PID Accelerated Value Iteration Algorithm

Amir-massoud Farahmand[(1)] Mohammad Ghavamzadeh[(2)]

(1) Vector Institute and University of Toronto, Canada
(2) Google Research, Mountain View, California, USA

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.

Contents

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 and Tsitsiklis, 1996, Sutton and Barto, 2019] and Fitted Value Iteration procedures [Ernst et al., 2005, Munos and Szepesvári, 2008, Farahmand et al., 2009, Mnih et al., 2015, Tosatto et al., 2017, Chen and 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. 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 and Tsitsiklis, 1996, Szepesvári, 2010, Sutton and 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

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

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 and Bishop [2008], Khalil [2001], Aström and Wittenmark [1994], Krstic et al. [1995], Burl [1998], Bertsekas and Shreve [1978], Zhou and Doyle [1998], Skogestad and 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 and 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

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

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:

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

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 $\mathrm{BR}(V_i)$, for $i \le k$. From the filtering theory perspective, it is an auto-regressive filter that performs low-pass filtering over the sequence of Bellman residuals (as long as $|\beta| < 1$). The additional

[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.

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. 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*

$$\begin{bmatrix} e_{k+1} \\ e_k \\ z_{k+1} \end{bmatrix} = \begin{bmatrix} (\mathbf{I} - K_p) + \gamma K_P \mathcal{P}^\pi + \alpha K_I(\gamma \mathcal{P}^\pi - \mathbf{I}) + K_d & -K_d & \beta K_I \\ \mathbf{I} & 0 & 0 \\ \alpha(\gamma \mathcal{P}^\pi - \mathbf{I}) & 0 & \beta \mathbf{I} \end{bmatrix} \begin{bmatrix} e_k \\ e_{k-1} \\ z_k \end{bmatrix} \triangleq A_{\text{PID}} \begin{bmatrix} e_k \\ e_{k-1} \\ z_k \end{bmatrix}. \tag{15}$$

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.

[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.

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 hyper-parameters 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

$$\|V - V^\pi\|_\infty \le \frac{\|T^\pi V - V\|_\infty}{1-\gamma}. \tag{16}$$

[4] More generally, we can unroll the accelerated algorithm for $T \ge 1$ 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.

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:

$$J_{\mathrm{BE}}(k) = \frac{1}{2}\left\|T^{\pi}V_k - V_k\right\|_2^2 = \frac{1}{2}\left\|\mathrm{BR}(V_k)\right\|_2^2, \tag{17}$$

$$J^*_{\mathrm{BE}}(k) = \frac{1}{2}\left\|T^{*}Q_k - Q_k\right\|_2^2 = \frac{1}{2}\left\|\mathrm{BR}^*(Q_k)\right\|_2^2. \tag{18}$$

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

$$\frac{\partial J_{\mathrm{BE}}(k)}{\partial \kappa_{\cdot}} = \left\langle \mathrm{BR}(V_k)\,,\, \frac{\partial \mathrm{BR}(V_k)}{\partial \kappa_{\cdot}} \right\rangle_{\mathcal{X}}, \tag{19}$$

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 $J^*_{\mathrm{BE}}(k)$ 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 (19). 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:

$$J_{\mathrm{BE(norm)}}(k) = \frac{\left\|\mathrm{BR}(V_k)\right\|_2^2}{2\left\|\mathrm{BR}(V_{k-1})\right\|_2^2}, \tag{20}$$

$$J^*_{\mathrm{BE(norm)}}(k) = \frac{\left\|\mathrm{BR}^*(Q_k)\right\|_2^2}{2\left\|\mathrm{BR}^*(Q_{k-1})\right\|_2^2}. \tag{21}$$

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

$$\kappa_{\cdot}(k+1) \leftarrow \kappa_{\cdot}(k) - \eta \frac{\left\langle \mathrm{BR}(V_k)\,,\, \frac{\partial \mathrm{BR}(V_k)}{\partial \kappa_{\cdot}} \right\rangle_{\mathcal{X}}}{\left\|\mathrm{BR}(V_{k-1})\right\|_2^2 + \varepsilon}, \tag{22}$$

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.

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., $\pi(x;Q) \leftarrow \mathrm{argmax}_{a\in\mathcal{A}} Q(x,a)$.

