PID Accelerated Value Iteration Algorithm

Amir-massoud Farahmand [1 2] **Mohammad Ghavamzadeh** [3]

Abstract

The convergence rate of Value Iteration (VI), a fundamental procedure in dynamic programming and reinforcement learning, for solving MDPs can be slow when the discount factor is close to one. We propose modifications to VI in order to potentially accelerate its convergence behaviour. The key insight is the realization that the evolution of the value function approximations $(V_k)_{k\geq 0}$ in the VI procedure can be seen as a dynamical system. This opens up the possibility of using techniques from *control theory* to modify, and potentially accelerate, this dynamics. We present such modifications based on simple controllers, such as PD (Proportional-Derivative), PI (Proportional-Integral), and PID. We present the error dynamics of these variants of VI, and provably (for certain classes of MDPs) and empirically (for more general classes) show that the convergence rate can be significantly improved. We also propose a *gain adaptation* mechanism in order to automatically select the controller gains, and empirically show the effectiveness of this procedure.

1. Introduction

Value Iteration (VI) is a key algorithm for solving Dynamic Programming (DP) problems, and its sampled-based variants are the basis for many Reinforcement Learning (RL) algorithms, e.g., TD-like sample-based asynchronous update algorithms (Bertsekas & Tsitsiklis, 1996; Sutton & Barto, 2019) and Fitted Value Iteration procedures (Ernst et al., 2005; Munos & Szepesvári, 2008; Farahmand et al., 2009; Mnih et al., 2015; Tosatto et al., 2017; Chen & Jiang, 2019). VI finds the fixed point of the Bellman T^π or the Bellman optimality T^* operators, which are the value or action-value functions of policy π or the optimal policy, by repeatedly applying the Bellman operator to the current approximation of the value or action-value functions, i.e., $V_{k+1} \leftarrow T^\pi V_k$ or $Q_{k+1} \leftarrow T^* Q_k$. For discounted MDPs, the Bellman operator is a contraction, and standard fixed-point iteration results, such as Banach fixed-point theorem, guarantee the convergence of the sequence generated by VI to the true value function (either the optimal one or the one of a given policy, depending on the choice of the Bellman operator). The convergence of VI to the value function depends on the discount factor $\gamma < 1$ and is of $O(\gamma^k)$. This is slow when $\gamma \approx 1$. The goal of this research is to investigate whether one might *accelerate* the convergence of VI, i.e., developing a procedure that converges to the value function faster than the conventional VI.

This work brings tools from control theory to accelerate VI. The goal of control theory, generally speaking, is to design a controller for a given dynamical system in order to make it behave in a certain desired way. Depending on the problem, the desired behaviour can be convergence to a set-point with a certain speed or robustness to disturbances. Casting the VI procedure as a dynamical system, we may wonder whether we can design a controller to modify its dynamics, and perhaps make it faster or more robust to disturbances.

This paper investigates this question in some details. We establish a connection between VI and dynamical systems in Section 2. This connection allows us to take the novel perspective of using controllers to modify, and in particular to accelerate, the dynamics of VI. We introduce accelerated variants of VI by coupling its dynamics with PD (Proportional-Derivative), PI (Proportional-Integral), and PID controllers (Section 3). These controllers are among the simplest and yet most ubiquitous and effective classes of controllers in the arsenal of control theory and engineering. We call the resulting algorithms *accelerated VI*, and refer to them by PD/PI/PID VI. As an example of such a controller, the update rule of the (simplified) PD VI is $V_{k+1} = T^\pi V_k + \kappa_d (V_k - V_{k-1})$. The derivative term $(V_k - V_{k-1})$ measures the rate of change in the value function. We describe the error dynamics of these accelerated variants of VI for the policy evaluation problem (when the Bellman operator is T^π) in Section 4. We briefly describe the problem of choosing the controller gains in the same section, and describe four different approaches in more details in Appendix D. Specifically, we provide an analytical solution for the class of *reversible* Markov chains in Appendix E.

[1]Vector Institute, Toronto, Canada [2]Department of Computer Science, University of Toronto, Canada [3]Google Research, Mountain View, California, USA. Correspondence to: Amir-massoud Farahmand <farahmand@vectorinstitute.ai>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

We propose a gain adaptation/meta-learning procedure to automatically select the controller gains in Section 5. We empirically study the behaviour of these variants on some simple problems, and observe that they can be effective in accelerating the convergence of VI (Section 6). Due to the space limitation, we refer to appendices for more detailed analyses and studies.

2. Value Iteration as a Dynamical System

We show how the VI algorithm can be represented as a dynamical system. This prepares us for the next section when we use PID controller in order to change VI's dynamics. Before that, let us briefly introduce our notations.

We consider a discounted Markov Decision Process (MDP) $(\mathcal{X}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma)$ (Bertsekas & Tsitsiklis, 1996; Szepesvári, 2010; Sutton & Barto, 2019). We defer formal definitions to Appendix A. We only mention that for a policy π, we denote by $\mathcal{P}^\pi$ its transition kernel, by $r^\pi : \mathcal{X} \to \mathbb{R}$ the expected value of its reward distribution, and by V^π and Q^π its state-value and action-value functions. We also represent the optimal state and action-value functions by V^* and Q^*. Finally, we define the Bellman operator $T^\pi : \mathcal{B}(\mathcal{X}) \to \mathcal{B}(\mathcal{X})$ for policy π and the Bellman optimality operator $T^* : \mathcal{B}(\mathcal{X} \times \mathcal{A}) \to \mathcal{B}(\mathcal{X} \times \mathcal{A})$ as

$$(T^\pi V)(x) \triangleq r^\pi(x) + \gamma \int \mathcal{P}^\pi(\mathrm{d}y|x) V(y),$$

$$(T^* Q)(x,a) \triangleq r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}y|x,a) \max_{a' \in \mathcal{A}} Q(y,a').$$

The value function V^π and optimal action-value function Q^* are the fixed points of the operators T^π and T^*, respectively.

We start our discussion by describing the VI procedure for the policy evaluation (PE) problem, which uses the Bellman operator of a policy π (i.e., T^π), instead of the problem of finding the optimal value function (simply referred to as control), which uses the Bellman optimality operator T^*. For the PE problem, the involved dynamical systems are linear, and the discussion of how to the design controllers is easier and more intuitive. The methods, however, work in the control case too, where the operator T^* and the involved dynamical systems are nonlinear. Algorithmically, it does not matter whether the underlying Bellman operator is linear or not. We also note that even though the developed algorithms work for general state and action spaces (computational issues aside), we focus on finite state space problems in the analysis of the error dynamics in Section 4, as it allows us to use tools from linear algebra. In this case, $\mathcal{P}^\pi$ is a $d \times d$ matrix with $d = |\mathcal{X}|$ being the number of states.

