_0}}\left(\frac{\gamma^{k-k_0}-\beta^{k-k_0}}{1-\gamma} + \frac{\beta^{k-k_0}}{1-\gamma}\right) \\ &= 1 \end{aligned}$$

where we used the facts that $\frac{\lambda\gamma}{\gamma-\beta} = \frac{1}{1-\beta}$ and $(1-\beta)(1-\lambda\gamma) = 1-\gamma$. ■

B.1 Proof of Equation 17 (a bound with respect to the Bellman residual)

We first need the following lemma:

Lemma 33 *The bias and the distance are related as follows:*

$$b_k \geq (I - \gamma P_*)d_k.$$

Proof Since π_{k+1} is greedy with respect to v_k, $\mathcal{T}_{k+1}v_k \geq \mathcal{T}_* v_k$ and

$$\begin{aligned} b_k &= \mathcal{T}_{k+1}v_k - v_k \\ &= \mathcal{T}_{k+1}v_k - \mathcal{T}_* v_k + \mathcal{T}_* v_k - \mathcal{T}_* v_* + v_* - v_k \\ &\geq \gamma P_*(v_k - v_*) + v_* - v_k \\ &= (I - \gamma P_*)d_k. \end{aligned}$$

■

We thus have:

$$d_{k_0} \leq (I - \gamma P_*)^{-1} b_{k_0}.$$

Then Equation 41 becomes

$$\begin{aligned} l_k &\leq \left[\gamma^{k-k_0}(P_*)^{k-k_0}(I-\gamma P_*)^{-1} - \left(\frac{\gamma^{k-k_0}}{1-\gamma}\right) E'_{kk_0}\right] b_{k_0} \\ &= \frac{\gamma^{k-k_0}}{1-\gamma}\left[E_{kk_0} - E'_{kk_0}\right] b_{k_0} \end{aligned}$$

where:

$$E_{kk_0} := (1-\gamma)(P_*)^{k-k_0}(I-\gamma P_*)^{-1}$$

is a stochastic matrix.

B.2 Proof of Equation 16 (a bound with respect to the distance)

From Lemma 33, we know that

$$-b_{k_0} \leq (I-\gamma P_*)(-d_{k_0}).$$

Then, Equation 41 becomes

$$\begin{aligned} l_k &\leq \left[\gamma^{k-k_0}(P_*)^{k-k_0} - \left(\frac{\gamma^{k-k_0}}{1-\gamma}\right) E'_{kk_0}(I-\gamma P_*)\right] d_{k_0} \\ &= \frac{\gamma^{k-k_0}}{1-\gamma}\left[F_{kk_0} - E'_{kk_0}\right] d_{k_0} \end{aligned}$$

where

$$F_{kk_0} := (1-\gamma)P_*^{k-k_0} + \gamma E'_{kk_0}P_*$$

is a stochastic matrix.

B.3 Proof of Equation 18 (a bound with respect to the distance and the loss of the greedy policy)

Define $\hat{v}_{k_0} := v_{k_0} - K\mathbf{e}$ where K is some constant. The following statements are equivalent:

$$\begin{aligned} \hat{b}_{k_0} &\geq 0 \\ \mathcal{T}_{k_0+1}\hat{v}_{k_0} &\geq \hat{v}_{k_0} \\ r_{k_0+1} + \gamma P_{k_0+1}(v_{k_0} - K\mathbf{e}) &\geq v_{k_0} - K\mathbf{e} \\ (I-\gamma P_{k_0+1})K\mathbf{e} &\geq -r_{k_0+1} + (I-\gamma P_{k_0+1})v_{k_0} \\ K\mathbf{e} &\geq (I-\gamma P_{k_0+1})^{-1}(-r_{k_0+1}) + v_{k_0} \\ K\mathbf{e} &\geq v_{k_0} - v^{\pi_{k_0+1}}. \end{aligned}$$

The minimal K for which $\hat{b}_{k_0} \geq 0$ is thus $K := \max_s[v_{k_0}(s) - v^{\pi_{k_0+1}}(s)]$. As $\hat{v}_{k_0}$ and v_{k_0} only differ by a constant vector, they generate the same sequence of policies $\pi_{k_0+1}, \pi_{k_0+2}...$ Then, as $\hat{b}_{k_0} \geq 0$, Equation 41 tells us that

$$\begin{aligned} v_* - v^{\pi_k} &\leq \gamma^{k-k_0}(P_*)^{k-k_0}(v_* - \hat{v}_{k_0}) \\ &= \gamma^{k-k_0}(P_*)^{k-k_0}(v_* - v_{k_0} + K\mathbf{e}). \end{aligned}$$

Now notice that

$$\begin{aligned} K &= \max_s [v_{k_0}(s) - v_*(s) + v_*(s) - v^{\pi_{k_0+1}}(s)] \\ &\leq \max_s [v_{k_0}(s) - v_*(s)] + \max_s [v_*(s) - v^{\pi_{k_0+1}}(s)] \\ &= -\min_s [v_*(s) - v_{k_0}(s)] + \|v_*(s) - v^{\pi_{k_0+1}}\|_\infty . \end{aligned}$$

Then, using the fact that $(P_*)^{k-k_0}\mathbf{e} = \mathbf{e}$, we get:

$$v_* - v^{\pi_k} \leq \gamma^{k-k_0} \left[(P_*)^{k-k_0} \left((v_* - v_{k_0}) - \min_s [v_*(s) - v_{k_0}(s)] e \right) + \|v_*(s) - v^{\pi_{k_0+1}}\|_\infty \mathbf{e} \right].$$

Appendix C. Proofs of Equation 22 in Lemma 21 (componentwise bounds on the error propagation)

We here use the loss bound analysis of Appendix A to derive an asymptotic analysis of approximate λ Policy Iteration with respect to the approximation error. The results stated here constitute a proof of the first inequality of Lemma 21 page 21.