	$\frac{\partial \mathrm{BR}(V_k)}{\partial \cdot}$	$\frac{\partial \mathrm{BR}^*(Q_k)}{\partial \cdot}$
κ_p	$-(\mathbf{I}-\gamma\mathcal{P}^\pi)\mathrm{BR}(V_{k-1})$	$-(\mathbf{I}-\gamma\mathcal{P}^{\pi(Q_k)})\mathrm{BR}^*(Q_{k-1})$
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	$-\kappa_I(\mathbf{I}-\gamma\mathcal{P}^\pi)\mathrm{BR}(V_{k-1})$	$-\kappa_I(\mathbf{I}-\gamma\mathcal{P}^{\pi(Q_k)})\mathrm{BR}^*(Q_{k-1})$
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 $k = 1, \ldots, K$ do
    Compute $T^\pi V_k$
    Set $\mathrm{BR}(V_k) = T^\pi V_k - V_k$.
    Update $z$ and $V$ by
        $z_{k+1} = \beta^{(k)} z_t + \alpha^{(k)} \mathrm{BR}(V_k)$,
        $V_{k+1} = (1 - \kappa_p^{(k)}) V_k + \kappa_p^{(k)} T^\pi V_k + \kappa_I^{(k)} z_{k+1} + \kappa_d^{(k)} (V_k - V_{k-1})$.
    if $k \geq 3$ then
        Update the controller gains by (22)
    end if
end for
```

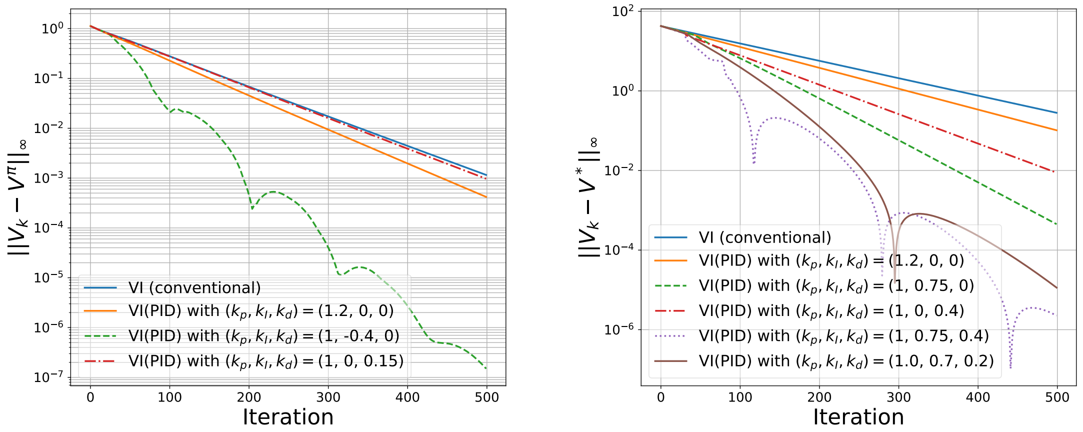
6.1 Experiments with Controller Gains

We use a chain walk problem with 50 states as a testbed, similar to Lagoudakis and 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



(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.

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 $\kappa_d \approx 0.28$.

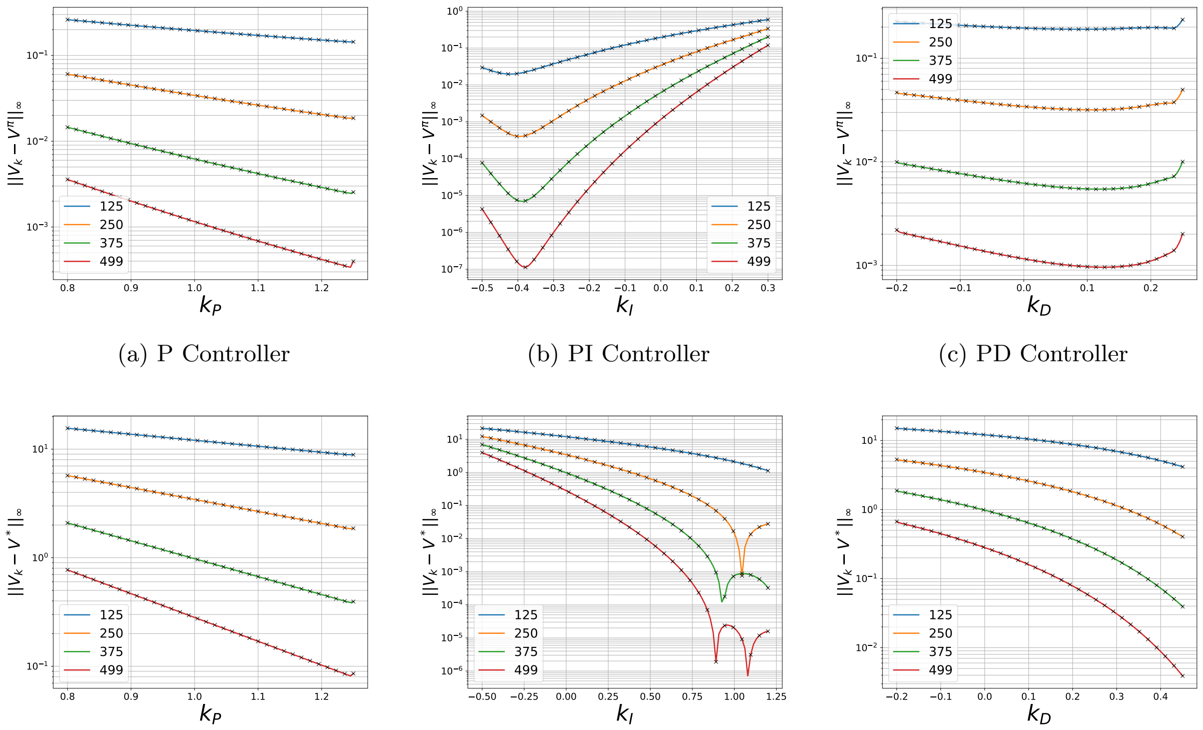
Figures 2d (P), 2e (PI), 2f (PD) present the result for the control case. One noticeable difference compared to the PE case is that κ_d has a significant effect on the error, and it can lead to acceleration of VI. We also observe that the range of values for κ_I that leads to acceleration is different 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 (22), 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



(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.

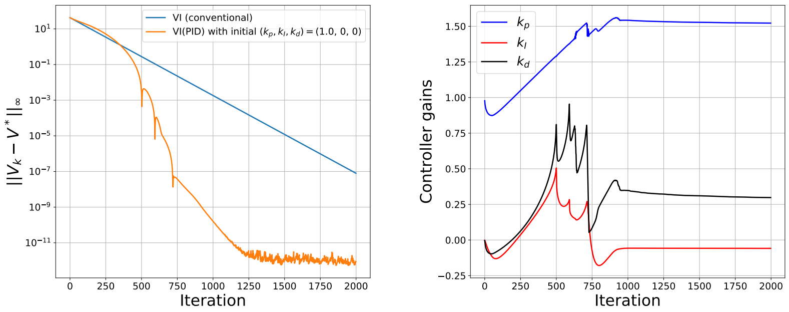
factor $\gamma = 0.99$. We provide more comprehensive empirical studies in Appendix I.2.

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.

7 Related Work

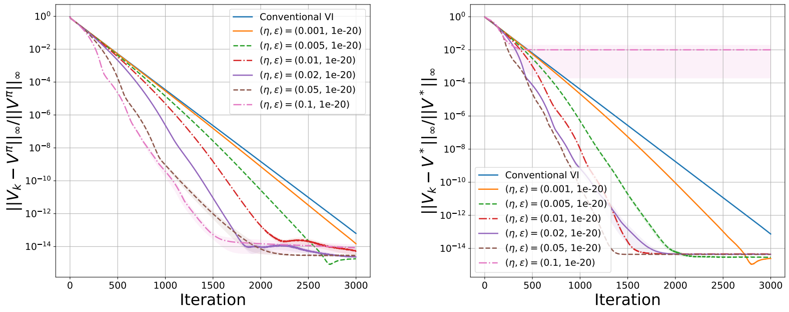
There have been recent work on accelerating RL [Geist and Scherrer, 2018, Shi et al., 2019, Vieillard et al., 2020b, Goyal and Grand-Clement, 2020]. As opposed to this work, those approaches do not start by establishing connection between the planning methods commonly used to solve MDPs and the methods often



(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.

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., 2020b, Goyal and Grand-Clement, 2020]. The work by Goyal and 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 and 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 and 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 its variants are an important class of methods that asynchronously update the value of states [Moore and Atkeson, 1993, Peng and Williams, 1993, McMahan and Gordon, 2005, Wingate and 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 and 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.

A Markov Decision Processes

We consider a discounted Markov Decision Process (MDP) $(\mathcal{X}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma)$ [Szepesvári, 2010, Bertsekas and Shreve, 1978, Bertsekas and Tsitsiklis, 1996, Sutton and Barto, 2019]. Our notation would be most similar to Szepesvári [2010]. Here $\mathcal{X}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{R} : \mathcal{X} \times \mathcal{A} \to \mathcal{M}(\mathbb{R})$ is the reward distribution, $\mathcal{P} : \mathcal{X} \times \mathcal{A} \to \mathcal{M}(\mathcal{X})$ is the transition probability kernel, and $0 \leq \gamma < 1$ is the discount factor.

For a set Ω, the space of bounded functions is denoted by $\mathcal{B}(\Omega)$, and the space of probability distributions is denoted by $\mathcal{M}(\Omega)$. Here we do not go into the measurability issues, so we omit the detail of the necessary σ-algebra. The policy $\pi : \mathcal{X} \to \mathcal{M}(\mathcal{A})$ (stochastic policy) or $\pi : \mathcal{X} \to \mathcal{A}$ (deterministic) is a Markov stationary policy. Given a policy, we can define $\mathcal{P}^\pi$ as the transition probability kernel of following π, and it would be

$$\mathcal{P}^\pi(\cdot|x) = \int \mathcal{P}(\cdot|x, a)\pi(\mathrm{d}a|x).$$

We can define $\mathcal{P}^{\pi^{(m)}} : \mathcal{X} \to \mathcal{M}(\mathcal{X})$ for $m \geq 0$ recursively. For $m = 0$, we use the convention that it is equal to $\mathbf{I}$, the identity operator (or matrix). For $m \geq 1$, we have

$$\mathcal{P}^{\pi^{(m)}}(\cdot|x) = \int \mathcal{P}^\pi(\mathrm{d}y|x)\mathcal{P}^{\pi^{(m-1)}}(\cdot|y).$$

We may also define $\mathcal{P}^{\pi^{(m)}} : \mathcal{X} \times \mathcal{A} \to \mathcal{M}(\mathcal{X})$, as the transition probability kernel of choosing action a at state x, following $\mathcal{P}(\cdot|x, a)$ for one step, and afterwards, following policy π for the remaining $m - 1$ steps. Formally, for $m = 1$, $\mathcal{P}^{\pi^{(1)}} = \mathcal{P}^\pi = \mathcal{P}$. For $m \geq 2$, we have

$$\mathcal{P}^{\pi^{(m)}}(\cdot|x, a) = \int \mathcal{P}(\mathrm{d}y|x, a)\mathcal{P}^{\pi^{(m-1)}}(\cdot|y).$$

We can define $\mathcal{R}^\pi : \mathcal{X} \to \mathcal{M}(\mathbb{R})$ in a similar fashion. The functions $r : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ and $r^\pi : \mathcal{X} \to \mathbb{R}$ are the expected value of the reward distribution.

We use V^π and Q^π to denote the state-value and action-value functions for a policy π. We use V^* and Q^* to denote the optimal value and action-value functions.

The Bellman operator $T^\pi : \mathcal{B}(\mathcal{X}) \to \mathcal{B}(\mathcal{X})$ for policy π and the Bellman optimality operator $T^* : \mathcal{B}(\mathcal{X} \times \mathcal{A}) \to \mathcal{B}(\mathcal{X} \times \mathcal{A})$ are defined as

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

For countable state and action spaces, the integrals are replaced by summations. The Bellman operators $T^\pi : \mathcal{B}(\mathcal{X} \times \mathcal{A}) \to \mathcal{B}(\mathcal{X} \times \mathcal{A})$ (applying on an action-value function) and $T^* : \mathcal{B}(\mathcal{X}) \to \mathcal{B}(\mathcal{X})$ (applying on the state-value function) are defined similarly. We do not use them in the paper, so we do not explicitly define here.

We denote $\pi(\cdot; Q)$ as the greedy policy w.r.t. Q, i.e., at each state x, we have

$$
\pi(x; Q) \leftarrow \operatorname*{argmax}_{a \in \mathcal{A}} Q(x,a). \tag{23}
$$

B Error Dynamics: Proofs and Simplified Results

We prove Proposition 2, which describes the error dynamics of the PD, PI, and PID VI. Proposition 1 from Section 4 is its special case for the PID VI case. After the proof, we provide some simplified, and perhaps more intuitive, results for the error dynamics and the location of the modified eigenvalues. Corollary 3 shows the dynamics for a diagonalizable $\mathcal{P}^\pi$. Proposition 4 presents the location of the roots of the PID variant as a solution to cubic equations. The roots depends on the eigenvalues of $\mathcal{P}^\pi$ and the controller gains. Corollary 5 is a similar result specialized for the PD and PI controllers. In those two cases, the eigenvalues are the solution of quadratic equations, and have relatively simple solutions.

Proposition 2 (Error Dynamics of PD, PI, and PID Value Iteration)**.** *Let $e_k = V_k - V^\pi$. Define $\bar{e}_k = [e_k, e_{k-1}]^\top$. The error dynamics of the **PD controller** with a matrix gain K_d and a scalar κ_p (11) is*

$$
\bar{e}_{k+1} = \begin{bmatrix} (1-\kappa_p)\mathbf{I} + \gamma \kappa_p \mathcal{P}^\pi + K_d & -K_d \\ \mathbf{I} & 0 \end{bmatrix} \bar{e}_k \triangleq A_{\mathrm{PD}}\, \bar{e}_k. \tag{24}
$$

*The dynamics of the error e_k and the integrator's state z of the **PI controller** with a matrix gain K_I and the scalar gain $k_p = 1$ is*

$$
\begin{aligned}
\begin{bmatrix} e_{k+1} \\ z_{k+1} \end{bmatrix} &= \begin{bmatrix} \gamma \mathcal{P}^\pi + \alpha K_I (\gamma \mathcal{P}^\pi - \mathbf{I}) & \beta K_I \\ \alpha(\gamma \mathcal{P}^\pi - \mathbf{I}) & \beta \mathbf{I} \end{bmatrix} \begin{bmatrix} e_k \\ z_k \end{bmatrix} \\
&\triangleq A_{\mathrm{PI}} \begin{bmatrix} e_k \\ z_k \end{bmatrix}.
\end{aligned} \tag{25}
$$