Consider the VI procedure for policy evaluation:

$$V_{k+1} = T^\pi V_k, \tag{1}$$

which is a shorthand notation for $V_{k+1}(x) = (T^\pi V_k)(x), \ \forall x \in \mathcal{X}$. Let us define $e_k = V_k - V^\pi$ as the error between the value function approximation V_k and true value function V^π. The error dynamics can be written as

$$\begin{aligned} e_{k+1} &= V_{k+1} - V^\pi = T^\pi V_k - V^\pi = T^\pi V_k - T^\pi V^\pi \\ &= \gamma \mathcal{P}^\pi (V_k - V^\pi) = \gamma \mathcal{P}^\pi e_k. \end{aligned} \tag{2}$$

The behaviour of this dynamics is related to the eigenvalues of $\gamma \mathcal{P}^\pi$. Since $\mathcal{P}^\pi$ is a stochastic matrix, it has one eigenvalue equal to 1, and the others are all within the interior of the unit circle. Hence, the largest eigenvalue of $\gamma \mathcal{P}^\pi$ has a magnitude of γ. As $\gamma < 1$, this ensures the (exponential) stability of this error dynamical system, which behaves as $c_1 \gamma^k$, for a constant $c_1 > 0$.

With some extra conditions, we can say more about the location of eigenvalues than merely being within the unit circle. If the Markov chain induced by $\mathcal{P}^\pi$ is *reversible*, that is, its stationary distribution ρ^π satisfies the *detailed balance* equation

$$\rho^\pi(x) \mathcal{P}^\pi(y|x) = \rho^\pi(y) \mathcal{P}^\pi(x|y), \tag{3}$$

for all $x, y \in \mathcal{X}$, then all eigenvalues are real.

Let us study the error dynamics (2) more closely. Assume that $\mathcal{P}^\pi$ is *diagonalizable* (having distinct eigenvalues is a sufficient, but not necessary, condition for diagonalizability). In this case, there exists a $d \times d$ similarity transformation S such that $\mathcal{P}^\pi = S \Lambda S^{-1}$, with $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_d)$. We denote by $\varepsilon_k = S^{-1} e_k$, the error in the transformed coordinate system. By multiplying both sides of (2) from left by S^{-1}, we obtain

$$\begin{aligned} & S^{-1} e_{k+1} = \gamma S^{-1} \mathcal{P}^\pi e_k = \gamma S^{-1} S \Lambda S^{-1} e_k \\ \implies & \varepsilon_{k+1} = \gamma \Lambda \varepsilon_k. \end{aligned} \tag{4}$$

Thus, the dynamics of the i-th component $\varepsilon_k(i)$ of the sequence $(\varepsilon_k)_{k \geq 0}$ can be written as $\varepsilon_{k+1}(i) = \gamma \lambda_i \varepsilon_k(i)$, for $i = 1, \ldots, d$. As a result, $\varepsilon_k(i) = (\gamma \lambda_i)^k \varepsilon_0(i)$, or more succinctly $\varepsilon_k = \gamma^k \Lambda^k \varepsilon_0$. Therefore, the error dynamics is

$$e_k = S (\gamma \Lambda)^k S^{-1} e_0.$$

Since the eigenvalue with the largest magnitude is $\lambda_1 = 1$, the behaviour of the slowest term is of $O(\gamma^k)$, which determines the dominant behaviour of the VI procedure. Note that if we have complex eigenvalues in Λ, which always come in conjugate pairs, the error e_k of the VI procedure might show oscillatory, yet convergent, behaviour.

Is it possible to modify this error dynamics, so that we obtain a faster convergence rate? Changing the behaviour of a dynamical system is an important topic in control theory. Depending on whether the underlying dynamics is linear or

nonlinear, or whether it is known or unknown, etc., various approaches have been developed, see e.g., Dorf & Bishop (2008); Khalil (2001); Aström & Wittenmark (1994); Krstic et al. (1995); Burl (1998); Bertsekas & Shreve (1978); Zhou & Doyle (1998); Skogestad & Postlethwaite (2005). In this work, we would like to study the feasibility of using these techniques for the purpose of accelerating the dynamical system of obtaining the value function. Introducing some simple methods for this purpose is the topic of the next section.

3. PID-Like Controllers for Accelerating VI

PD, PI, and PID (Proportional-Integral-Derivative) are among the simplest, and yet most practical controllers in control engineering (Dorf & Bishop, 2008; Ogata, 2010). They can be used to change the dynamics of a plant (i.e., the system to be controlled) to behave in a desired way. Objectives such as stabilizing the dynamics, improving the transient behaviour, or improving the robustness to external disturbances are commonly achieved using these controllers. They have been used to control both linear and nonlinear dynamical systems. Even though they are not necessarily optimal controllers for a given plant, their ease of use and robustness have made them the controller of choice in many applications. We now show how these controllers can be used for changing the dynamics of the VI algorithm.

P Controller. To see how VI can be viewed as a Proportional feedback controller, consider the plant to be a simple integrator with u_k being its input,

$$V_{k+1} = V_k + u_k. \tag{5}$$

As the desired value (known as reference signal in control engineering parlance) of this system is V^π, the feedback error is $e_k = V_k - V^\pi$. The controller is the transformation that takes e_k and generates u_k. A proportional linear controller generates u_k by multiplying e_k by a matrix K_p. Let us choose the matrix of the form $K_p = -\kappa_p(\mathbf{I} - \gamma\mathcal{P}^\pi)$, with κ_p being a real number. Therefore,

$$u_k = -\kappa_p(\mathbf{I} - \gamma\mathcal{P}^\pi)e_k. \tag{6}$$

With this choice of controller, the dynamics of the feedback controlled system would be

$$V_{k+1} = V_k + u_k = V_k - \kappa_p(\mathbf{I} - \gamma\mathcal{P}^\pi)(V_k - V^\pi).$$

This can be simplified, by adding and subtracting r^π and some simple algebraic manipulations, to

$$\begin{aligned} V_{k+1} &= (1-\kappa_p)V_k - \kappa_p\big(-V^\pi + (r^\pi + \gamma\mathcal{P}^\pi V^\pi) \\ &\qquad - (r^\pi + \gamma\mathcal{P}^\pi V_k)\big) \\ &= (1-\kappa_p)V_k - \kappa_p\left(-V^\pi + T^\pi V^\pi - T^\pi V_k\right) \\ &= (1-\kappa_p)V_k + \kappa_p T^\pi V_k, \end{aligned} \tag{7}$$

where we used $V^\pi = T^\pi V^\pi$ in the last step. With the choice of $\kappa_p = 1$, this proportional controller for the specified plant is the same as the conventional VI (1).[1]

The control signal generated by this particular controller (6) is closely related to the *Bellman residual*. Recall that the Bellman residual of a value function V is $\mathrm{BR}(V) = T^\pi V - V$. Since $V^\pi = T^\pi V^\pi$ and $e = V - V^\pi$, we may write

$$\begin{aligned} \mathrm{BR}(V) &= T^\pi V - V = (T^\pi V - V^\pi) - (V - V^\pi) \\ &= (T^\pi V - T^\pi V^\pi) - (V - V^\pi) \\ &= \gamma\mathcal{P}^\pi e - e = -(\mathbf{I} - \gamma\mathcal{P}^\pi)e. \end{aligned} \tag{8}$$

So the dynamics of the P variant of VI (7) can be written as

$$V_{k+1} = V_k + \kappa_p \mathrm{BR}(V_k). \tag{9}$$

Comparing with (5), we see that the P variant of VI is a simple integrator with a control signal u_k that is proportional to the Bellman residual.