C.1 Proof of Equation 22

Since the loss satisfies

$$l_k = d_k + s_k \leq \overline{d_k} + \overline{s_k}, \tag{42}$$

an upper bound of the loss can be derived from the upper bound of the distance and the shift.

Let us first concentrate on the bound $\overline{d_k}$ of the distance. Lemmas 14 and 15 imply that:

$$\begin{aligned} \overline{d_k} &= \sum_{i=k_0}^{k-1} \gamma^{k-1-i} (P_*)^{k-1-i} y_i + \mathcal{O}(\gamma^{k-k_0}), \\ y_i &= \frac{\lambda\gamma}{1-\lambda\gamma} A_{i+1}(-\underline{b_i}) - \gamma P_* \epsilon_i, \\ -\underline{b_i} &= \sum_{j=k_0}^{i} \beta^{i-j} \left(A_i A_{i-1} ... A_{j+1}\right)(-x_j) + \mathcal{O}(\beta^{i-k_0}), \\ \text{and} \quad -x_j &= (I - \gamma P_j)\epsilon_j. \end{aligned} \tag{43}$$

Writing

$$X_{i,j,k} := (P_*)^{k-1-i} A_{i+1} A_i ... A_{j+1}$$

and putting all things together, we see that:

$$\begin{aligned}
\overline{d_k} &= \frac{\lambda\gamma}{1-\lambda\gamma}\sum_{i=k_0}^{k-1}\gamma^{k-1-i}\left(\sum_{j=k_0}^{i}\beta^{i-j}X_{i,j,k}(I-\gamma P_j)\epsilon_j+\mathcal{O}(\beta^{i-k_0})\right) \\
&\qquad -\sum_{i=k_0}^{k-1}\gamma^{k-i}(P_*)^{k-i}\epsilon_i+\mathcal{O}(\gamma^{k-k_0}) \\
&= \frac{\lambda\gamma}{1-\lambda\gamma}\sum_{i=k_0}^{k-1}\sum_{j=k_0}^{i}\gamma^{k-1-i}\beta^{i-j}X_{i,j,k}(I-\gamma P_j)\epsilon_j-\sum_{i=k_0}^{k-1}\gamma^{k-i}(P_*)^{k-i}\epsilon_i+\mathcal{O}(\gamma^{k-k_0}) \\
&= \frac{\lambda\gamma}{1-\lambda\gamma}\sum_{j=k_0}^{k-1}\sum_{i=j}^{k-1}\gamma^{k-1-i}\beta^{i-j}X_{i,j,k}(I-\gamma P_j)\epsilon_j-\sum_{j=k_0}^{k-1}\gamma^{k-j}(P_*)^{k-j}\epsilon_j+\mathcal{O}(\gamma^{k-k_0}) \\
&= \sum_{j=k_0}^{k-1}\left[\left(\frac{\lambda\gamma}{1-\lambda\gamma}\sum_{i=j}^{k-1}\gamma^{k-1-i}\beta^{i-j}X_{i,j,k}(I-\gamma P_j)\right)-\gamma^{k-j}(P_*)^{k-j}\right]\epsilon_j+\mathcal{O}(\gamma^{k-k_0}) \quad (44)
\end{aligned}$$

where between the first two lines, we used the fact that:

$$\frac{\lambda\gamma}{1-\lambda\gamma}\sum_{i=k_0}^{k-1}\gamma^{k-1-i}\beta^{i-k_0}=\frac{\lambda\gamma}{1-\lambda\gamma}\frac{\gamma^{k-k_0}-\beta^{k-k_0}}{\gamma-\beta}=\frac{\gamma^{k-k_0}-\beta^{k-k_0}}{1-\gamma}=\mathcal{O}(\gamma^{k-k_0}) \quad (45)$$

using the identities $\lambda\gamma=\frac{\gamma-\beta}{1-\beta}$ and $1-\gamma\lambda=\frac{1-\gamma}{1-\beta}$.

Let us now consider the bound $\overline{s_k}$ of the shift. From Lemma 13 and the bound on b_k in Equation 43, we have

$$\begin{aligned}
\overline{s_k} &= \beta(I-\gamma P_k)^{-1}A_k(-\underline{b_{k-1}}) \\
&= \beta(I-\gamma P_k)^{-1}A_k\left[\left(\sum_{j=k_0}^{k-1}\beta^{k-1-j}\left(A_{k-1}A_{k-2}...A_{j+1}\right)(-x_j)\right)+\mathcal{O}(\gamma^{k-k_0})\right] \\
&= \sum_{j=k_0}^{k-1}\frac{\beta^{k-j}}{1-\gamma}Y_{j,k}(I-\gamma P_j)\epsilon_j+\mathcal{O}(\gamma^{k-k_0}) \quad (46)
\end{aligned}$$

with

$$Y_{j,k}:=(1-\gamma)(I-\gamma P_k)^{-1}A_kA_{k-1}...A_{j+1}.$$

Eventually, from Equations 42, 44 and 46 we get:

$$\begin{aligned}
l_k \leq \sum_{j=k_0}^{k-1}\left[\left(\frac{\lambda\gamma}{1-\lambda\gamma}\sum_{i=j}^{k-1}\gamma^{k-1-i}\beta^{i-j}X_{i,j,k}+\frac{\beta^{k-j}}{1-\gamma}Y_{j,k}\right)(I-\gamma P_j)-\gamma^{k-j}(P_*)^{k-j}\right]\epsilon_j \\
+\mathcal{O}(\gamma^{k-k_0}). \quad (47)
\end{aligned}$$