*Finally, the dynamics of the **PID controller** with gains K_p, K_d, K_I is*

$$
\begin{bmatrix} e_{k+1} \\ e_k \\ z_{k+1} \end{bmatrix} = \begin{bmatrix} (\mathbf{I} - K_p) + \gamma K_P \mathcal{P}^\pi + \alpha K_I (\gamma \mathcal{P}^\pi - \mathbf{I}) + K_d & -K_d & \beta K_I \\ \mathbf{I} & 0 & 0 \\ \alpha(\gamma \mathcal{P}^\pi - \mathbf{I}) & 0 & \beta \mathbf{I} \end{bmatrix} \begin{bmatrix} e_k \\ e_{k-1} \\ z_k \end{bmatrix} \triangleq A_{\mathrm{PID}} \begin{bmatrix} e_k \\ e_{k-1} \\ z_k \end{bmatrix}. \tag{26}
$$

Proof of Proposition 2. For the **PD controller**, we subtract V^π from both sides of (11), benefit from $V^\pi = T^\pi V^\pi$, and simplify, to get

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

Reorganizing this in the matrix form gives the desired result.

The error dynamics for the **PI variant** is obtained similarly. By subtracting V^π from both sides of the dynamics of V_k in (12), we get

$$\begin{aligned} e_{k+1} = V_{k+1} - V^\pi &= T^\pi V_k - V^\pi + K_I \left[\beta z_k + \alpha(T^\pi V_k - V^\pi + V^\pi - V_k)\right] \\ &= T^\pi V_k - T^\pi V^\pi + K_I \left[\beta z_k + \alpha(T^\pi V_k - T^\pi V^\pi - V_k + V^\pi)\right] \\ &= \gamma \mathcal{P}^\pi e_k + K_I \left[\beta z_k + \alpha(\gamma \mathcal{P}^\pi - \mathbf{I})e_k\right]. \end{aligned}$$

The term $\text{BR}(V_k)$ in the dynamics of z_k can be written as $(\gamma\mathcal{P}^\pi - \mathbf{I})e_k$ as shown in (8). Reorganizing these two equations in the matrix form gives the desired result.

Finally, the error dynamics of the **PID variant** is

$$\begin{aligned} e_{k+1} = V_{k+1} - V^\pi = &(1 - K_p)(V_k - V^\pi) + K_p(T^\pi V_k - V^\pi) + \\ &K_I \left[\beta z_k + \alpha(T^\pi V_k - V^\pi + V^\pi - V_k)\right] + \\ &K_d \left((V_k - V^\pi) - (V_{k-1} - V^\pi)\right) \\ = &(1 - K_p)e_k + K_p \gamma \mathcal{P}^\pi e_k + K_I \left[\beta z_k + \alpha(\gamma\mathcal{P}^\pi - \mathbf{I})e_k\right] + K_d(e_k - e_{k-1}). \end{aligned}$$

Here we used similar substitutions. Considering the dynamics of z_k, as in the PI case, this leads to the desired result. □

Proposition 2 can be presented in a simpler and more intuitive form, if we only consider scalar gains and assume that $\mathcal{P}^\pi$ is diagonalizable, i.e., $\mathcal{P}^\pi = S\Lambda S^{-1}$ with $\Lambda = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_d)$. The following corollary shows the result.

Corollary 3. *Assume that $\mathcal{P}^\pi$ is diagonalizable in complex field, that is, $\mathcal{P}^\pi = S\Lambda S^{-1}$ with $S \in \mathbb{C}^{d \times d}$ and $\Lambda = diag(\lambda_1, \lambda_2, \dots, \lambda_d)$ with $\lambda_i \in \mathbb{C}$, for $i = 1, \dots, d$. Let $e_k = V_k - V^\pi$. Define $\varepsilon_k = S^{-1}e_k$ and $\bar{\varepsilon}_k = [\varepsilon_k, \varepsilon_{k-1}]^\top$. Consider scalar gains $K_p = \kappa_p \mathbf{I}$, $K_d = \kappa_d \mathbf{I}$, and $K_I = \kappa_I \mathbf{I}$. The error dynamics of the **PD controller** is*

$$\bar{\varepsilon}_{k+1} = \begin{bmatrix} (1 - \kappa_p + \kappa_d)\mathbf{I} + \gamma\kappa_p\Lambda & -\kappa_d\mathbf{I} \\ \mathbf{I} & 0 \end{bmatrix} \bar{\varepsilon}_k. \tag{27}$$

*For the PI variant, define $\zeta_k = S^{-1}z_k$. The error dynamics of the error and ζ of the **PI variant** with $\kappa_p = 1$ is*

$$\begin{bmatrix} \varepsilon_{k+1} \\ \zeta_{k+1} \end{bmatrix} = \begin{bmatrix} \gamma\Lambda + \alpha\kappa_I(\gamma\Lambda - \mathbf{I}) & \beta\kappa_I\mathbf{I} \\ \alpha(\gamma\Lambda - \mathbf{I}) & \beta\mathbf{I} \end{bmatrix} \begin{bmatrix} \varepsilon_k \\ \zeta_k \end{bmatrix}. \tag{28}$$

*For the **PID variant**, the error dynamics is*

$$\begin{bmatrix} \varepsilon_{k+1} \\ \varepsilon_k \\ \zeta_{k+1} \end{bmatrix} = \begin{bmatrix} (1 - \kappa_p - \alpha\kappa_I + \kappa_d)\mathbf{I} + \gamma(\kappa_p + \alpha\kappa_I)\Lambda & -\kappa_d\mathbf{I} & \beta\kappa_I\mathbf{I} \\ \mathbf{I} & 0 & 0 \\ \alpha(\gamma\Lambda - \mathbf{I}) & 0 & \beta\mathbf{I} \end{bmatrix} \begin{bmatrix} \varepsilon_k \\ \varepsilon_{k-1} \\ \zeta_k \end{bmatrix}. \tag{29}$$

Proof. We obtain the result by some simple algebraic manipulations. First consider the dynamics of the **PD control** (24), and particularly the equation described by its first row. Multiplying both sides of $e_{k+1} = \gamma\mathcal{P}^\pi e_k + \kappa_d(e_k - e_{k-1})$ from left by S^{-1}, and substituting $\mathcal{P}^\pi$ with $S\Lambda S^{-1}$, we get

$$\begin{aligned} \underbrace{S^{-1}e_{k+1}}_{=\varepsilon_{k+1}} &= (1 - \kappa_p)S^{-1}e_k + \gamma\kappa_p S^{-1}(S\Lambda S^{-1})e_k + \kappa_d S^{-1}(e_k - e_{k-1}) \\ &= [(1 - \kappa_p)\mathbf{I} + \gamma\kappa_p\Lambda]\varepsilon_k + \kappa_d(\varepsilon_k - \varepsilon_{k-1}). \end{aligned}$$

For the **PI variant** (25), we multiply both sides by S^{-1}. The dynamics of ε_k is

$$\begin{aligned} \varepsilon_{k+1} = S^{-1}e_{k+1} &= \gamma S^{-1}(S\Lambda S^{-1})e_k + S^{-1}\kappa_I \left[\beta z_t + \alpha(\gamma S^{-1}\Lambda S^{-1} - SS^{-1})e_k\right] \\ &= \gamma\Lambda\varepsilon_k + \kappa_I \left[\beta\zeta_t + \alpha(\gamma\Lambda - \mathbf{I})\varepsilon_k\right]. \end{aligned}$$

The dynamics of ζ_k can be related to the dynamics of z_k by multiplying S^{-1} to both sides of $z_{k+1} = \beta z_k + \alpha(\gamma\mathcal{P}^\pi - \mathbf{I})e_k$ and noticing that $e = S\varepsilon$, as

$$\begin{aligned}\zeta_{k+1} = S^{-1}z_{k+1} &= \beta S^{-1}z_k + \alpha S^{-1}\left(\gamma(S\Lambda S^{-1}) - SS^{-1}\right)S\varepsilon_k \\ &= \beta\zeta_k + \alpha\left(\gamma\Lambda - \mathbf{I}\right)\varepsilon_k.\end{aligned}$$

The dynamics of ε_k of the **PID variant** is obtained by following similar derivations as

$$\begin{aligned}\varepsilon_{k+1} = S^{-1}e_{k+1} &= \left[(1-\kappa_p)S^{-1} + \gamma\kappa_p S^{-1}(S\Lambda S^{-1}) + \alpha\kappa_I S^{-1}(\gamma S\Lambda S^{-1} - SS^{-1})\right]S\varepsilon_k \\ &\quad - \kappa_d S^{-1}S\varepsilon_{k-1} + \beta\kappa_I S^{-1}S\zeta_k \\ &= \left[(1-\kappa_p)\mathbf{I} + \gamma\kappa_p\Lambda + \alpha\kappa_I(\gamma\Lambda - \mathbf{I}) + \kappa_d\mathbf{I}\right]\varepsilon_k - \kappa_d\varepsilon_{k-1} + \beta\kappa_I\zeta_k.\end{aligned}$$

□

The behaviour of the error dynamics of these modified procedures depends on the location of the eigenvalues. The dominant behaviour depends on the eigenvalue with the largest magnitude. We can benefit from the assumption of diagonalizability of $\mathcal{P}^\pi$ to obtain relatively simple formula for eigenvalues.

Proposition 4. *Assume that $\mathcal{P}^\pi$ is diagonalizable in complex field, i.e., $\mathcal{P}^\pi = S\Lambda S^{-1}$ with $S \in \mathbb{C}^{d\times d}$ and $\Lambda = diag(\lambda_1, \lambda_2, \ldots, \lambda_d)$ with $\lambda_i \in \mathbb{C}$, for $i = 1, \ldots, d$. The eigenvalues of the error dynamics (29) of the **PID variant** are located at the roots of the following polynomial:*

$$f(\mu) = \prod_{i=1}^{d}\left[\mu^3 - \left((1+\kappa_d+\beta) - (\kappa_p + \alpha\kappa_I)(1-\gamma\lambda_i)\right)\mu^2 + \left(\beta(1+\kappa_d) + \kappa_d - \beta\kappa_p(1-\gamma\lambda_i)\right)\mu - \kappa_d\beta\right]. \tag{30}$$

Proof. Let us consider the dynamics matrix in (29). We permute the ordering of variables to $(\varepsilon_{k+1}, \zeta_{k+1}, \varepsilon_k)$ in order to make the calculations easier. The modified matrix is

$$F = \begin{bmatrix} (1-\kappa_p-\alpha\kappa_I+\kappa_d)\mathbf{I} + \gamma(\kappa_p+\alpha\kappa_I)\Lambda & \beta\kappa_I\mathbf{I} & -\kappa_d\mathbf{I} \\ \alpha(\gamma\Lambda - \mathbf{I}) & \beta\mathbf{I} & 0 \\ \mathbf{I} & 0 & 0\end{bmatrix}.$$

The eigenvalues are the roots of the characteristic polynomial

$$\det(\mu\mathbf{I}_{3d\times 3d} - F).$$

Recall that if D is invertible, then

$$\det\begin{bmatrix} A & B \\ C & D\end{bmatrix} = \det(D)\det(A - BD^{-1}C). \tag{31}$$

Using this equality twice, we get that for $\mu \neq \{0, \beta\}$,