We now introduce the PD, PI, and PID variants of VI.

PD Controller. We start with a simplified PD variant. Given a scalar gain κ_d, we define it as

$$V_{k+1} = T^\pi V_k + \kappa_d\left(V_k - V_{k-1}\right). \tag{10}$$

The term $(V_k - V_{k-1})$ is the derivative term of a PD controller.[2] The role of the derivative term can be thought of as approximating the value function V_{k+1} using a linear extrapolation based on the most recent values V_k and V_{k-1}. When the change between V_k and V_{k-1} is large, this term encourages a large change to V_{k+1}.

More generally, we can allow the P term to have a gain κ_p other than 1, and instead of using a scalar gain κ_d, we can use a matrix gain $K_d \in \mathbb{R}^{d\times d}$. This leads to

$$V_{k+1} = (1-\kappa_p)V_k + \kappa_p T^\pi V_k + K_d\left(V_k - V_{k-1}\right). \tag{11}$$

With the choice of $K_d = \kappa_d\mathbf{I}$ and $\kappa_p = 1$, we retrieve (10). Note that adding a derivative term does not change the fixed point of the Bellman operator, as at the fixed point we have $V^\pi = (1-\kappa_p)V^\pi + \kappa_p T^\pi V^\pi + K_d(V^\pi - V^\pi) = T^\pi V^\pi$. The addition of the derivative term, however, can change the convergence property to the fixed point. The goal is to find the controller gains to accelerate this convergence. We study the dynamics in Section 4.

PI Controller. The PI controller is defined as the following coupled equations:

$$\begin{aligned} z_{k+1} &= \beta z_k + \alpha\mathrm{BR}(V_k), \\ V_{k+1} &= T^\pi V_k + K_I\left[\beta z_k + \alpha\mathrm{BR}(V_k)\right], \end{aligned} \tag{12}$$

[1] This iterative procedure is known as the *Krasnoselskii* iteration in the fixed-point iteration literature (Berinde, 2007). It is reduced to the Picard iteration for $\kappa_p = 1$ of the conventional VI.

[2] It is technically a finite difference and not a derivative. Yet, it is common to call it a derivative term.

where α, β are scalars and K_I is a matrix with an appropriate size. When we use a scalar integrator gain κ_I, we replace K_I with κ_I. Here we present the special case with $\kappa_p = 1$, but generalization is the same as before (and we show a more general PID case soon). The variable z_k is the exponentially weighted average of the Bellman residuals $\text{BR}(V_i)$, for $i \leq k$. From the filtering theory perspective, it is an autoregressive filter that performs low-pass filtering over the sequence of Bellman residuals (as long as $|\beta| < 1$). The additional integrator term $K_I[\beta z_k + \alpha \text{BR}(V_k)] = K_I z_{k+1}$ adds this weighted average of the past Bellman residuals to the current approximation of the value function. This is similar to the momentum term in SGD, with a difference that instead of gradients, we are concerned about the Bellman residuals.[3] Note that $(z, V) = (0, V^\pi)$ is the fixed point of this modified dynamics. So as long as this new dynamics is stable, its V_k converges to the desired value function V^π. The dynamics depends on the choice of the controller gains. We will study it in Section 4.

PID Controller. Finally, the PID variant would be

$$\begin{aligned} z_{k+1} =& \beta z_t + \alpha \text{BR}(V_k), \\ V_{k+1} =& (1 - K_p)V_k + K_p T^\pi V_k + K_I \left[\beta z_k + \alpha \text{BR}(V_k)\right] + \\ & K_d(V_k - V_{k-1}). \end{aligned} \tag{13}$$

Control Case. The accelerated VI for control is essentially the same. We only use the action-value function Q instead of V (though nothing prevents us from using V). The main change is the use of the Bellman optimality operator T^* instead of the Bellman operator T^π. The definition of the Bellman residual would consequently change to $\text{BR}^*(Q) = T^*Q - Q$. The PID accelerated VI for control is then

$$\begin{aligned} z_{k+1} =& \beta z_t + \alpha \text{BR}^*(Q_k), \\ Q_{k+1} =& (1 - K_p)Q_k + K_p T^* Q_k + K_I \left[\beta z_k + \alpha \text{BR}^*(Q_k)\right] \\ & + K_d(Q_k - Q_{k-1}). \end{aligned} \tag{14}$$

Notice that in this case the dimension of z is the same as Q, i.e., $z : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$. The control variants of PD and PI VI algorithms are similar too. We briefly compare these variations with a few other algorithms in Section 7, and postpone the detailed comparison to Appendix G.

4. The Error Dynamics

We first present Proposition 1, which describes the dynamics of error $e_k = V_k - V^\pi$ in PID VI, similar to what we have for the conventional VI in (2). Additional results, including the dynamics of PI and PD VI, are reported in Appendix B.

[3] One may wonder why we did not define the integrator based on the value errors $e_k = V_k - V^\pi$, and had $z_{k+1} = \beta z_t + \alpha e_k$ instead. The reason is that we cannot compute e_k because V^π is not known before solving the problem. The Bellman residual plays the role of a proxy for the value error.

We then briefly describe several ways the dynamics can be modified, paving our way for the gain adaptation method described in Section 5. The results of this section and the aforementioned appendix are for policy evaluation with a finite state space. We discuss the necessary changes needed to deal with the control problem (using T^*) in Appendix C.

Proposition 1 (Error Dynamics of PID VI)**.** *Let $e_k = V_k - V^\pi$ and the integrator's state be z_k. The dynamics of the **PID controller** with gains K_p, K_I, K_d is*

$$\begin{bmatrix} e_{k+1} \\ e_k \\ z_{k+1} \end{bmatrix} = A_{\text{PID}} \begin{bmatrix} e_k \\ e_{k-1} \\ z_k \end{bmatrix}, \tag{15}$$

with $A_{\text{PID}} \triangleq \begin{bmatrix} (\mathbf{I} - K_p) + \gamma K_p \mathcal{P}^\pi + \alpha K_I(\gamma \mathcal{P}^\pi - \mathbf{I}) + K_d & -K_d & \beta K_I \\ \mathbf{I} & 0 & 0 \\ \alpha(\gamma \mathcal{P}^\pi - \mathbf{I}) & 0 & \beta \mathbf{I} \end{bmatrix}$.

This result can be presented in a simpler and more intuitive form, if we only consider scalar gains and assume that $\mathcal{P}^\pi$ is diagonalizable. We postpone reporting the result to Appendix B. Just as an example, we can show that the eigenvalues of the PD VI are located at the roots of the polynomial $\prod_{i=1}^{d} \left[\mu^2 - (1 + \kappa_d - \kappa_p(1 - \gamma\lambda_i))\mu + \kappa_d\right]$ (Corollary 5 in Appendix B).