Introduce the following matrices:

$$\begin{aligned} B_{jk} &:= \frac{1-\gamma}{\gamma^{k-j}}\left[\frac{\lambda\gamma}{1-\lambda\gamma}\sum_{i=j}^{k-1}\gamma^{k-1-i}\beta^{i-j}X_{i,j,k}+\frac{\beta^{k-j}}{1-\gamma}Y_{j,k}\right] \\ B'_{jk} &:= \gamma B_{jk}P_j+(1-\gamma)(P_*)^{k-j}. \end{aligned}$$

Lemma 34 *B_{jk} and B'_{jk} are stochastic matrices.*

Proof It is clear from the definition of $X_{i,j,k}$ and $Y_{j,k}$ that normalizing B_{jk} and B'_{jk} gives stochastic matrices. So we just need to check that their max norm is 1.

$$\begin{aligned} \|B_{jk}\| &= \frac{(1-\gamma)}{\gamma^{k-j}}\left[\frac{\lambda\gamma}{1-\lambda\gamma}\sum_{i=j}^{k-1}\gamma^{k-1-i}\beta^{i-j}+\frac{\beta^{k-j}}{1-\gamma}\right] \\ &= \frac{(1-\gamma)}{\gamma^{k-j}}\left[\frac{\lambda\gamma}{1-\lambda\gamma}\frac{\gamma^{k-j}-\beta^{k-j}}{\gamma-\beta}+\frac{\beta^{k-j}}{1-\gamma}\right] \\ &= \frac{(1-\gamma)}{\gamma^{k-j}}\left[\frac{\gamma^{k-j}-\beta^{k-j}}{(1-\lambda\gamma)(1-\beta)}+\frac{\beta^{k-j}}{1-\gamma}\right] \\ &= \frac{(1-\gamma)}{\gamma^{k-j}}\left[\frac{\gamma^{k-j}-\beta^{k-j}}{1-\gamma}+\frac{\beta^{k-j}}{1-\gamma}\right] \\ &= 1. \end{aligned}$$

where we used the identities: $\lambda\gamma=\frac{\gamma-\beta}{1-\beta}$ and $(1-\beta)(1-\gamma\lambda)=1-\gamma$. Then it is also clear that $\|B'_{jk}\|=1$. ■

Thus, Equation 47 can be rewritten as follows:

$$\begin{aligned} l_k &\leq \sum_{j=k_0}^{k-1}\left[\frac{\gamma^{k-j}}{1-\gamma}B_{jk}(I-\gamma P_j)-\gamma^{k-j}(P_*)^{k-j}\right]\epsilon_j+\mathcal{O}(\gamma^{k-k_0}) \\ &= \frac{1}{1-\gamma}\sum_{j=k_0}^{k-1}\gamma^{k-j}\left[B_{jk}-B'_{jk}\right]\epsilon_j+\mathcal{O}(\gamma^{k-k_0}). \end{aligned}$$

Taking the supremum limit, we see that for all k_0,

$$\limsup_{k\to\infty} l_k \leq \frac{1}{1-\gamma}\limsup_{k\to\infty}\sum_{j=k_0}^{k-1}\gamma^{k-j}\left[B_{jk}-B'_{jk}\right]\epsilon_j. \tag{48}$$

Appendix D. Proofs of Equations 23-24 in Lemma 21 (componentwise bounds with respect to the Bellman residuals)

In this section, we study the loss

$$l_k := v_* - v^{\pi_k}$$

with respect to the two following **Bellman residuals**:

$$b'_k := \mathcal{T}_k v_k - v_k$$

$$\text{and} \quad b_k := \mathcal{T}_{k+1} v_k - v_k = \mathcal{T} v_k - v_k.$$

The term b'_k says how much v_k differs from the value of π_k while b_k says how much v_k differs from the value of the policies π_{k+1} and π_*. The results stated here prove the last two inequalities of Lemma 21 page 21.

D.1 Proof of Equation 23 (bounds with respect to the Policy Bellman residual)

Our analysis relies on the following lemma

Lemma 35 *Suppose that we have a policy π, a function v that is an approximation of the value v^π of π in the sense that its residual $b' := T^\pi v - v$ is small. Taking the greedy policy π' with respect to v reduces the loss as follows:*

$$v_* - v^{\pi'} \leq \gamma P_*(v_* - v^\pi) + \left(\gamma P_*(I - \gamma P)^{-1} - \gamma P'(I - \gamma P')^{-1}\right) b'$$

where P and P' are the stochastic matrices which correspond to π and π'.

Proof We have:

$$\begin{aligned} v_* - v^{\pi'} &= \mathcal{T}_* v_* - \mathcal{T}^{\pi'} v^{\pi'} \\ &= \mathcal{T}_* v_* - \mathcal{T}_* v^\pi + \mathcal{T}_* v^\pi - \mathcal{T}_* v + \mathcal{T}_* v - \mathcal{T}^{\pi'} v + \mathcal{T}^{\pi'} v - \mathcal{T}^{\pi'} v^{\pi'} \\ &\leq \gamma P_*(v_* - v^\pi) + \gamma P_*(v^\pi - v) + \gamma P'(v - v^{\pi'}) \end{aligned} \tag{49}$$

where we used the fact that $\mathcal{T}_* v \leq \mathcal{T}^{\pi'} v$. One can see that:

$$\begin{aligned} v^\pi - v &= \mathcal{T}^\pi v^\pi - v \\ &= \mathcal{T}^\pi v^\pi - \mathcal{T}^\pi v + \mathcal{T}^\pi v - v \\ &= \gamma P(v^\pi - v) + b' \\ &= (I - \gamma P)^{-1} b' \end{aligned} \tag{50}$$

and that

$$\begin{aligned} v - v^{\pi'} &= v - \mathcal{T}^{\pi'} v^{\pi'} \\ &= v - \mathcal{T}^\pi v + \mathcal{T}^\pi v - \mathcal{T}^{\pi'} v + \mathcal{T}^{\pi'} v - \mathcal{T}^{\pi'} v^{\pi'} \\ &\leq -b' + \gamma P'(v - v^{\pi'}) \\ &\leq (I - \gamma P')^{-1}(-b'). \end{aligned} \tag{51}$$

where we used the fact that $\mathcal{T}^\pi v \leq \mathcal{T}^{\pi'} v$. We get the result by putting back Equations 50 and 51 into Equation 49. ■