$$\begin{aligned}\det(\mu\mathbf{I}_{3d\times 3d} - F) &= \det\left(\begin{array}{cc|c} \mu\mathbf{I} - F_{11} & -\beta\kappa_I\mathbf{I} & \kappa_d\mathbf{I} \\ -\alpha(\gamma\Lambda - \mathbf{I}) & (\mu-\beta)\mathbf{I} & 0 \\ \hline -\mathbf{I} & 0 & \mu\mathbf{I}\end{array}\right) \\ &= \det(\mu\mathbf{I})\det\left(\begin{array}{c|c} \mu\mathbf{I} - F_{11} + \frac{\kappa_d}{\mu}\mathbf{I} & -\beta\kappa_I\mathbf{I} \\ \hline \alpha(\mathbf{I} - \gamma\Lambda) & (\mu-\beta)\mathbf{I}\end{array}\right) \\ &= \det(\mu\mathbf{I})\det((\mu-\beta)\mathbf{I}))\det\left(\mu\mathbf{I} - F_{11} + \frac{\kappa_d}{\mu}\mathbf{I} - \frac{(-\beta\kappa_I\mathbf{I})(\alpha(\mathbf{I} - \gamma\Lambda))}{\mu - \beta}\right) \\ &= \det\left(\mu^3\mathbf{I} - \mu^2[\beta\mathbf{I} + F_{11}] + \mu[\beta F_{11} + \kappa_d\mathbf{I} + \alpha\beta\kappa_I(\mathbf{I} - \gamma\Lambda)] - \kappa_d\beta\mathbf{I}\right) \\ &= \det\Big(\mu^3\mathbf{I} - \mu^2\left[(1+\kappa_d+\beta)\mathbf{I} - (\kappa_p + \alpha\kappa_I)(\mathbf{I} - \gamma\Lambda)\right] + \\ &\qquad \mu\left[\beta\left((1+\kappa_d)\mathbf{I} - \beta\kappa_p(\mathbf{I} - \gamma\Lambda)\right) + \kappa_d\mathbf{I}\right] - \kappa_d\beta\mathbf{I}\Big).\end{aligned}$$

As all the involved matrices are diagonal, the characteristic polynomial is the multiplication of the diagonal terms, i.e.,

$$\prod_{i=1}^{d} \left[\mu^3 - \mu^2 \left((1+\kappa_d+\beta) - (\kappa_p + \alpha\kappa_I)(1-\gamma\lambda_i)\right) + \mu\left(\beta(1+\kappa_d) + \kappa_d - \beta\kappa_p(1-\gamma\lambda_i)\right) - \kappa_d\beta\right].$$

□

Corollary 5. *Consider the same setup as in Proposition 4. The eigenvalues of the error dynamics (27) of the **PD variant** are located at the roots of the following characteristic polynomial:*

$$f(\mu) = \prod_{i=1}^{d} \left[\mu^2 - (1+\kappa_d - \kappa_p(1-\gamma\lambda_i))\mu + \kappa_d\right], \tag{32}$$

which would be $\frac{(1+\kappa_d-\kappa_p(1-\gamma\lambda_i))\pm\sqrt{(1+\kappa_d-\kappa_p(1-\gamma\lambda_i))^2-4\kappa_d}}{2}$*, for* $i = 1, \dots, d$*.*

*The eigenvalues of the error dynamics (28) of the **PI variant** are located at the roots of the following characteristic polynomial:*

$$f(\mu) = \prod_{i=1}^{d} \left[\mu^2 - \left((1+\beta) - (1+\alpha\kappa_I)(1-\gamma\lambda_i)\right)\mu + \beta\gamma\lambda_i\right],$$

which would be $\frac{s\pm\sqrt{s^2-4\beta\gamma\lambda_i}}{2}$*, for* $i = 1, \dots, d$*, with* $s = (1+\beta) - (1+\alpha\kappa_I)(1-\gamma\lambda_i)$*.*

Proof. We can prove the claims by following a similar calculations as in the proof Proposition 4. Or instead, we may set $\kappa_I = 0$, $\alpha = \beta = 0$, and $\kappa_p = 1$ in (30) to obtain the characteristic polynomial of the PD variant, and set $\kappa_d = 0$ and $\kappa_p = 1$ in (30) to obtain the characteristic polynomial of the PI variant. This approach, however, requires some caution. The PID variant has $3d$ variables, and hence eigenvalues, but the PD and PI variants have $2d$ variables and eigenvalues. We need to remove the extra zero eigenvalues resulting from having an extra dynamics (I for PD, D for PI). □

C Error Dynamics for Control

The analysis of error dynamics described in Section 4 and Appendix B was for the PE problem where the policy π is fixed. It resulted in linear time-invariant (LTI) dynamical systems. For an LTI system, the locations of eigenvalues of the matrices A_{PD}, A_{PI}, and A_{PID} determine the behaviour of the dynamics. The spectral radius of those matrices determines the dominant behaviour. If $\rho(A) < 1$, the dynamics is asymptotically stable.

For the control problem, where we use T^* instead of T^π, the dynamical system is not a linear time-invariant system anymore, but is a linear *time-variant* one instead. To see this, notice that at the k-th iteration, the function $T^* Q_k$ appearing in the dynamics of the PID variant (14) for the control case is the same as $T^{\pi_k} Q_k$, with $\pi_k(x) = \pi(x; Q_k) = \operatorname{argmax}_{a\in\mathcal{A}} Q_k(x, a)$ being the greedy policy w.r.t. Q_k, see (23). This is also the same for the PI or PD variants for the control case. Therefore, denoting $\text{BR}^\pi(Q) = T^\pi Q - Q$ for a π, the dynamics of PID variant (14) can also be written as

$$\begin{aligned} z_{k+1} &= \beta z_k + \alpha \text{BR}^{\pi_k}(Q_k), \\ Q_{k+1} &= (1-K_p)Q_k + K_p T^{\pi_k} Q_k + K_I\left[\beta z_k + \alpha\text{BR}^{\pi_k}(Q_k)\right] + K_d(Q_k - Q_{k-1}). \end{aligned} \tag{33}$$

The PD and PI variants for the control case are similar too.

With this observation, the dynamics for the control case is described essentially the same as what we have in Proposition 1 in Section 4, and Proposition 2 and Corollary 3 in Appendix B. The difference would be that the dynamics matrix A would be a function of π_k, e.g., the dynamics of the PID variant is

$$\begin{bmatrix} e_{k+1} \\ e_k \\ z_{k+1} \end{bmatrix} = A_{\text{PID}}(\pi_k) \begin{bmatrix} e_k \\ e_{k-1} \\ z_k \end{bmatrix}. \tag{34}$$

This is a linear time-variant dynamical system. For such a system, the location of eigenvalues is not enough to establish the stability of the dynamics. It is possible that all eigenvalues are within the stability region of a linear time-invariant system (within unit circle for discrete-time dynamics, and negative half-plane for continuous-time dynamics), yet the dynamics be unstable, e.g., see Example 4.22 of Khalil [2001] for a continuous-time linear time-variant dynamical system and Chapter 1 of Jungers [2009] for a discrete-time linear time-variant dynamical system.

One way to study the stability of this system is by considering it as a switched linear dynamical system and its connection to the concept of *joint spectral radius* introduced by Rota and Strang [1960]; see also Jungers [2009] for a comprehensive reference. A switched linear dynamical system is

$$\begin{aligned} & y_{k+1} = A_k y_k, \qquad A_k \in \Sigma, \\ & y_0 \in \mathbb{R}^d, \end{aligned} \tag{35}$$

where Σ is a set of $d \times d$ matrices. The state of this dynamical system at time k is $y_k = A_{k-1} \cdots A_0 y_0$. The temporal evolution of $(y_k)_{k \geq 0}$ depends on what particular A_is are selected at each time step i. The (asymptotic) stability of this dynamical system is defined as whether for any bounded y_0 and any sequence of matrices $(A_k)_k$, y_k goes to zero as $k \to \infty$. The joint spectral radius, which we shall define soon, characterizes the maximum growth of this system for any possible choice of $A_i \in \Sigma$, and can be used as a criteria for its stability.

Let us first focus on when Σ only has one element (i.e, $\Sigma = \{A\}$), which means that we have a linear time-invariant system. We have $y_k = A^k y_0$. The norm of y_k is $\|y_k\| \leq \left\|A^k\right\| \|y_0\|$, where $\left\|A^k\right\|$ is the vector-induced matrix norm. Recall that the spectral radius of a matrix A is

$$\rho(A) = \lim_{k \to \infty} \left\|A^k\right\|^{1/k}.$$

Therefore, $\lim_{k \to \infty} \|y_k\| \leq \lim_{k \to \infty} \left\|A^k\right\| \|y_0\| = \lim_{k \to \infty} \rho(A)^k \|y_0\|$. If $\rho(A) < 1$, $\|y_k\|$ goes to zero as k grows, which means that $y_k \to 0$, i.e., the dynamics is stable.

The joint spectral radius generalizes this concept to when Σ possibly has more than one member. We define

$$\hat{\rho}_k(\Sigma) \triangleq \sup \left\{ \|A_1 \ldots A_k\|^{1/k} \, : \, A_1, \ldots A_k \in \Sigma \right\}.$$

One can see that for a switched linear dynamics, we have $\|y_k\| \leq \hat{\rho}_k(\Sigma)^k \|y_0\|$. The joint spectral radius of a bounded set of matrices Σ is defined as

$$\rho(\Sigma) = \lim_{k \to \infty} \hat{\rho}_k(\Sigma).$$

The joint spectral radius characterizes the stability of the switched dynamical system. Corollary 1.1 of Jungers [2009] states that for any bounded set of matrices Σ, the switched dynamical system (35) is stable if and only of $\rho(\Sigma) < 1$.

The dynamics of the accelerated variants of VI for the control case, e.g., the PID one (34), can be seen as a switched linear dynamical system with Σ being $\Sigma = \{ A_{\text{PID}}(\pi) \, : \, \pi \in \Pi \}$, where Π is the space of all possible policies (either deterministic or stochastic). If one can establish that $\rho(\Sigma) < 1$ for a particular set of controller gains, the stability of the accelerated variant of VI for the control case is shown. We remark that that there might be some structure in how $(\pi_k)_k$ are generated throughout the iterations of an accelerated VI procedure. In other words, we do not necessarily face all $\pi \in \Pi$. Because of this, formulating the error dynamics of accelerated VI for the control case as a switching linear dynamical system might be conservative. Goyal and Grand-Clement [2020] also use the joint spectral radius as a way to characterize the stability of their Accelerated Value Iteration (which is different from any of our methods). This is not surprising as the joint spectral radius is a standard tool in the stability analysis of linear time-variant systems.