The convergence behaviour of the PD, PI, and PID variants of VI for PE is completely specified by the location of the eigenvalues of the error dynamics matrices A_{PD}, A_{PI}, and A_{PID}. The dominant behaviour depends on their spectral radius $\rho(A)$, the eigenvalue with the largest modulus. The dynamics is convergent when $\rho < 1$, and is accelerated compared to the conventional VI, if $\rho < \gamma$. Whenever convergent, the smaller the value of ρ is, the faster the rate would be. When $(\kappa_p, \kappa_I, \kappa_d) = (1, 0, 0)$ and $(\alpha, \beta) = (0, 0)$ (for PID), we retrieve the original VI dynamics. The same is true for the PD and PI variants, with obvious modifications. So by falling back on the default parameters, these methods can be at least as fast the conventional VI.

As the eigenvalues are continuous functions of the elements of a matrix, changing the controller gains leads to a continuous change of the spectral radius, hence the possibility of acceleration. The spectral radius, however, is a complicated function of a matrix, so a simple equation for an arbitrary $\mathcal{P}^\pi$ and controller gains does not exist. A natural question is then: *How should one choose the gains in order to achieve the intended acceleration?*

We suggest four possibilities: (i) consider them as hyperparameters to be adjusted through a model selection procedure; (ii) formulate the controller design problem as an optimization problem; (iii) analytically find a set of gains that accelerate a subset of MDPs, without the exact knowledge of the MDP itself; and finally, (iv) adapt gains throughout the accelerated VI procedure. We discuss (i), (ii), and (iii) in more details in Appendix D. We only briefly note

that (ii) may not be feasible for large MDPs, especially if our ultimate goal is to extend these methods to the RL setting. Option (iii) is possible if we make some extra assumptions. One such assumption is the *reversibility* of the Markov chain induced by the policy. With that assumption, we can analytically find the gains for PD VI and show that if we choose $\kappa_p^* = \frac{2}{1+\sqrt{1-\gamma^2}}$ and $\kappa_d^* = (\frac{\sqrt{1+\gamma}-\sqrt{1-\gamma}}{\sqrt{1+\gamma}+\sqrt{1-\gamma}})^2$, we get the effective rate of

$$\gamma_{\text{PD}} = \frac{\sqrt{1+\gamma}-\sqrt{1-\gamma}}{\sqrt{1+\gamma}+\sqrt{1-\gamma}}.$$

This is smaller than γ for $\gamma < 1$, showing that the procedure is accelerated (Proposition 6 in Appendix E). We note that the class of reversible Markov chains is limited.

We focus on (iv), the gain adaptation approach, in the next section, which seems to be the most promising because it is not restricted to a subset of MDPs, can adapt to the problem in hand, and is feasible for large MDPs. It also appears to be extendable to the RL setting.

Remark 1. For the control case, the error dynamics has the same form as in Proposition 1, but with a time-varying A_{PID} matrix. In that case, the spectral radius does not determine the stability. Instead, we can use the *joint spectral radius* (Jungers, 2009). See Appendix C for more discussions.

5. Gain Adaptation

We describe a method to adaptively tune the controller gains $(\kappa_p, \kappa_I, \kappa_d)$ throughout the iterations of an accelerated VI procedure. Our approach is based on computing the gradient of an appropriately-defined loss function w.r.t. the gains, and updating them based on the gradient. The idea has conceptual similarities to the learning rate adaptation mechanisms, such as Incremental Delta-Bar-Delta (IDBD) (Sutton, 1992; Almeida et al., 1999), stochastic meta-descent (SMD) (Schraudolph, 1999; Mahmood et al., 2012), and hyper-gradient descent (Baydin et al., 2018).

To define a gradient-based gain adaptation algorithm, we need to specify a loss function. An example would be the norm of the error $e_k = V^\pi - V_k$ (or $e_k = Q^* - Q_k$ for control) at the next iteration, i.e., at iteration $k-1$, we compute the gradient of $\|e_k\|_2^2$ w.r.t. the controller gains.[4] This approach, however, is not practical: the errors e_k's cannot be computed, as we do not know V^π or Q^*. Thus, we use a surrogate loss function that is easy to compute.

We choose the Bellman errors $\|T^\pi V_k - V_k\|_2^2$ (PE) or $\|T^* Q_k - Q_k\|_2^2$ (control) as the surrogate loss functions. These quantities can be computed given the value function, V_k or Q_k, and the Bellman operator, T^π or T^*. The Bellman error is a reasonable surrogate because having a zero Bellman error implies having a zero error in approximating the value function. We can also quantify the relation between them more precisely. For example, if the error is measured according to the supremum norm (and not the ℓ_2-norm as here), we have $\|V - V^\pi\|_\infty \leq \frac{\|T^\pi V - V\|_\infty}{1-\gamma}$ (Williams & Baird, 1993). For the ℓ_2-norm, we have similar results, e.g., Theorem 5.3 of Munos (2007) for the Bellman optimality error or Proposition 7 in Appendix F.1 for the Bellman error in the PE case. Therefore, we define the following loss functions for the PE and control cases:

$$J_{\text{BE}}(k) = \frac{1}{2}\|T^\pi V_k - V_k\|_2^2 = \frac{1}{2}\|\text{BR}(V_k)\|_2^2, \tag{16}$$

$$J^*_{\text{BE}}(k) = \frac{1}{2}\|T^* Q_k - Q_k\|_2^2 = \frac{1}{2}\|\text{BR}^*(Q_k)\|_2^2. \tag{17}$$

For the control case, we could also define the loss based on $\text{BR}^*(V) = T^* V - V$. The gradient of $J_{\text{BE}}(k)$ w.r.t. the controller parameters is

$$\frac{\partial J_{\text{BE}}(k)}{\partial \kappa_\cdot} = \left\langle \text{BR}(V_k), \frac{\partial \text{BR}(V_k)}{\partial \kappa_\cdot} \right\rangle_{\mathcal{X}}, \tag{18}$$

where $\langle V_1, V_2 \rangle_{\mathcal{X}} = \sum_{x\in\mathcal{X}} V_1(x)V_2(x)$ (or an appropriately defined integral when $\mathcal{X}$ is a continuous state space). The derivatives of $J^*_{\text{BE}}(k)$ is similar, with obvious changes. To compute these derivatives, we require the derivative of the Bellman Residual w.r.t. each of the controller gains. Table 1 reports them for both PE and control cases. Note that as the Bellman optimality operator is nonlinear, we need some care in computing its derivative. The details of how these derivatives are obtained as well as some intuition on what they capture are in Appendices F.2 and F.3.

The gain adaptation procedure can be achieved by a receding-horizon-like procedure: At each iteration k of the accelerated VI algorithm, it computes the gradient of this objective w.r.t. the controller gains, updates the controller gains by moving in the opposite direction of the gradient, performs one step of accelerated VI to obtain the value function at iteration $k+1$, and repeats the procedure again.