To derive a bound for λ Policy Iteration, we simply apply the above lemma to $\pi = \pi_k$, $v = v_k$ and $\pi' = \pi_{k+1}$. We thus get:

$$l_{k+1} \leq \gamma P_* l_k + \left(\gamma P_* (I - \gamma P_k)^{-1} - \gamma P_{k+1} (I - \gamma P_{k+1})^{-1}\right) b'_k.$$

Introduce the following stochastic matrices:

$$C_k := (1-\gamma)^2 (I - \gamma P_*)^{-1} \left(P_* (I - \gamma P_k)^{-1}\right),$$

$$C'_k := (1-\gamma)^2 (I - \gamma P_*)^{-1} \left(P_{k+1} (I - \gamma P_{k+1})^{-1}\right).$$

This leads to the following componentwise bound:

$$\limsup_{k\to\infty} \; l_k \leq \frac{\gamma}{(1-\gamma)^2} \limsup_{k\to\infty} \; \left[C_k - C'_k\right] b'_k.$$

D.2 Proof of Equation 24 (bounds with respect to the Bellman residual)

We rely on the following lemma (which is for instance proved by Munos (2007))

Lemma 36 *Suppose that we have a function v. Let π be the greedy policy with respect to v. Then*

$$v_* - v^\pi \leq \gamma \left[P_* (I - \gamma P_*)^{-1} - P^\pi (I - \gamma P^\pi)^{-1}\right] (\mathcal{T}^\pi v - v).$$

We provide a proof for the sake of completeness:

Proof Using the fact that $\mathcal{T}_* v \leq \mathcal{T}^\pi v$, we see that

$$\begin{aligned}
v_* - v^\pi &= \mathcal{T}_* v_* - \mathcal{T}^\pi v^\pi \\
&= \mathcal{T}_* v_* - \mathcal{T}_* v + \mathcal{T}_* v - \mathcal{T}^\pi v + \mathcal{T}^\pi v - \mathcal{T}^\pi v^\pi \\
&\leq \mathcal{T}_* v_* - \mathcal{T}_* v + \mathcal{T}^\pi v - \mathcal{T}^\pi v^\pi \\
&= \gamma P_* (v_* - v) + \gamma P^\pi (v - v^\pi) \\
&= \gamma P_* (v_* - v^\pi) + \gamma P_* (v^\pi - v) \gamma P^\pi (v - v^\pi) \\
&\leq (I - \gamma P_*)^{-1} (\gamma P_* - \gamma P^\pi)(v^\pi - v).
\end{aligned}$$

Using Equation 50 we see that:

$$v^\pi - v = (I - \gamma P^\pi)^{-1} (\mathcal{T}^\pi v - v).$$

Thus

$$\begin{aligned}
v_* - v^\pi &\leq (I - \gamma P_*)^{-1} (\gamma P_* - \gamma P^\pi)(I - \gamma P^\pi)^{-1} (\mathcal{T}^\pi v - v) \\
&= (I - \gamma P_*)^{-1} (\gamma P_* - I + I - \gamma P^\pi)(I - \gamma P^\pi)^{-1} (\mathcal{T}^\pi v - v) \\
&= \left[(I - \gamma P_*)^{-1} - (I - \gamma P^\pi)^{-1}\right] (\mathcal{T}^\pi v - v) \\
&= \gamma \left[P_* (I - \gamma P_*)^{-1} - P^\pi (I - \gamma P^\pi)^{-1}\right] (\mathcal{T}^\pi v - v).
\end{aligned}$$

■

To derive a bound for λ Policy Iteration, we simply apply the above lemma to $v = v_{k-1}$ and $\pi = \pi_k$. We thus get:

$$l_k \leq \frac{\gamma}{1-\gamma} \left[D - D'_k\right] b_{k-1} \tag{52}$$

where

$$\begin{aligned} D &:= (1-\gamma)P_*(I-\gamma P_*)^{-1} \\ \text{and } D'_k &:= (1-\gamma)P_k(I-\gamma P_k)^{-1} \end{aligned}$$

are stochastic matrices.

Appendix E. Proofs of Corollary 27

Ths section provides a proof of Corollary 27 page 25, in which we refine the bounds when the value or the policy converges.