Even though it is assuring that we can mathematically characterize what is needed to guarantee the stability, we do not pursue the path of computation of the joint spectral radius and optimizing it for controller design any further. We have two reasons for it. The first is that the computation of the joint spectral radius is difficult (see Section 2.2 of Jungers [2009] for a summary of results). Tsitsiklis and Blondel [1997] show that approximating $\rho(\Sigma)$ within an ε-accuracy is an NP-Hard problem. Furthermore, the problem of whether $\rho(\Sigma) \leq 1$ or not is undecidable [Blondel and Tsitsiklis, 2000]. The notion of stability used in this section, however, is based on $\rho(\Sigma) < 1$. It is not known whether this problem is undecidable or not [Blondel and Tsitsiklis, 2000, Jungers, 2009]. Despite these theoretical computational difficulties, there are methods that perform reasonably well in practice, see Section 2.3 of Jungers [2009]. The second reason is the same as what we shall shortly discuss as the challenges of formulating the controller design problem as an optimization problem in Appendix D, i.e., the unknown transition matrix $\mathcal{P}$, which makes it unsuitable for an extension to the RL setting, and impracticality of the optimization problem for large state and action spaces.

D On Selecting Controller Gains

The results of Section 4 show that one can change the error dynamics of the new variants of VI by changing the controller gains. With proper choice of the gains, one can hope to accelerate the convergence of the value function to the true value function. Given that changing the dynamics is possible, an important question is how one should choose the gains in order to achieve the intended acceleration? We briefly present four possibilities in this section.

The first approach is to treat controller gains as hyper-parameters, similar to other hyper-parameters in RL and ML, such as the learning rate of a learning algorithm or the eligibility trace parameter for TD(λ) type of online algorithms, and use some form of model selection to choose the best hyper-parameters. We perform several empirical studies in Section 6.1 and Appendix I.1 to study the effect of changing the gains on the convergence rate. Those results show that we often can find a fixed set of controller gains that leads to significant acceleration.

The second approach is to formulate the controller design problem as an optimization problem. As the dominant behaviour of the convergence is determined by the spectral radius of the error dynamics matrices A_{PD}, A_{PI}, or A_{PID} (Proposition 2), we can define the controller design problem as finding gains such that the spectral radius is as small as possible, that is,

$$\kappa \leftarrow \operatorname*{argmin}_{\kappa} \rho(A(\kappa)),$$

with κ parametrizing the controller gains. This approach, however, faces two challenges: The first challenge is that the spectral radius minimization problem is difficult (non-convex problem and possibly non-Lipschitz Overton and Womersley 1988). The second challenge, which is perhaps practically more important, is related to our eventual goal of extending these methods to the RL setting, even though that is not the focus of this work per se. The challenge is that the spectral optimization-based approach does not scale well to the RL setting because **(a)** the transition matrix $\mathcal{P}^{\pi}$ is unknown and **(b)** oftentimes we are interested in problems with very large state and action spaces, which makes the optimization problem impractical, as one has to deal with matrices with the same order of dimension as the number of states.

The third approach is to analytically find a set of gains that accelerates a subset of MDPs, without the exact knowledge of the MDP itself. We show in Appendix E how to choose the PD gains for reversible Markov chains. Reversible Markov chains have the property that their eigenvalues are all real, which makes the calculations easier. Proposition 6 in that appendix shows that if we choose $\kappa_p^* = \frac{2}{1+\sqrt{1-\gamma^2}}$ and $\kappa_d^* = (\frac{\sqrt{1+\gamma}-\sqrt{1-\gamma}}{\sqrt{1+\gamma}+\sqrt{1-\gamma}})^2$, the largest eigenvalue, hence the convergence rate, becomes

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

which is smaller than γ for $\gamma < 1$. We conjecture that it is not possible to find a single controller that significantly, or even at all, accelerates PID variants of VI for *all* MDPs. We also do not believe that the set of reversible Markov chains are relevant to most MDPs that we face in practice. For example, even the chain walk or the Garnet problems we considered earlier (Section 5) do not induce a reversible Markov chain.

The fourth approach is to design a gain adaptation mechanism that adaptively changes the gains throughout the accelerated VI procedure without the need of solving an expensive spectral minimization problem. This procedure, which might be interpreted as meta-learning of controller gains, is described in Section 5. This procedure seems to be more amenable to an RL setting, as we shall explain.

E Acceleration of the PD Variant for Reversible Markov Chains

The third approach discussed in Appendix D was to analytically find a set of gains that accelerates a subset of MDPs, without the exact knowledge of the MDP itself. This can be formulated as solving

$$\operatorname*{argmin}_{\kappa} \sup_{\mathcal{P}^\pi \in \mathcal{M}} \rho(A(\kappa; \mathcal{P}^\pi)),$$

where $A(\kappa; \mathcal{P}^\pi)$ is the error dynamics matrix for a particular transition kernel $\mathcal{P}^\pi$ and $\mathcal{M}$ is the set of MDPs we are interested in. This robust formulation ensures that no matter what MDP in the set $\mathcal{M}$ we face, the selected gain leads to a good performance (assuming that the resulting spectral radius is smaller than γ).

Computing the spectral radius for an arbitrary matrix is difficult, and does not have an analytical closed-form solution, so presumably optimizing over a set of MDPs, as specified by $\sup_{\mathcal{P}^\pi \in \mathcal{M}}$, might even be more difficult. It turns out, however, that in some cases we can solve such a problem. By focusing on the subset of *reversible Markov chains*, the calculations become much easier, and we can prescribe a set of gains that leads to acceleration. We do not, however, believe that it is possible to find a single controller that accelerates VI uniformly for all MDPs. The following proposition shows such a result.

Proposition 6. *Assume that $\mathcal{P}^\pi$ is diagonalizable in complex field. Furthermore, assume that the Markov chain induced by $\mathcal{P}^\pi$ is reversible, i.e., all eigenvalues of $\mathcal{P}^\pi$ are real-valued. With the choice of*

$$\kappa_p^* = \frac{2}{1 + \sqrt{1-\gamma^2}}, \qquad \kappa_d^* = \left(\frac{\sqrt{1+\gamma} - \sqrt{1-\gamma}}{\sqrt{1+\gamma} + \sqrt{1-\gamma}} \right)^2,$$

*the modules of all eigenvalues of the error dynamics of the **PD variant** is*

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

Proof. The roots of the error dynamics of the PD variant are specified by the quadratic equation (32) of Corollary 5. They are

$$\frac{(1 + \kappa_d - \kappa_p(1-\gamma\lambda_i)) \pm \sqrt{(1 + \kappa_d - \kappa_p(1-\gamma\lambda_i))^2 - 4\kappa_d}}{2},$$

for $i = 1, \ldots, d$. By the reversibility assumption, all λ_i are real-valued numbers between -1 and $+1$.

Under the condition that the discriminant $\Delta = \Delta(\kappa_p, \kappa_d, \lambda) = (1 + \kappa_d - \kappa_p(1-\gamma\lambda_i))^2 - 4\kappa_d$ of this equation is non-positive, the roots become complex conjugates with the magnitude of