One can perform gradient descent based on (18). This, however, may not lead to a desirable result. The reason is that if the dynamics of the accelerated VI is stable, both Bellman residual and its gradient will go to zero exponentially fast. Therefore, the gradient converge to zero too fast to allow enough adaptation of controller gains. To address this issue, we define the following normalized loss functions instead:

$$J_{\text{BE(norm)}}(k) = \frac{\|\text{BR}(V_k)\|_2^2}{2\,\|\text{BR}(V_{k-1})\|_2^2}, \tag{19}$$

$$J^*_{\text{BE(norm)}}(k) = \frac{\|\text{BR}^*(Q_k)\|_2^2}{2\,\|\text{BR}^*(Q_{k-1})\|_2^2}. \tag{20}$$

[4] More generally, we can unroll the accelerated algorithm for $T \geq 1$ iterations, and back-propagate the gradient of $\|e_{k+T-1}\|_2^2$ w.r.t. the parameters. For simplicity of exposition, we do not describe this in more detail here.

Table 1. Partial derivatives of the Bellman residual w.r.t. the controller's parameters for PE and control cases. $\pi(Q_k)$ is the greedy policy w.r.t. Q_k, i.e., $\pi(x;Q) \leftarrow \text{argmax}_{a\in\mathcal{A}} Q(x,a)$.

	$\frac{\partial \text{BR}(V_k)}{\partial \cdot}$	$\frac{\partial \text{BR}^*(Q_k)}{\partial \cdot}$
κ_p	$-(\mathbf{I}-\gamma\mathcal{P}^{\pi})\text{BR}(V_{k-1})$	$-(\mathbf{I}-\gamma\mathcal{P}^{\pi(Q_k)})\text{BR}^*(Q_{k-1})$
κ_d	$-(\mathbf{I}-\gamma\mathcal{P}^{\pi})(V_{k-1}-V_{k-2})$	$-(\mathbf{I}-\gamma\mathcal{P}^{\pi(Q_k)})(Q_{k-1}-Q_{k-2})$
κ_I	$-(\mathbf{I}-\gamma\mathcal{P}^{\pi})z_k$	$-(\mathbf{I}-\gamma\mathcal{P}^{\pi(Q_k)})z_k$
α	$-\kappa_I(\mathbf{I}-\gamma\mathcal{P}^{\pi})\text{BR}(V_{k-1})$	$-\kappa_I(\mathbf{I}-\gamma\mathcal{P}^{\pi(Q_k)})\text{BR}^*(Q_{k-1})$
β	$-\kappa_I(\mathbf{I}-\gamma\mathcal{P}^{\pi})z_{k-1}$	$-\kappa_I(\mathbf{I}-\gamma\mathcal{P}^{\pi(Q_k)})z_{k-1}$

Algorithm 1 PID-Accelerated Value Iteration

Initialize V_1 (e.g., equal to 0) and $z_1 = 0$.
Initialize $(\kappa_p^{(1:2)}, \kappa_I^{(1:2)}, \kappa_d^{(1:2)}) = (1, 0, 0)$.
for $k = 1, \dots, K$ **do**
  Compute $T^{\pi}V_k$
  Set $\text{BR}(V_k) = T^{\pi}V_k - V_k$.
  Update z and V by

$$z_{k+1} = \beta^{(k)} z_t + \alpha^{(k)} \text{BR}(V_k),$$
$$V_{k+1} = (1-\kappa_p^{(k)})V_k + \kappa_p^{(k)} T^{\pi}V_k + \kappa_I^{(k)} z_{k+1} + \kappa_d^{(k)}(V_k - V_{k-1}).$$

  if $k \geq 3$ **then**
    Update the controller gains by (21)
  end if
end for

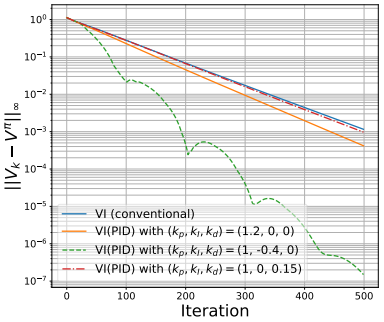
In these variants, the denominator is considered fixed at the k-th iteration, i.e., we do not compute its gradient. These normalized variants have an interesting interpretation. The ratio of two consecutive terms in a sequence is its rate of convergence (in the limit). So by considering the ratio of the squared Bellman errors, we are taking the gradient of the squared convergence rate w.r.t. the controller gains. With the normalized loss, the update rule for $\kappa_\cdot \in \{\kappa_p, \kappa_I, \kappa_d\}$ becomes

$$\kappa_\cdot(k+1) \leftarrow \kappa_\cdot(k) - \eta \frac{\left\langle \text{BR}(V_k), \frac{\partial \text{BR}(V_k)}{\partial \kappa_\cdot} \right\rangle_{\mathcal{X}}}{\|\text{BR}(V_{k-1})\|_2^2 + \varepsilon}, \quad (21)$$

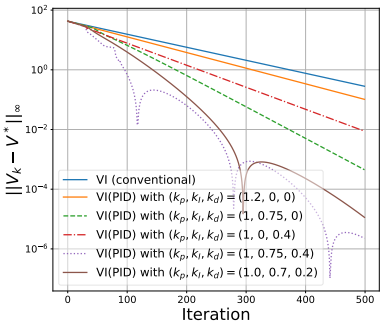
where $\eta > 0$ is the meta-learning rate, and $\varepsilon > 0$ is to avoid numerical instability when the Bellman error is too small. The new controller gains are used to compute V_{k+1} (or Q_{k+1} for the control case). This is summarized in Algorithm 1.

6. Experiments

We conduct two sets of experiments. In the first set, we observe the effect of choosing controller gains on the error. In the second set, we study the behaviour of the gain adaptation procedure. This section is only a summary of the experiments we conducted, and more comprehensive experiments can be found in Appendix I. The detailed description of the domains used in the experiments is available in Appendix H.



(a) Policy Evaluation (PE)



(b) Control

Figure 1. (Chain Walk) Sample error behaviour for a 50-state chain walk problem for various accelerated variants of VI and the conventional VI.

6.1. Experiments with Controller Gains

We use a chain walk problem with 50 states as a testbed, similar to Lagoudakis & Parr (2003). We consider both policy evaluation and control cases. For PE, we only show the results for a policy that always chooses the first action, i.e., $\pi(x) = a_1$. We set $\gamma = 0.99$ in these experiments.

In the first experiment, we showcase a typical behaviour of VI and accelerated VI with different controller gains. In particular, we show $\log_{10}\|V_k - V^{\pi}\|_\infty$ (PE) and $\log_{10}\|V_k - V^*\|_\infty$ (control) as a function of iteration k in Figure 1 (the result would be qualitatively similar for other norms). To compute the "true" V^{π} or Q^*, needed for the computation of the norms, we run the conventional VI (no acceleration) for 10-times more iterations. This results in the error of $O(\gamma^{5000}) \approx 1.5 \times 10^{-22}$. This would be sufficient if the effective discount factor γ' of the accelerated VI is larger than γ^{10}, e.g., ≈ 0.904 with our choice of γ. For the accelerated ones, the gains are shown in the legend of the figure. For the PI variant, we always use $\beta = 0.95$ and $\alpha = 1 - \beta = 0.05$. With the right choice of parameters, they significantly improve the convergence behaviour.