E.1 Proof of the first inequality of Corollary 27 (when the value converges)

Suppose that λ Policy Iteration converges to some value v. Let policy π be the corresponding greedy policy, with stochastic matrix P. Let b be the Bellman residual of v. It is also clear that the approximation error also converges to some ϵ. Indeed from Algorithm 3 and Equation 10, we get:

$$b = \mathcal{T}v - v = (I - \lambda\gamma P)(-\epsilon).$$

From the bound with respect to the Bellman residual (Equation 52 page 47), we can see that:

$$\begin{aligned} v_* - v^\pi &\leq \left[(I-\gamma P_*)^{-1} - (I-\gamma P)^{-1}\right] b \\ &= \left[(I-\gamma P)^{-1} - (I-\gamma P_*)^{-1}\right] (I-\lambda\gamma P)\epsilon \\ &= \left[(I-\gamma P)^{-1}(I-\lambda\gamma P) - (I-\gamma P_*)^{-1}(I-\lambda\gamma P)\right] \epsilon \\ &= \left[(I-\gamma P)^{-1}(I-\gamma P+\gamma P-\lambda\gamma P) - (I-\gamma P_*)^{-1}(I-\lambda\gamma P)\right] \epsilon \\ &= \left[\left(I+(1-\lambda)(I-\gamma P)^{-1}\gamma P + \lambda(I-\gamma P_*)^{-1}\gamma P\right) - (I-\gamma P_*)^{-1}\right] \epsilon \\ &= \left[\left((1-\lambda)(I-\gamma P)^{-1}\gamma P + \lambda(I-\gamma P_*)^{-1}\gamma P\right) - (I-\gamma P_*)^{-1}\gamma P_*\right] \epsilon \\ &= \frac{\gamma}{1-\gamma}\left[B_v - D\right]\epsilon. \end{aligned}$$

where

$$\begin{aligned} B_v &:= (1-\gamma)\left((1-\lambda)(I-\gamma P)^{-1}P + \lambda(I-\gamma P_*)^{-1}P\right) \\ D &:= (1-\gamma)P_*(I-\gamma P_*)^{-1}. \end{aligned}$$

Lemma 37 *B_v and D are stochastic matrices.*

Proof It is clear that $\|D\| = 1$. Also:

$$\begin{aligned} \|B_v\| &= (1-\gamma)\left(1 + \frac{(1-\lambda)\gamma}{1-\gamma} + \frac{\lambda\gamma}{1-\gamma}\right) \\ &= (1-\gamma)\left(1 + \frac{\gamma}{1-\gamma}\right) \\ &= 1. \end{aligned}$$

Then, the first bound of Corollary 27 follows from the application of Lemmas 23 and 25. ■

E.2 Proof of the second inequality of Corollary 27 (when the policy converges)

Suppose that λ Policy Iteration converges to some policy π. Write P the corresponding stochastic matrix and

$$A^\pi := (1-\lambda\gamma)P(I-\lambda\gamma P)^{-1}.$$

Then for some big enough k_0, we have:

$$l_k \leq \sum_{j=k_0}^{k-1} \left[\frac{\gamma^{k-j}}{1-\gamma} A^\pi_{jk} A^\pi (I-\gamma P) - \gamma^{k-j}(P_*)^{k-j} \right] \epsilon_j + \mathcal{O}(\gamma^{k-k_0})$$

where

$$A^\pi_{jk} := \frac{1-\gamma}{\gamma^{k-j}} \left[\frac{\lambda\gamma}{1-\lambda\gamma} \sum_{i=j}^{k-1} \gamma^{k-1-i}\beta^{i-j}(P_*)^{k-1-i}(A^\pi)^{i-j} + \beta^{k-j}(I-\gamma P)^{-1}(A^\pi)^{k-1-j} \right]$$

is a stochastic matrix (for the same reasons why B_{jk} is a stochastic matrix in Lemma 34). Noticing that

$$\begin{aligned} A^\pi(I-\gamma P) &= (1-\lambda\gamma)P(I-\lambda\gamma P)^{-1}(I-\gamma P) \\ &= (1-\lambda\gamma)P(I-\lambda\gamma P)^{-1}(I-\lambda\gamma P+\lambda\gamma P-\gamma P) \\ &= (1-\lambda\gamma)P(I-(1-\lambda)(I-\lambda\gamma P)^{-1}\gamma P) \\ &= (1-\lambda\gamma)P-\gamma(1-\lambda)A^\pi P \end{aligned}$$

we can deduce that

$$\begin{aligned} l_k &\leq \sum_{j=k_0}^{k-1} \left[\frac{\gamma^{k-j}}{1-\gamma} A^\pi_{jk} \left[(1-\lambda\gamma)P - \gamma(1-\lambda)A^\pi P \right] - \gamma^{k-j}(P_*)^{k-j} \right] \epsilon_j + \mathcal{O}(\gamma^{k-k_0}) \\ &= \sum_{j=k_0}^{k-1} \gamma^{k-j} \left[\frac{1-\lambda\gamma}{1-\gamma} A^\pi_{jk} P - \left[\frac{\gamma(1-\lambda)}{1-\gamma} A^\pi_{jk} A^\pi P + (P_*)^{k-j} \right] \right] \epsilon_j + \mathcal{O}(\gamma^{k-k_0}) \\ &= \frac{1-\lambda\gamma}{1-\gamma} \sum_{j=k_0}^{k-1} \gamma^{k-j} \left[B^\pi_{jk} - B'^\pi_{jk} \right] \epsilon_j + \mathcal{O}(\gamma^{k-k_0}) \end{aligned} \tag{53}$$

where

$$\begin{aligned} B^\pi_{jk} &:= A^\pi_{jk} P \\ B'^\pi_{jk} &:= \frac{1-\gamma}{1-\lambda\gamma} \left[\frac{\gamma(1-\lambda)}{1-\gamma} A^\pi_{jk} A^\pi P + (P_*)^{k-j} \right]. \end{aligned}$$

Lemma 38 *B^π_{jk} and B'^π_{jk} are stochastic matrices.*

Proof It is clear that $\|B^\pi_{jk}\| = 1$. Also:

$$\begin{aligned} \|B'^\pi_{jk}\| &= \frac{1-\gamma}{1-\lambda\gamma} \left(1 + \frac{\gamma(1-\lambda)}{1-\gamma} \right) \\ &= \frac{1-\gamma}{1-\lambda\gamma} \frac{1-\gamma+\gamma-\lambda\gamma}{1-\gamma} \\ &= 1. \end{aligned}$$

Then, the second bound of Corollary 27 follows from the application of Lemmas 23 and 25. ■

Appendix F. Proofs of Lemmas 18 and 23 (from componentwise bounds to span seminorm bounds)

This section contains the proofs of Lemmas 18 and 23 that enable us to derive span seminorm performance bounds from the componentwise analysis developped in the previous sections. It is easy to see that Lemma 18 is a special case of Lemma 23, so we only prove the latter.