$$\begin{aligned} |\mu_{1,2}| &= \frac{1}{2} \left| (1 + \kappa_d - \kappa_p(1-\gamma\lambda_i) \pm j\sqrt{-\Delta} \right| \\ &= \frac{1}{2} \sqrt{(1 + \kappa_d - \kappa_p(1-\gamma\lambda_i)^2 + (\sqrt{-\Delta})^2} = \frac{1}{2}\sqrt{4\kappa_d} = \sqrt{\kappa_d}. \end{aligned} \tag{36}$$

We notice that this is independent of λ_i.

To ensure that $\Delta(\lambda, \kappa_p, \kappa_d) \le 0$ for any $\lambda \in [-1, +1]$, we first find its roots as a function of κ_p, and then choose a κ_p and a κ_d such that Δ remains non-positive for any λ in that interval. The discriminant Δ can be written as the following quadratic equation in κ_p:

$$(1-\gamma\lambda)^2 \kappa_p^2 - 2(1-\gamma\lambda)(1+\kappa_d)\kappa_p + (1-\kappa_d)^2,$$

whose roots, after some algebraic manipulations, are

$$\frac{(1+\kappa_d) \pm 2\sqrt{\kappa_d}}{1-\gamma\lambda}.$$

If κ_p is between these two roots, the discriminant is non-positive. The locations of the roots, however, are a function of λ, and change. So we need to find a κ_p such that for any $\lambda \in [-1, +1]$, it is still within the roots. Let us assume that $\kappa_d \ge 0$. The minimum of the larger root is

$$\min_{\lambda \in [-1,+1]} \frac{(1+\kappa_d) + 2\sqrt{\kappa_d}}{1-\gamma\lambda} = \frac{(1+\kappa_d) + 2\sqrt{\kappa_d}}{1+\gamma}$$

and the maximum of the smaller root is

$$\max_{\lambda \in [-1,+1]} \frac{(1+\kappa_d) - 2\sqrt{\kappa_d}}{1-\gamma\lambda} = \frac{(1+\kappa_d) - 2\sqrt{\kappa_d}}{1-\gamma}$$

We choose κ_d such that these two match, that is,

$$\frac{(1+\kappa_d) + 2\sqrt{\kappa_d}}{1+\gamma} = \frac{(1+\kappa_d) - 2\sqrt{\kappa_d}}{1-\gamma} \implies \left(\frac{1+\sqrt{\kappa_d}}{1-\sqrt{\kappa_d}}\right)^2 = \frac{1+\gamma}{1-\gamma} = c.$$

Solving for κ_d, we get that

$$\kappa_d^* = \left(\frac{\sqrt{c}-1}{\sqrt{c}+1}\right)^2 = \left(\frac{\sqrt{1+\gamma}-\sqrt{1-\gamma}}{\sqrt{1+\gamma}+\sqrt{1-\gamma}}\right)^2,$$

which is also non-negative as required by assumption.

The value κ_p that is between the two roots, with this choice of κ_d^*, is

$$\kappa_p^* = \frac{1+\kappa_d^* + 2\sqrt{\kappa_d^*}}{1+\gamma} = \frac{2}{1+\sqrt{1-\gamma^2}}.$$

And the modulus of the eigenvalues (36) would be

$$|\mu_{1,2}| = \sqrt{\kappa_d^*} = \frac{\sqrt{1+\gamma}-\sqrt{1-\gamma}}{\sqrt{1+\gamma}+\sqrt{1-\gamma}}.$$

□

Some observations and remarks are in order. The first is that γ_{PD} is always less than or equal to γ. This means that with this choice of κ_p^* and κ_d^*, we can always accelerate any reversible Markov chain. As we shall discuss in Appendix G.1, the PD VI with this particular choice for controller gains coincides with the Momentum Value Iteration/Computation methods of Goyal and Grand-Clement [2020], which has the same rate as for the reversible Markov chain (for the PE case).

We would like to mention that the reversibility assumption may not be very reasonable for the MDPs in general, e.g., the chain walk problem or the Garnet problems both violate it. This is in contrast with Markov chains appearing in some MCMC methods such as Metropolis-Hasting, in which the proposal distribution is designed so that the detailed balance equation holds, and hence the reversibility of the Markov chain is ensured [MacKay, 2003, Chapter 29]. And even though a violation of an assumption does not mean that the algorithm would not work in practice, we empirically observe that the PD VI with these controller gains might actually diverge in problems such as the chain walk problem or the Garnet problems (Appendix G.1).

F Detail of Gain Adaptation

We provide further detail about the gain adaptation procedure introduced in Section 5. First, we start with explaining the general idea behind the procedure in more detail. We then justify the use of the Bellman errors as surrogates in Appendix F.1. Appendix F.2 provides the derivations of the gain adaptation procedure for the PE case. Moreover, it provides some interpretation of what each derivative term is trying to achieve, in a simplified setting. The derivation of the control case, which requires some extra care because of the nonlinearity of the Bellman optimality operator, is in Appendix F.3. Appendix F.4 provides some justifications for the use of normalization in (20) and (21). Finally, we remark on some issues, including the choice of the time horizon in the definition of the loss function and its stability, in Appendix F.5.

To define the gain adaptation algorithm, let us start defining an impractical loss function, and then find a practical surrogate for it. Suppose that at the beginning of the k-th iteration of the accelerated VI procedure, we decide to perform the procedure for T more iterations. We define the loss function as

$$J(\kappa_p, \kappa_I, \kappa_d; k, T) = \frac{1}{2}\sum_{t=0}^{T-1} a_t \left\|e_{k+t}\right\|_2^2, \tag{37}$$

with $e_i = V^\pi - V_i$ (or $e_i = Q^* - Q_i$ for control) and a_t's defining the importance of each future step. For example, if $a_0 = \cdots = a_{T-2} = 0$ and $a_{T-1} = 1$, we only care about the error after T iterations. As another example, choosing $a_t = \frac{1}{T}$, for all $t = 0, \dots, T-1$, gives the same weight to all iterations. Since the error e_i's cannot be computed, as we do not know V^π or Q^* before solving the problem itself, this is not a practical loss function. Instead, we use the Bellman error $\|T^\pi V_i - V_i\|_2^2$ (PE) or $\|T^* Q_i - Q_i\|_2^2$ (control) as surrogates. These quantities can be computed given the value function V_i or Q_i and the Bellman operator T^π or T^*. In the next section, we justify the use of Bellman error as a surrogate. To simplify the exposition, we consider the case of $T = 1$.

F.1 Justification for the Use of Bellman Error as Surrogate

As the errors $e_i = V^\pi - V_i$ (PE) or $e_i = Q^* - Q_i$ (control) could not be computed because V^π or Q^* are not available during the VI process, we used the Bellman error as a surrogate. The justification was that the Bellman error provides an upper bound on the value function error. For example, if the error is measured according to the supremum norm (and not the ℓ_2-norm as here), we have

$$\|V - V^\pi\|_\infty \le \frac{\|T^\pi V - V\|_\infty}{1-\gamma}. \tag{38}$$

This result is standard, e.g., Proposition 3.1 of Williams and Baird [1993]. A similar result holds for other L_p-norms too. Let us state and prove both of them. For this result, we do not assume the finiteness of the state space. We shall comment on how it is translated to the finite state space.

Given a probability distribution $\rho \in \mathcal{M}(\mathcal{X})$ and $1 \le p < \infty$, we define the $L_p(\rho)$-norm of a function $V : \mathcal{X} \to \mathbb{R}$ as

$$\|V\|_{p,\rho} \triangleq \sqrt[p]{\int \mathrm{d}\rho(x) |V(x)|^p}.$$

For a finite state space $\mathcal{X} = \{x_1, \cdots, x_d\}$, we get the standard ℓ_p-norm $\|V\|_p = \sqrt[p]{\sum_{i=1}^d |V(x_i)|^p}$ by choosing ρ to be the uniform distribution over the state space, i.e., $\rho(x) = \frac{1}{d}$ for all $x \in \mathcal{X}$ (up to a multiplicative constant of $\sqrt[p]{d}$). This choice, with $p = 2$, corresponds to the norm we used in the definition of the loss of the gain adaptation (37) and its surrogates (17) and (18).

Given a transition probability kernel $\mathcal{P}^\pi$, we can define the γ-discounted future-state distribution ρ_γ^π as follows. Denote the future-state distribution of following policy π from state x for k steps by $\mathcal{P}^\pi(\cdot|x;k)$, i.e., $\mathcal{P}^\pi(\cdot|x;k) \triangleq (\mathcal{P}^\pi)^k(\cdot|x)$, with the understanding that $(\mathcal{P}^\pi)^0(\cdot|x) = \mathbf{I}$ is the identity map. For an initial probability distribution $\rho \in \mathcal{M}(\mathcal{X})$, the distribution

$$\rho(\mathcal{P}^\pi)^k(\cdot) = \int \rho(\mathrm{d}x)\mathcal{P}^\pi(\cdot|x;k),$$

is the distribution of selecting the initial state according to ρ and following $\mathcal{P}^\pi$ for k steps. The γ-discounted future-state distribution ρ_γ^π is defined as

$$\rho_\gamma^\pi(\cdot) = \rho_\gamma(\cdot;\mathcal{P}^\pi) \triangleq (1-\gamma)\sum_{k\ge 0}\gamma^k \int \mathrm{d}\rho(x)\mathcal{P}^\pi(\cdot|x;k). \tag{39}$$