Figure 1a shows some sample behaviours for the PE problem. The gains for these controllers are selected to showcase a good performance in certain range, but they are not numerically optimized. We observe that all accelerated variants lead to faster convergence. The PI variant with $\kappa_I = -0.4$ is particularly noticeable as it leads to several orders of magnitude decrease in error after 500 iterations (from around 10^{-3} to $\approx 10^{-7}$). The improvement due to PD is insignificant. It is interesting to observe that the P variant improves the performance by having a larger than 1 gain of $\kappa_p = 1.2$. Figure 1b repeats the same experiment for the control case. Both PD and PI controllers significantly improve the performance. We also observe that having both D and I terms can improve upon the performance of either of them. We show two PID variants, one with $(\kappa_p, \kappa_I, \kappa_d) = (1, 0.75, 0.4)$ and the other with $(\kappa_p, \kappa_I, \kappa_d) = (1, 0.7, 0.2)$. Their performances are comparable, suggesting that the performance is not too sensitive to the choice of parameters.

Each curve in these figures is for a particular choice of gains. *What happens if we change the gains?* To study this, we fix all gains except one, and compute the norm of the error $\log_{10} \|V_k - V^\pi\|_2$ (or similar for the control case) as a function of the gain parameter at various iterations k. For the P controller, we change κ_p around 1, while setting $\kappa_I = \kappa_d = 0$ (recall that the conventional VI corresponds to $(\kappa_p, \kappa_I, \kappa_d) = (1, 0, 0)$). For the PD controller, we change κ_d in a range that includes both negative and positive values, while setting $\kappa_p = 1$ and $\kappa_I = 0$. For the PI controller, likewise, we change κ_I, while setting $\kappa_p = 1$ and $\kappa_d = 0$. These PI and PD controllers are special cases of more general PI and PD controllers as we set their κ_p equal to 1.

Figures 2a (P), 2b (PI), 2c (PD) present the results for the PE case. We observe the change of the error as a function of each gain. The influence of κ_I is more significant compared to κ_p and κ_d. In particular, the error curves for κ_d do not show much change as a function of its parameter, which suggests that the PD controller is not very suitable for this problem. We also note that the overall shape of each curve is similar at different iterations, but they are not exactly the same. In earlier iterations, the effect of smaller eigenvalues is relatively more significant than in the later iterations. As k grows, the behaviour of the error would be mostly determined by the dominant eigenvalue. We also remark that the behaviour is not always smooth. The dynamics might become unstable, and in that case, the error actually grows exponentially as k increases. The range chosen for this figure is such that we are on the cusp of becoming unstable. For example, for the PD variant, the dynamics is unstable for $\kappa_d \approx 0.28$.

Figures 2d (P), 2e (PI), 2f (PD) present the result for the control case. One noticeable difference compared to the PE case is that κ_d has a significant effect on the error, and it can lead to acceleration of VI. We also observe that the range of values for κ_I that leads to acceleration is different than the range for the PE case. Here, positive values of κ_I leads to acceleration, while in the PE case, negative values did. We remark in passing that we benefitted from these sweep studies to choose reasonable, but not necessarily optimal, values for Figure 1.

We report additional experiments in Appendix I.1. For example, we show how the simultaneous change of two gains affect the error behaviour (it can lead to even faster acceleration), and how the change of one gain relocates the eigenvalues.

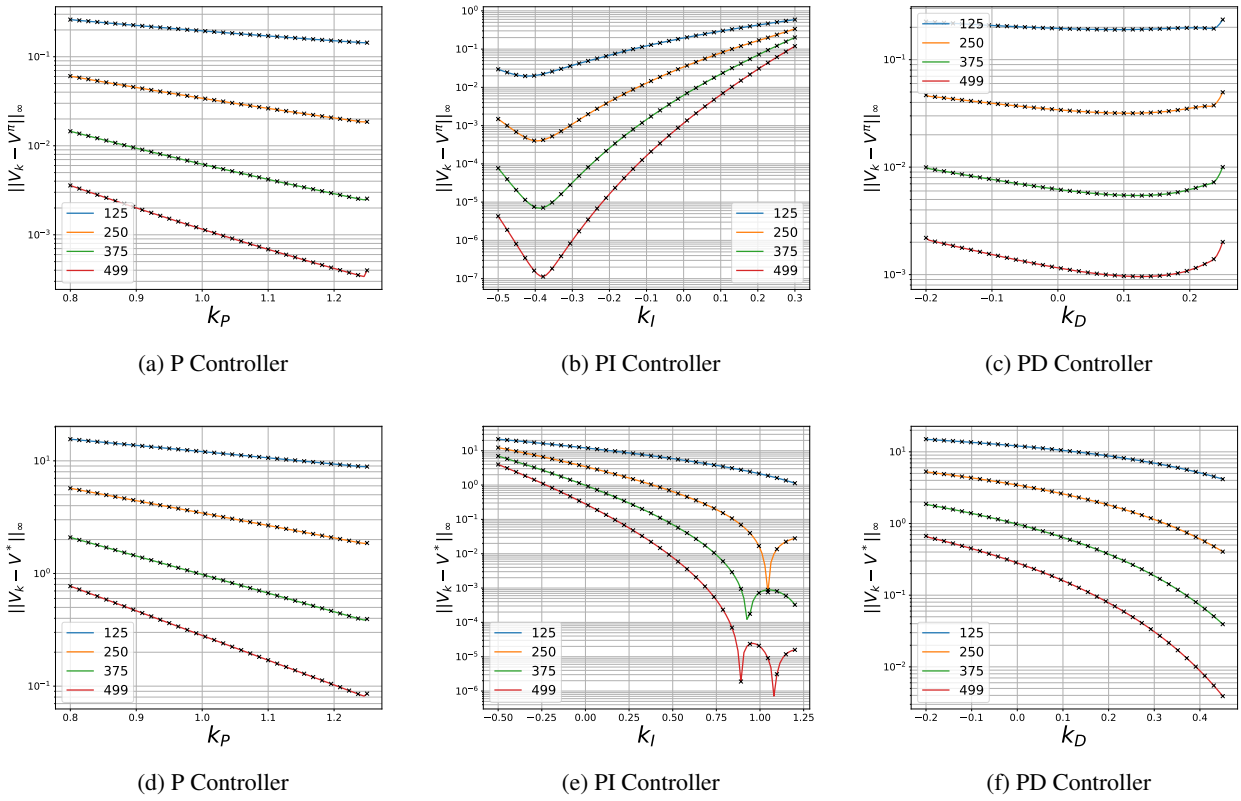
Two takeaway messages of these empirical results are: 1) we can accelerate VI, sometimes substantially, with the right choice of controller gains, and 2) the gains that lead to most acceleration is problem-dependent and varies between the PE and control cases.

6.2. Experiments with Gain Adaptation

In the second set of experiments, we study the behaviour of the gain adaptation procedure to see if it can lead to acceleration. We adapt κ_p, κ_I, and κ_d by (21), or similarly for the control case. We do not adapt α and β, and use fixed $\beta = 0.95$ and $\alpha = 1 - \beta = 0.05$ in all reported results. These results are for the discount factor $\gamma = 0.99$. We provide more comprehensive empirical studies in Appendix I.2.