Consider the notations of Lemma 23. Write $a_{kj} := \arg\min_a \|y_j - a\mathbf{e}\|_{p,\mu_{kj}}$. As X_{jk} and X'_{jk} are stochastic matrices, $X_k\mathbf{e} = \mathbf{X'_k}\mathbf{e} = \mathbf{e}$ and we can write that:

$$\limsup_{k\to\infty} |x_k| \leq K \limsup_{k\to\infty} \sum_{j=k_0}^{k-1} \xi_{k-j}(X_{kj} - X'_{kj})(y_j - a_{kj}\mathbf{e}).$$

By taking the absolute value we get

$$\limsup_{k\to\infty} |x_k| \leq K \limsup_{k\to\infty} \sum_{j=k_0}^{k-1} \xi_{k-j}(X_{kj} + X'_{kj})|y_j - a_{kj}\mathbf{e}|.$$

It can then be seen that

$$\begin{aligned}
&\limsup_{k\to\infty} \left(\|x_k\|_{p,\mu}\right)^p \\
= \;& K^p \limsup_{k\to\infty} \mu^T (|x_k|)^p \\
\leq \;& K^p \limsup_{k\to\infty} \mu^T \left[\sum_{j=k_0}^{k-1} \xi_{k-j}(X_{kj} + X'_{kj}) (|y_j - a_{kj}\mathbf{e}|)\right]^p \\
= \;& K^p \limsup_{k\to\infty} \mu^T \left[\frac{\left(\sum_{j=k_0}^{k-1} \xi_{k-j} \frac{1}{2}(X_{kj} + X'_{kj}) 2 (|y_j - a_{kj}\mathbf{e}|)\right)}{\sum_{j=k_0}^{k-1} \xi_{k-j}}\right]^p \left(\sum_{j=k_0}^{k-1} \xi_{k-j}\right)^p.
\end{aligned}$$

By using Jensen's inequality (with the convex function $x \mapsto x^p$), we get:

$$\begin{aligned}
&\limsup_{k\to\infty} \left(\|x_k\|_{p,\mu}\right)^p \\
\leq \;& K^p \limsup_{k\to\infty} \mu^T \frac{\sum_{j=k_0}^{k-1} \xi_{k-j} \frac{1}{2}(X_{kj} + X'_{kj}) (2|y_j - a_{kj}\mathbf{e}|)^p}{\sum_{j=k_0}^{k-1} \xi_{k-j}} \left(\sum_{j'=k_0}^{k-1} \xi_{k-j'}\right)^p \\
= \;& K^p \limsup_{k\to\infty} \sum_{j=k_0}^{k-1} \xi_{k-j} {\mu_{kj}}^T [2|y_j - a_{kj}\mathbf{e}|]^p \left(\sum_{j'=k_0}^{k-1} \xi_{k-j'}\right)^{p-1} \\
\leq \;& K^p \limsup_{k\to\infty} \sum_{j=k_0}^{k-1} \xi_{k-j} \left[2 \|y_j - a_{kj}\mathbf{e}\|_{p,\mu_{kj}}\right]^p K'^{p-1}
\end{aligned}$$

$$
\begin{aligned}
&= K^p K'^{p-1} \limsup_{k\to\infty} \sum_{j=k_0}^{k-1} \xi_{k-j} \left[\operatorname*{span}_{p,\mu_{kj}} \left[y_j \right] \right]^p \\
&\leq K^p K'^{p-1} \limsup_{k\to\infty} \sum_{j=k_0}^{k-1} \xi_{k-j} \left[\sup_{k'\geq j'\geq k_0} \operatorname*{span}_{p,\mu_{k'j'}} \left[y_{j'} \right] \right]^p \\
&= K^p K'^{p-1} K' \left[\sup_{k'\geq j'\geq k_0} \operatorname*{span}_{p,\mu_{k'j'}} \left[y_{j'} \right] \right]^p \\
&= K^p K'^{p} \left[\sup_{k'\geq j'\geq k_0} \operatorname*{span}_{p,\mu_{k'j'}} \left[y_{j'} \right] \right]^p
\end{aligned}
$$

where we used $\sum_{j=k_0}^{k-1} \xi_{k-j} = K'$.

As this is true for all k_0, and as $k_0 \mapsto \sup_{k'\geq j'\geq k_0} \operatorname{span}_{p,\mu_{k'j'}} \left[y_{j'} \right]$ is non-increasing, the result follows.

Appendix G. Proofs of Lemma 28 and Proposition 30 (analysis of the undiscounted case)

This last section contains the Proofs of Lemma 28 and Proposition 30 that provide the analysis of an undiscounted problem like Tetris.

G.1 Proof of Lemma 28 (componentwise bound)

First of all, the relation expressed in Equation 29 between the loss and the stochastic matrices, which we restate here for clarity:

$$
\forall k_0, \quad \limsup_{k\to\infty} \; v_* - v^{\pi_k} \leq \limsup_{k\to\infty} \; \sum_{j=k_0}^{k-1} \delta_{k-j} \left[G_{jk} - G'_{jk} \right] \epsilon_j,
$$

is obtained by simply rewriting the first inequality of Lemma 21 with $\gamma = 1$ and $\beta = 1$ (note in particular that the terms δ_{k-j} collapse through the definition of G_{jk} and G'_{jk}).