Given two probability distributions $\mu, \nu \in \mathcal{M}(\mathcal{X})$ with μ being an absolutely continuous w.r.t. ν, we denote the supremum of their Radon-Nikodym derivative as follows:

$$\left\|\frac{\mathrm{d}\mu}{\mathrm{d}\nu}\right\| = \sup_{x\in\mathcal{X}} \frac{\mathrm{d}\mu}{\mathrm{d}\nu}(x).$$

We use this to define the concentrability coefficient $\|\frac{\mathrm{d}\rho_\gamma^\pi}{\mathrm{d}\mu}\|_\infty$, which appears in the next result. This coefficient compares the concentration of γ-discounted future-state distribution to a base distribution μ, and computes their maximum ratio [Munos, 2007, Farahmand et al., 2010].

The following result upper bounds the value function approximation by the Bellman error, for the policy evaluation problem.

Proposition 7. *The value function error is upper bounded by the Bellman error as follows:*

$$\|V^\pi - V\|_\infty \leq \frac{\|T^\pi V - V\|_\infty}{1-\gamma}.$$

Moreover, for two distributions $\rho, \mu \in \mathcal{M}(\mathcal{X})$, and $1 \leq p < \infty$, we also have

$$\|V^\pi - V\|_{p,\rho} \leq \frac{1}{1-\gamma} \left\| \frac{\mathrm{d}\rho^\pi_\gamma}{\mathrm{d}\mu} \right\|_\infty^{\frac{1}{p}} \|T^\pi V - V\|_{p,\mu}.$$

Proof. We relate the error in value function approximation $V^\pi - V$ to its Bellman residual as follows: We consider $V^\pi - V$, and add and subtract $T^\pi V^\pi$ and $T^\pi V$ to get

$$V^\pi - V = V^\pi - T^\pi V^\pi + T^\pi V^\pi - T^\pi V + T^\pi V - V = \gamma \mathcal{P}^\pi (V^\pi - V) + T^\pi V - V.$$

where we used $V^\pi - T^\pi V^\pi = 0$. By re-arranging, we get

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

As the supremum norm of $\mathcal{P}^\pi$ is equal to 1, $(\mathbf{I} - \gamma \mathcal{P}^\pi)$ is invertible (see e.g., Lemma 2.3.3 of Golub and Van Loan 2013), and we have

$$V^\pi - V = (\mathbf{I} - \gamma \mathcal{P}^\pi)^{-1}(T^\pi V - V) = \sum_{k \geq 0} \gamma^k {\mathcal{P}^\pi}^k (T^\pi V - V). \tag{40}$$

Taking the supremum norm of both sides leads to

$$\|V^\pi - V\|_\infty \leq \sum_{k \geq 0} \gamma^k \|\mathcal{P}^\pi\|_\infty^k \|T^\pi V - V\|_\infty = \frac{\|T^\pi V - V\|_\infty}{1-\gamma},$$

where we used $\|\mathcal{P}^\pi\|_\infty = 1$.

To obtain the second part, we first take the absolute values of both sides of (40), raise it to the power of p, and integrate w.r.t. the probability measure ρ, and then apply the Jensen's inequality twice to get

$$\begin{aligned}
\|V^\pi - V\|_{p,\rho}^p = \rho |V^\pi - V|^p &= \int \mathrm{d}\rho(x) \, |V^\pi(x) - V(x)|^p \\
&= \rho \left| \sum_{k \geq 0} \gamma^k {\mathcal{P}^\pi}^k (T^\pi V - V) \right|^p \\
&\overset{\text{(a)}}{=} \rho \left| \frac{1}{1-\gamma} \sum_{k \geq 0} (1-\gamma) \gamma^k {\mathcal{P}^\pi}^k (T^\pi V - V) \right|^p \\
&\leq \frac{1}{(1-\gamma)^p} \sum_{k \geq 0} (1-\gamma) \gamma^k \rho \left| {\mathcal{P}^\pi}^k (T^\pi V - V) \right|^p \\
&\leq \frac{1}{(1-\gamma)^p} \sum_{k \geq 0} (1-\gamma) \gamma^k \rho {\mathcal{P}^\pi}^k |T^\pi V - V|^p \\
&= \frac{1}{(1-\gamma)^p} \int \mathrm{d}\rho^\pi_\gamma(x) \, |T^\pi V(x) - V(x)|^p \\
&= \frac{1}{(1-\gamma)^p} \int \frac{\mathrm{d}\rho^\pi_\gamma}{\mathrm{d}\mu}(x) \mathrm{d}\mu(x) \, |T^\pi V(x) - V(x)|^p \\
&\overset{\text{(b)}}{\leq} \frac{1}{(1-\gamma)^p} \left\| \frac{\mathrm{d}\rho^\pi_\gamma}{\mathrm{d}\mu} \right\|_\infty \|T^\pi V - V\|_{p,\mu}^p.
\end{aligned}$$

Note that at step (a), by proper normalization, we constructed terms in the form of $(1-\gamma)\gamma^k$ within the summation. These terms are positive and sum to 1. This allows us to treat them as probabilities, and therefore, apply the Jensen's inequality. At step (b), we used a change of measure argument. □

This result implies that the Bellman error $\|T^\pi V_k - V_k\|$ of a value function V_k provides an upper bound on the error of $\|V^\pi - V_k\|$, hence justifying the use of the Bellman error as a proxy. This result is for the PE case. For the control case, we have similar results, e.g., Theorem 5.3 of Munos [2007] provides a somewhat similar result. The difference, however, is that the result of Munos [2007] is not about relating the value error to the Bellman error, but is about the performance of a greedy policy of a value function to its Bellman error.

Let us focus on the L_p-norm case. This is the norm (with $p = 2$) that we used in the gain adaptation procedure in Section 5. We notice that the result allows the flexibility of choosing norms with two different probability distributions: the norm on the error of the value function compared to the true value function is w.r.t. the distribution ρ, whereas the norm with which we compute the Bellman error is w.r.t. the distribution μ. When the initial state distribution ρ is the stationary distribution ρ^π of the policy π (i.e., $\rho^\pi \mathcal{P}^\pi = \rho^\pi$), the discounted future-state distribution is the same as the stationary distribution, i.e., $\rho^\pi_\gamma = \rho^\pi$. If μ is also selected to be ρ^π, we get the simplified result of

$$\|V^\pi - V\|_{p,\rho^\pi} \leq \frac{1}{1-\gamma} \|T^\pi V - V\|_{p,\rho^\pi} .$$

For the finite state case with $\rho(x) = \mu(x) = \frac{1}{d}$ for all $x \in \mathcal{X}$, one can provide a (conservative) upper bound on the R-N derivative $\|\frac{\mathrm{d}\rho^\pi_\gamma}{\mathrm{d}\mu}\|_\infty$. The numerator is maximized when all next-states concentrate on a single state, say, x_1. In that case $\rho^\pi_\gamma(x_1) = (1-\gamma)[\frac{1}{d} + \frac{d}{d}(\gamma + \gamma^2 + \dots)] = \frac{1}{d}(1-\gamma) + \gamma$ and $\rho^\pi_\gamma(x_i) = (1-\gamma)\frac{1}{d}$ (for $i = 2, \dots, d$). Given the denominator $\mu(x) = \frac{1}{d}$, the supremum of the R-N derivative is $(1-\gamma)[1 + d\frac{\gamma}{1-\gamma}] \leq 1 + d\gamma \approx d\gamma$, where the approximation is for large enough d. This gives the upper bound of

$$\|V^\pi - V\|_2 \leq \frac{\sqrt{1+\gamma d}}{1-\gamma} \|T^\pi V - V\|_2 .$$

We note that this is quite an unfavourable situation, which is constructed based on an extreme concentration of the next-states. In any case, this shows that one may use the Bellman error as an upper bound surrogate of the error of value function, which is the one we should be interested in.

F.2 Derivation of Gain Adaptation for Policy Evaluation and Some Interpretations

By choosing the Bellman error as the surrogate loss, and only considering one-step ahead rollout ($T = 1$), we define the following loss functions for the PE and control cases:

$$J_{\mathrm{BE}}(k) = \frac{1}{2} \|T^\pi V_k - V_k\|_2^2 = \frac{1}{2} \|\mathrm{BR}(V_k)\|_2^2 , \tag{41}$$

$$J^*_{\mathrm{BE}}(k) = \frac{1}{2} \|T^* Q_k - Q_k\|_2^2 = \frac{1}{2} \|\mathrm{BR}^*(Q_k)\|_2^2 . \tag{42}$$

These are the same as (17) and (18), which we quote here for ease of reference. We could also define the loss for the control case based on $\mathrm{BR}^*(V) = T^* V - V$.

The gain adaptation process can be performed by taking the gradient of $J_{\mathrm{BE}}(k)$ (41) w.r.t. the controller parameters at the k-th iteration. The result is already reported in Section 5. Here we derive the result for the PE case, and provide some insights and interpretations. The result for the control case is derived in Appendix F.3.

We consider that V_k is generated using the PID variant (13) with scalar gains. To emphasize the dependence of controller parameters, which change in a gain adaptation framework, on the iteration number, we may use a superscript (k). The PID variant of VI is then

$$\begin{aligned} z_{k+1} &= \beta^{(k)} z_k + \alpha^{(k)} \mathrm{BR}(V_k), \\ V_{k+1} &= (1 - \kappa_p^{(k)}) V_k + \kappa_p^{(k)} T^\pi V_k + \kappa_I^{(k)} \left[\beta^{(k)} z_k + \alpha^{(k)} \mathrm{BR}(V_k) \right] + \kappa_d^{(k)} (V_k - V_{k-1}). \end{aligned}$$

The Bellman residual $\mathrm{BR}(V_k)$, as a function of V_{k-1}, z_{k-1}, and controller parameters, is

$$\begin{aligned} \mathrm{BR}(V_k) &= T^\pi V_k - V_k \\ &= r^\pi + (\gamma \mathcal{P}^\pi - \mathbf{I}) \left[(1-\kappa_p) V_{k-1} + \kappa_p T^\pi V_{k-1} + \kappa_I \left[\beta z_{k-1} + \alpha \mathrm{BR}(V_{k-1}) \right] + \kappa_d (V_{k-1} - V_{k-2}) \right] . \end{aligned} \tag{43}$$

We have