Figure 3a compares the error of the accelerated PID VI with gain adaptation with the conventional VI on the Chain Walk problem (control case). Here we set the meta-learning parameter to $\eta = 0.05$ and normalizing factor to $\varepsilon = 10^{-20}$. We observe a significant acceleration. Figure 3b shows how the gains evolve as a function of the iteration number.

To study the behaviour of the gain adaptation procedure on a variety of MDPs, we also use Garnet problems, which are randomly generated MDPs (Bhatnagar et al., 2009). We consider the Garnet problem with 50 states, 4 actions, a branching factor of 3, and 5 non-zero rewards throughout the state space (see Appendix H for a detailed description). Figure 4 shows the average behaviour of the gain adaptation for different values of the meta-learning rate η (the normalizing constant is fixed to $\varepsilon = 10^{-20}$). For the PE case, we observe that all tested values of meta-learning rate η leads to acceleration, though the acceleration would be insignificant for very small values, e.g., $\eta = 10^{-3}$. For the control case, we observe that large values of meta-learning rate (e.g., $\eta = 0.1$) lead to non-convergence, while values smaller than that lead to significant acceleration. These results show that gain adaptation is a viable approach for achieving acceleration for an unknown MDP.



(a) P Controller (b) PI Controller (c) PD Controller

(d) P Controller (e) PI Controller (f) PD Controller

Figure 2. (Chain Walk) (Top: Policy Evaluation; Bottom: Control) Error norm behaviour $\log_{10} \|V_k(k.) - V^\pi\|_\infty$ (Policy Evaluation) and $\log_{10} \|Q_k(k.) - Q^*\|_\infty$ (Control) as one of the controller gains is changed. Each curve corresponds to a different iteration k. Crosses on the curves are placed at every 3 computed points in order to decrease the clutter.

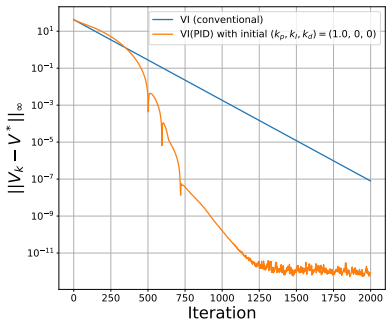
7. Related Work

There have been recent work on accelerating RL (Geist & Scherrer, 2018; Shi et al., 2019; Vieillard et al., 2020; Goyal & Grand-Clement, 2020). As opposed to this work, those approaches do not start by establishing connection between the planning methods commonly used to solve MDPs and the methods often used in control theory/engineering. Instead, some of them are inspired by methods in optimization, and some by other numerical techniques. Here, we only briefly mention some connections to these methods and provide an in-depth comparison in Appendix G.1.

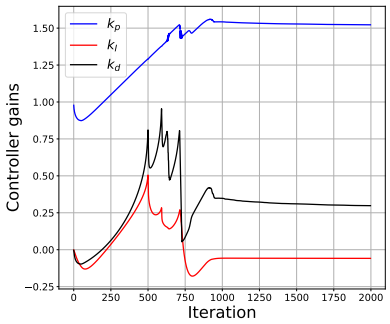
One class of these methods is based on borrowing ideas from the optimization theory to modify basic algorithms such as VI (Vieillard et al., 2020; Goyal & Grand-Clement, 2020). The work by Goyal & Grand-Clement (2020) is noticeable because some of their proposed methods happen to coincide with some of ours (Appendix G.1.1). By making an analogy between VI and the gradient descent (GD), they propose *Relaxed Value Iteration*, which is the same as P VI (7) and the Accelerated Jacobi method of Kushner & Kleinman (1971). By making an analogy between VI and the Polyak's momentum method (or heavy ball) (Polyak, 1987, Section 3.2), they propose a method called *Momentum Value Iteration/Computation*, which is essentially the PD VI method in (11). They suggest a specific choice of κ_p and κ_d based on the comparison of VI and the optimal choice of parameters in the momentum GD. This choice is suitable for *reversible* Markov chains, but it may diverge for more general chains that we deal with in solving MDPs. In fact, this is something that we observed in our experiments. Moreover, they do not provide any solution for cases other than reversible Markov chains, while we propose a *gain adaptation* mechanism and empirically show that it is a reasonable approach for VI acceleration in general MDPs. There is no method analogous to the PI or PID variants of VI in their work.

Geist & Scherrer (2018) and Shi et al. (2019) use acceleration techniques from other areas of numerical methods, such as Anderson acceleration (Anderson, 1965), to modify Value or Policy Iteration procedures (Appendix G.1.2).

There are other, less similar, approaches to acceleration in RL and DP (Appendix G.1.3). Prioritized sweeping and

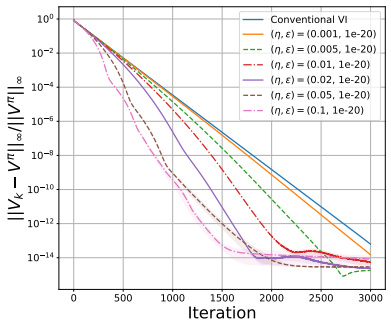


(a) Error behaviour

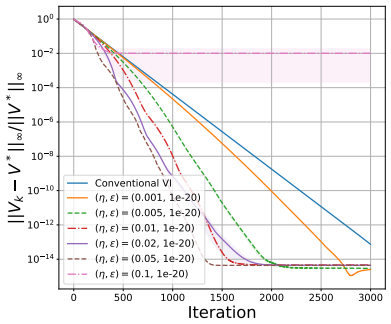


(b) Gains

Figure 3. (Chain Walk - Control) Gain adaptation results for $(\eta, \varepsilon) = (0.05, 10^{-20})$.



(a) Policy Evaluation



(b) Control

Figure 4. (Garnet) Gain adaptation for a 50-state Garnet problem with $\gamma = 0.99$ for the PE and control cases for different meta-learning rates η. The normalizing factor is $\varepsilon = 10^{-20}$. The mean and standard errors are evaluated based on 100 runs.

its variants are an important class of methods that asynchronously update the value of states (Moore & Atkeson, 1993; Peng & Williams, 1993; McMahan & Gordon, 2005; Wingate & Seppi, 2005). By changing the order of state updates, they might converge to the value function faster. It is, however, orthogonal to what we suggest, and they can potentially be combined. Speedy Q-Learning is an accelerated variant of Q-Learning (Azar et al., 2011). It decomposes the update rule of Q-Learning in a specific way and use a more aggressive learning rate on one of its terms. Zap Q-Learning is a second-order stochastic approximation method that uses a matrix gain, instead of a scalar one, to minimize the asymptotic variance of the value function estimate (Devraj & Meyn, 2017).

8. Conclusion

We viewed the value iteration (VI) procedure as a dynamical system and used tools from control theory to acceleration it. We specifically focused on simple, yet effective, PID controllers to modify VI. We expressed the error dynamics of the accelerated VI procedures for the policy evaluation problem as a linear dynamical system. We empirically showed that the modified VI can indeed lead to accelerated behaviour. Moreover, we proposed a gain adaptation procedure to automatically adjust the controller.