To complete the proof of the lemma, we need to show that the matrices G_{jk} and G'_{jk} are substochastic matrices. By construction, these matrices are sum of non-negative matrices so we only need to show that their max norm is smaller than or equal to 1.

For all n, write $\mathcal{M}_n$ the set of matrices that is defined as follows:

- for all sets of n policies $(\pi_1, \pi_2, \cdots, \pi_n)$, $\quad P_{\pi_1} P_{\pi_2} \cdots P_{\pi_n} \in \mathcal{M}_n$;
- for all $\eta \in (0,1)$, and $(P, Q) \in \mathcal{M}_n \times \mathcal{M}_n$, $\quad \eta P + (1-\eta) Q \in \mathcal{M}_n$.

The motivation for introducing this set is that we have the following properties: For all n, $P \in \mathcal{M}_n$ is a substochastic matrix such that $\|P\|_\infty \leq \alpha^{\left\lfloor \frac{n}{n_0} \right\rfloor}$. We use the somewhat abusive notation Π_n for denoting any element of $\mathcal{M}_n$. For instance, for some matrix P, writing $P = a\Pi_i + b\Pi_j\Pi_k = a\Pi_i + b\Pi_{j+k}$ should be read as follows: there exists $P_1 \in \mathcal{M}_i$, $P_2 \in \mathcal{M}_j$, $P_3 \in \mathcal{M}_k$ and $P_4 \in \mathcal{M}_{k+j}$ such that $P = aP_1 + bP_2P_3 = aP_1 + bP_4$.

Recall the definition of the substochastic matrix

$$A_k = (1-\lambda)(I-\lambda P_k)^{-1}P_k = (1-\lambda)\sum_{i=0}^{\infty}\lambda^i \Pi_{i+1}.$$

Let $i \leq j < k$. It can be seen that

$$\begin{aligned}(P_*)^{k-1-i}A_{i+1}A_i...A_{j+1} &= \Pi_{k-1-i}\underbrace{\left((1-\lambda)\sum_{i=0}^{\infty}\lambda^i\Pi_{i+1}\right)\cdots\left((1-\lambda)\sum_{i=0}^{\infty}\lambda^i\Pi_{i+1}\right)}_{i-j+1 \text{ terms}} \\ &= \Pi_{k-j}\underbrace{\left((1-\lambda)\sum_{i=0}^{\infty}\lambda^i\Pi_{i}\right)\cdots\left((1-\lambda)\sum_{i=0}^{\infty}\lambda^i\Pi_{i}\right)}_{i-j+1 \text{ terms}}. \qquad (54)\end{aligned}$$

Now, observe that

$$\begin{aligned}\left\|\sum_{i=0}^{\infty}\lambda^i\Pi_i\right\|_\infty &\leq \sum_{i=0}^{\infty}\lambda^i \left\|\Pi_i\right\|_\infty \\ &\leq \sum_{i=0}^{\infty}\lambda^i\alpha^{\left\lfloor\frac{i}{n_0}\right\rfloor} \\ &= \sum_{j=0}^{\infty}\sum_{i=0}^{n_0-1}\lambda^{jn_0+i}\alpha^j \\ &= \sum_{j=0}^{\infty}(\lambda^{n_0}\alpha)^j\sum_{i=0}^{n_0-1}\lambda^i \\ &= \frac{1-\lambda^{n_0}}{(1-\lambda^{n_0}\alpha)(1-\lambda)}. \qquad (55)\end{aligned}$$

As a consequence, writing $\eta := \frac{1-\lambda^{n_0}}{1-\lambda^{n_0}\alpha}$, we see from Equation 54 that

$$\left\|(P_*)^{k-1-i}A_{i+1}A_i...A_{j+1}\right\|_\infty \leq \alpha^{\left\lfloor\frac{k-j}{n_0}\right\rfloor}\eta^{i-j+1}.$$

Similarly, by using Equation 55 and noticing that $\frac{1-\lambda^{n_0}}{1-\lambda} \xrightarrow{\lambda\to 1} n_0$, it can be seen that

$$\left\|(I-P_k)^{-1}A_kA_{k-1}\cdots A_{j+1}\right\|_\infty \leq \frac{n_0}{1-\alpha}\alpha^{\left\lfloor\frac{k-j}{n_0}\right\rfloor}\eta^{k-j}.$$

We are ready to bound the norm of the matrix G_{jk}:

$$\begin{aligned}
\|G_{jk}\|_\infty &\leq \frac{\alpha^{\left\lfloor \frac{k-j}{n_0} \right\rfloor}}{\delta_{k-j}} \left[\frac{\lambda}{1-\lambda} \sum_{i=j}^{k-1} \eta^{i-j+1} + \frac{n_0 \eta^{k-j}}{1-\alpha} \right] \\
&= \frac{\alpha^{\left\lfloor \frac{k-j}{n_0} \right\rfloor}}{\delta_{k-j}} \left[\left(\frac{\lambda}{1-\lambda} \right) \eta \left(\frac{1-\eta^{k-j}}{1-\eta} \right) + \frac{n_0 \eta^{k-j}}{1-\alpha} \right] \\
&= \frac{\alpha^{\left\lfloor \frac{k-j}{n_0} \right\rfloor}}{\delta_{k-j}} \left[\left(\frac{\lambda}{1-\lambda} \right) \left(\frac{1-\lambda^{n_0}}{1-\lambda^{n_0}\alpha} \right) \left(\frac{1-\eta^{k-j}}{1-\eta} \right) + \frac{n_0 \eta^{k-j}}{1-\alpha} \right] \\
&= \frac{\alpha^{\left\lfloor \frac{k-j}{n_0} \right\rfloor}}{\delta_{k-j}} \left[\left(\frac{1-\lambda^{n_0}}{1-\lambda} \right) \left(\frac{\lambda}{1-\lambda^{n_0}\alpha} \right) \left(\frac{1-\eta^{k-j}}{1-\eta} \right) + \frac{n_0 \eta^{k-j}}{1-\alpha} \right] \\
&= 1.
\end{aligned}$$

where we used the definition of η. Therefore G_{jk} is a substochastic matrix. It trivially follows that G'_{jk} is also a substochastic matrix.

G.2 Proof of Proposition 30 (L_p norm bound)

In order to prove the L_p norm bound of Proposition 30, we rely on the following variation of Lemma 23.

Lemma 39 *If x_k and y_k are sequences of vectors and X_{jk}, X'_{jk} sequences of substochastic matrices satisfying*

$$\forall k_0, \quad \limsup_{k\to\infty} |x_k| \leq K \limsup_{k\to\infty} \sum_{j=k_0}^{k-1} \xi_{k-j}(X_{kj} - X'_{kj}) y_j,$$

where $(\xi_i)_{i\geq 1}$ is a sequence of non-negative weights satisfying:

$$\sum_{i=1}^{\infty} \xi_i = K' < \infty,$$

then, for all distribution μ,

$$\mu_{kj} := \frac{1}{2}(X_{kj} + X'_{kj})^T \mu$$

is a non-negative vector and $\tilde{\mu}_{kj} := \frac{\mu_{kj}}{\|\mu_{kj}\|_1}$ is a distribution, and

$$\limsup_{k\to\infty} \|x_k\|_{p,\mu} \leq K K' \lim_{k_0\to\infty} \left[\sup_{k\geq j\geq k_0} \|y_j\|_{p,\tilde{\mu}_{kj}} \right].$$

Proof The proof follows the lines of that of Lemma 23 in Appendix F. The differences are as follows:

- since X_{jk} and X'_{jk} are substochastic matrices (and not stochastic matrices), we have in general $X_{jk}e \neq X'_{jk}e$ and must take $a_{kj} = 0$, which in turn gives an L_p norm bound instead of the L_p span seminorm bound;
- to express the bound in terms of the distributions $\tilde{\mu}_{kj}$, we use the fact that $\mu_{kj} \leq \tilde{\mu}_{kj}$ which derives from $\|\mu_{kj}\|_1 \leq 1$ since X_{jk} and X'_{jk} are substochastic matrices. ■

Proposition 30 is obtained by applying this Lemma and an anologue of Lemma 25 for L_p norm on the componentwise bound (Lemma 28 — see previous subsection). The only remaining thing that needs to be checked is that $\sum_{i=1}^{\infty} \delta_i$ has the right value. This is what we do now.

Similary to Equation 55, one can see that:

$$\sum_{i=0}^{\infty} \alpha^{\left\lfloor \frac{i}{n_0} \right\rfloor} \eta^i = \frac{1-\eta^{n_0}}{(1-\eta^{n_0}\alpha)(1-\eta)}$$

and

$$\sum_{i=0}^{\infty} \alpha^{\left\lfloor \frac{i}{n_0} \right\rfloor} (1-\eta^i) = \frac{n_0}{1-\alpha} - \frac{1-\eta^{n_0}}{(1-\eta^{n_0}\alpha)(1-\eta)}.$$

As a consequence:

$$\begin{aligned}
\sum_{i=0}^{\infty} \delta_i &= \sum_{i=0}^{\infty} \alpha^{\left\lfloor \frac{i}{n_0} \right\rfloor} \left(\frac{1-\lambda^{n_0}}{1-\lambda}\right)\left(\frac{\lambda}{1-\lambda^{n_0}\alpha}\right)\left(\frac{1-\eta^i}{1-\eta}\right) + \frac{n_0\eta^i}{1-\alpha} \\
&= \left(\frac{1-\lambda^{n_0}}{1-\lambda}\right)\left(\frac{\lambda}{1-\lambda^{n_0}\alpha}\right)\left(\frac{\sum_{i=0}^{\infty} \alpha^{\left\lfloor \frac{i}{n_0} \right\rfloor}(1-\eta^i)}{1-\eta}\right) + \frac{n_0 \sum_{i=0}^{\infty} \alpha^{\left\lfloor \frac{i}{n_0} \right\rfloor}\eta^i}{1-\alpha} \\
&= \left(\frac{1-\lambda^{n_0}}{1-\lambda}\right)\left(\frac{\lambda}{1-\lambda^{n_0}\alpha}\right)\left(\frac{1}{1-\eta}\right)\left(\frac{n_0}{1-\alpha} - \frac{1-\eta^{n_0}}{(1-\eta^{n_0}\alpha)(1-\eta)}\right) \\
&\qquad + \left(\frac{n_0}{1-\alpha}\right)\left(\frac{1-\eta^{n_0}}{(1-\eta^{n_0}\alpha)(1-\eta)}\right) \\
&= \lambda f(\lambda)\frac{1}{1-\eta}(f(1)-f(\eta)) + f(1)f(\eta) \qquad (56)
\end{aligned}$$

with for all x, $f(x) := \frac{(1-x^{n_0})}{(1-x)(1-x^{n_0}\alpha)}$ and $f(1) = \frac{n_0}{1-\alpha}$ by continuity. Now, we can conclude by noticing that

$$\sum_{i=1}^{\infty} \delta_i = \sum_{i=0}^{\infty} \delta_i - \delta_0$$

and $\delta_0 = \frac{n_0}{1-\alpha} = f(1)$.