An important future research direction is extending the PID VI to the RL setting, where only samples are available. The sample-based extension seems feasible given that one can form an unbiased estimate of all key quantities in the PID VI updates using samples in the form of (X_t, R_t, X_{t+1}). For example, $R_t + \gamma V_k(X_{t+1}) + \kappa_d(V_k(X_t) - V_{k-1}(X_t))$ is an unbiased estimate of $T^\pi V_k + \kappa_d(V_k - V_{k-1})$ evaluated at X_t. Another exciting direction is designing controllers other than PID to accelerate fundamental DP algorithm.

Acknowledgements

We would like to thank the anonymous reviewers for their feedback. AMF acknowledges the funding from the Canada CIFAR AI Chairs program.

References

Almeida, L. B., Langlois, T., Amaral, J. D., and Plakhov, A. *Parameter Adaptation in Stochastic Optimization*, pp. 111–134. Publications of the Newton Institute. Cambridge University Press, 1999. 5, 33

Anderson, D. G. Iterative procedures for nonlinear integral equations. *Journal of the ACM*, 12(4):547–560, 1965. 8, 28, 31

Aström, K. J. and Wittenmark, B. *Adaptive Control*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 1994. 3, 27, 33

Azar, M., Munos, R., Ghavamzadeh, M., and Kappen, H. Speedy Q-learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 2411–2419, 2011. 9, 32

Baydin, A. G., Cornish, R., Rubio, D. M., Schmidt, M., and Wood, F. Online learning rate adaptation with hypergradient descent. In *International Conference on Learning Representations (ICLR)*, 2018. 5, 33

Berinde, V. *Iterative approximation of fixed points*, volume 1912. Springer, 2007. 3, 29

Bertsekas, D. P. and Shreve, S. E. *Stochastic Optimal Control: The Discrete-Time Case*. Academic Press, 1978. 3, 12

Bertsekas, D. P. and Tsitsiklis, J. N. *Neuro-Dynamic Programming*. Athena Scientific, 1996. 1, 2, 12, 32

Bhatnagar, S., Sutton, R. S., Ghavamzadeh, M., and Lee, M. Natural actor–critic algorithms. *Automatica*, 45(11): 2471–2482, 2009. 7, 34, 38

Burl, J. B. *Linear Optimal Control: H_2 and H_∞ Methods*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1998. 3

Chen, J. and Jiang, N. Information-theoretic considerations in batch reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019. 1

Devraj, A. M. and Meyn, S. P. Zap Q-learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 2235–2244, 2017. 9, 28, 32

Dorf, R. C. and Bishop, R. H. *Modern control systems*. Prentice Hall, 2008. ISBN 9780132270281. 3

Ernst, D., Geurts, P., and Wehenkel, L. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research (JMLR)*, 6:503–556, 2005. 1

Farahmand, A.-m., Ghavamzadeh, M., Szepesvári, Cs., and Mannor, S. Regularized fitted Q-iteration for planning in continuous-space Markovian Decision Problems. In *Proceedings of American Control Conference (ACC)*, pp. 725–730, June 2009. 1

Geist, M. and Scherrer, B. Anderson acceleration for reinforcement learning. In *European workshop on Reinforcement Learning (EWRL)*, 2018. 8, 28, 31

Goyal, V. and Grand-Clement, J. A first-order approach to accelerated value iteration. *arXiv:1905.09963v6*, March 2020. 8, 17, 19, 28, 29, 30, 33

Jungers, R. M. *The Joint Spectral Radius: Theory and Applications*. Springer-Verlag Berlin Heidelberg, 2009. 5, 16, 17

Khalil, H. K. *Nonlinear Systems (3rd Edition)*. Prentice Hall, 2001. 3, 16

Krstic, M., Kanellakopoulos, I., and Kokotovic, P. V. *Nonlinear and adaptive control design*. Wiley New York, 1995. 3

Kushner, H. J. and Kleinman, A. J. Accelerated procedures for the solution of discrete Markov control problems. *IEEE Transactions on Automatic Control*, 16(2):147–152, April 1971. 8, 28

Lagoudakis, M. G. and Parr, R. Least-squares policy iteration. *Journal of Machine Learning Research (JMLR)*, 4: 1107–1149, 2003. 6, 27, 32, 33

Mahmood, A. R., Sutton, R. S., Degris, T., and Pilarski, P. M. Tuning-free step-size adaptation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2121–2124, 2012. 5, 33

McMahan, H. B. and Gordon, G. J. Fast exact planning in Markov decision processes. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 2005. 9, 32

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, February 2015. 1

Moore, A. W. and Atkeson, C. G. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine learning*, 13(1):103–130, 1993. 9, 32

Munos, R. Performance bounds in L_p norm for approximate value iteration. *SIAM Journal on Control and Optimization*, pp. 541–561, 2007. 5, 21, 22

Munos, R. and Szepesvári, Cs. Finite-time bounds for fitted value iteration. *Journal of Machine Learning Research (JMLR)*, 9:815–857, 2008. 1

Ogata, K. *Modern Control Engineering*. Prentice hall Upper Saddle River, NJ, fifth edition, 2010. 3

Peng, J. and Williams, R. J. Efficient learning and planning within the Dyna framework. *Adaptive Behavior*, 1(4): 437–454, 1993. 9, 32

Polyak, B. T. *Introduction to optimization*. Optimization Software, Inc., 1987. 8, 29

Schraudolph. Local gain adaptation in stochastic gradient descent. In *International Conference on Artificial Neural Networks (ICANN)*, 1999. 5, 33

Shi, W., Song, S., Wu, H., Hsu, Y.-C., Wu, C., and Huang, G. Regularized Anderson acceleration for off-policy deep reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 10231–10241. 2019. 8, 28, 31

Skogestad, S. and Postlethwaite, I. *Multivariable Feedback Control: Analysis and Design*. Wiley New York, 2nd edition, 2005. 3

Sutton, R. S. Adapting bias by gradient descent: An incremental version of delta-bar-delta. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, 1992. 5, 33

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2019. 1, 2, 12

Szepesvári, Cs. *Algorithms for Reinforcement Learning*. Morgan Claypool Publishers, 2010. 2, 12

Tosatto, S., Pirotta, M., D'Eramo, C., and Restelli, M. Boosted fitted Q-iteration. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017. 1

Vieillard, N., Scherrer, B., Pietquin, O., and Geist, M. On momentum in reinforcement learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020. 8, 28, 30, 31

Williams, R. J. and Baird, L. C. Tight performance bounds on greedy policies based on imperfect value functions. Technical report, Northeastern University, 1993. 5, 20

Wingate, D. and Seppi, K. D. Prioritization methods for accelerating MDP solvers. *Journal of Machine Learning Research (JMLR)*, 6(29):851–881, 2005. 9, 32

Zhou, K. and Doyle, J. C. *Essentials of robust control*. Prentice hall Upper Saddle River, NJ, 1998. 3