$$\frac{\partial J_{\mathrm{BE}}(k)}{\partial \kappa_{\cdot}} = \left\langle \mathrm{BR}(V_k)\,,\, \frac{\partial \mathrm{BR}(V_k)}{\partial \kappa_{\cdot}} \right\rangle_{\mathcal{X}}, \tag{44}$$

where

$$\langle\, V_1\,,\, V_2\,\rangle_{\mathcal{X}} = \sum_{x \in \mathcal{X}} V_1(x) V_2(x)$$

for a discrete state space, or

$$\langle\, V_1\,,\, V_2\,\rangle_{\mathcal{X}} = \int V_1(x) V_2(x) \mathrm{d}\mu(x)$$

with a choice of $\mu \in \mathcal{M}(\mathcal{X})$ for more general state space (which of course, could be a discrete one, if we choose μ as a probability mass function). Taking the partial derivatives of (43) w.r.t. the parameters of the controller, we obtain the followings (see Table 1 in Section 5):

$$\begin{aligned}
\frac{\partial \mathrm{BR}(V_k)}{\partial \kappa_p} &= -(\mathbf{I} - \gamma \mathcal{P}^\pi)\mathrm{BR}(V_{k-1}) = (\mathbf{I} - \gamma \mathcal{P}^\pi)^2 e_{k-1}, \\
\frac{\partial \mathrm{BR}(V_k)}{\partial \kappa_d} &= -(\mathbf{I} - \gamma \mathcal{P}^\pi)(V_{k-1} - V_{k-2}) = -(\mathbf{I} - \gamma \mathcal{P}^\pi)(e_{k-1} - e_{k-2}), \\
\frac{\partial \mathrm{BR}(V_k)}{\partial \kappa_I} &= -(\mathbf{I} - \gamma \mathcal{P}^\pi)\left[\beta^{(k-1)} z_{k-1} + \alpha^{(k-1)} \mathrm{BR}(V_{k-1})\right] = -(\mathbf{I} - \gamma \mathcal{P}^\pi) z_k, \\
\frac{\partial \mathrm{BR}(V_k)}{\partial \alpha} &= -\kappa_I(\mathbf{I} - \gamma \mathcal{P}^\pi)\mathrm{BR}(V_{k-1}) = \kappa_I(\mathbf{I} - \gamma \mathcal{P}^\pi)(\mathbf{I} - \gamma \mathcal{P}^\pi) e_{k-1}, \\
\frac{\partial \mathrm{BR}(V_k)}{\partial \beta} &= -\kappa_I(\mathbf{I} - \gamma \mathcal{P}^\pi) z_{k-1}.
\end{aligned} \tag{45}$$

Several remarks are in order. For most cases, we have provided two equivalent formulae. One set of formulae depend on quantities that can be computed based on the information available throughout the iterations of PID VI, e.g., V_k, V_{k-1}, z_{k-1}, and the Bellman residual $\mathrm{BR}(V_{k-1})$. The second set of formulae are based on the error $e_{\cdot}$, and are obtained because of the relation between the Bellman residual and the error e_{k-1} (8). As $e_{k-1} = V_{k-1} - V^\pi$ is not known, those formulae are not for the computation purpose, but to provide another interpretation of what each derivative computes, as we discuss next.

To gain an intuition of each derivative term, we use the error-based formulae in (45) and make a simplifying assumption that $\mathcal{P}^\pi = \mathbf{I}$. This is a very simplified dynamics, in which each state transits back to itself. In this case, the Bellman residual is $\mathrm{BR}(V_k) = -(1-\gamma) e_k$, see (8), and is proportional to the error in the value function approximation. For the gain of the proportional term, we have

$$\frac{\partial J_{\mathrm{BE}}(k)}{\partial \kappa_p} \propto \langle\, e_k\,,\, e_{k-1}\,\rangle\,.$$

This shows that whenever the errors of two consecutive iterations are aligned (e.g., both are positive), the increase in κ_p leads to an increase in the Bellman error; we have to decrease the proportional gain to reduce the Bellman error. And vice versa for when they are not aligned.

For the gain of the derivative term, we have

$$\frac{\partial J_{\mathrm{BE}}(k)}{\partial \kappa_d} \propto -\langle\, e_k\,,\, e_{k-1} - e_{k-2}\,\rangle\,.$$

This shows that whenever the linear trend of errors, as quantified by $e_{k-1} - e_{k-2}$, is aligned with the current error e_k, the increase in κ_d leads to a decrease in the Bellman error.

To obtain an intuition for the derivative w.r.t. the gain of the integral term, first note that under the simplifying assumption of $\mathcal{P}^\pi = \mathbf{I}$, the dynamics of z_k would be (assuming fixed α and β)

$$z_{k+1} = \beta z_t - \alpha(1-\gamma) e_k = \beta z_t + \alpha(1-\gamma)(-e_k),$$

which is essentially an exponentially moving average (or low-pass filter) on the negative of the error terms ($-e_k$). Then,

$$\frac{\partial J_{\mathrm{BE}}(k)}{\partial \kappa_I} \propto - \langle\, e_k \,,\, z_k \,\rangle$$

can be interpreted as computing the alignment of the current error with the exponentially moving average of the negative of the past error terms. If e_k is aligned with the moving average of the past error terms (and not their negative values), an increase of κ_I leads to an increase of the Bellman error.

F.3 Derivation of Gain Adaptation for Control

Let us now turn to the problem of gain adaptation for the control scenario where the PID variant of the accelerated VI is

$$\begin{aligned} z_{k+1} &= \beta^{(k)} z_k + \alpha^{(k)} \mathrm{BR}^*(Q_k), \\ Q_{k+1} &= (1-\kappa_p^{(k)}) Q_k + \kappa_p^{(k)} T^* Q_k + \kappa_I^{(k)} \left[\beta^{(k)} z_k + \alpha^{(k)} \mathrm{BR}^*(Q_k) \right] + \kappa_d^{(k)} (Q_k - Q_{k-1}). \end{aligned} \tag{46}$$

To compute the derivation of the gradient of (42), we are facing the challenge that the Bellman optimality operator is nonlinear, so changing the order of the derivative and the Bellman operator requires some extra care not needed in the PE case. The problem, however, is not too much of a challenge when we notice that the (nonlinear) Bellman optimality operator applied to an action-value function Q corresponds to the (linear) Bellman operator of the greedy policy $\pi(\cdot; Q)$ (23) of Q applied to Q. We have

$$T^* Q = T^{\pi(Q)} Q.$$

Note that when $T^{\pi(Q)}$ is applied to an action-value function Q, its effect is

$$(T^{\pi(Q)} Q)(x,a) = r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}y|x,a) Q(y, \pi(y; Q)).$$

Therefore, this and (46) show that

$$\begin{aligned} \mathrm{BR}^*(Q_k) &= T^* Q_k - Q_k \\ &= r + \gamma \mathcal{P}^{\pi(Q_k)} Q_k - Q_k \\ &= r + \left(\gamma \mathcal{P}^{\pi(Q_k)} - \mathbf{I} \right) \Bigg[(1-\kappa_p) Q_{k-1} + \kappa_p T^* Q_{k-1} + \\ &\qquad\qquad \kappa_I \left(\beta^{(k-1)} z_{k-1} + \alpha^{(k-1)} \mathrm{BR}^*(Q_{k-1}) \right) + \\ &\qquad\qquad \kappa_d \left(Q_{k-1} - Q_{k-2} \right) \Bigg]. \end{aligned} \tag{47}$$

Before taking the derivative of this Bellman residual for the optimality operator, which is similar to the PE case, let us also provide two alternative formulations of $\mathrm{BR}^*(Q)$ that are expressed in terms of the value error, similar to (8) for the PE case. We have

$$\mathrm{BR}^*(Q) = T^* Q - Q = (T^* Q - Q^*) - (Q - Q^*) = \left(T^{\pi(Q)} Q - T^{\pi^*} Q^* \right) - (Q - Q^*). \tag{48}$$

The first term $T^{\pi(Q)} Q - T^{\pi^*} Q^*$ can be written as

$$\begin{aligned} T^{\pi(Q)} Q - T^{\pi^*} Q^* &= \gamma \left(\mathcal{P}^{\pi(Q)} Q - \mathcal{P}^{\pi^*} Q^* \right) \\ &= \gamma \left(\mathcal{P}^{\pi(Q)} Q - \mathcal{P}^{\pi(Q)} Q^* + \mathcal{P}^{\pi(Q)} Q^* - \mathcal{P}^{\pi^*} Q^* \right) \\ &= \gamma \left(\mathcal{P}^{\pi(Q)} (Q - Q^*) + \Delta \mathcal{P}^{\pi(Q)} Q^* \right), \end{aligned}$$

with $\Delta\mathcal{P}^{\pi(Q)} \triangleq \mathcal{P}^{\pi(Q)} - \mathcal{P}^{\pi^*}$ being the difference between the transition kernel corresponding to the greedy policy $\pi(Q)$ and the optimal policy π^*.

We also may decompose the first term differently:

$$
\begin{aligned}
T^{\pi(Q)}Q - T^{\pi^*}Q^* &= \gamma\left(\mathcal{P}^{\pi(Q)}Q - \mathcal{P}^{\pi^*}Q^*\right) \\
&= \gamma\left(\mathcal{P}^{\pi(Q)}Q - \mathcal{P}^{\pi^*}Q + \mathcal{P}^{\pi^*}Q - \mathcal{P}^{\pi^*}Q^*\right) \\
&= \gamma\left(\mathcal{P}^{\pi^*}(Q - Q^*) + \Delta\mathcal{P}^{\pi(Q)}Q\right).
\end{aligned}
$$

Together with (48), we get the following two forms for $\mathrm{BR}^*(Q)$:

$$
\begin{aligned}
\mathrm{BR}^*(Q) &= \left(\gamma\mathcal{P}^{\pi(Q)} - \mathbf{I}\right)e + \gamma\Delta\mathcal{P}^{\pi(Q)}Q^* \\
&= \left(\gamma\mathcal{P}^{\pi^*} - \mathbf{I}\right)e + \gamma\Delta\mathcal{P}^{\pi(Q)}Q. \qquad (49)
\end{aligned}
$$

Comparing with (8), we see that the Bellman residual with the optimality operator has an extra term. When the greedy policy of Q is the optimal policy π^*, the extra terms are zero.

Taking the partial derivatives of (47) w.r.t. the parameters of the controller, and using (49) (only the first relation), we get that

$$
\begin{aligned}
\frac{\partial \mathrm{BR}^*(Q_k)}{\partial \kappa_p} &= -(\mathbf{I} - \gamma\mathcal{P}^{\pi(Q_k)})\mathrm{BR}^*(Q_{k-1}) = (\mathbf{I} - \gamma\mathcal{P}^{\pi(Q_k)})\left[\left(\mathbf{I} - \gamma\mathcal{P}^{\pi(Q_{k-1})}\right)e_{k-1} - \gamma\Delta\mathcal{P}^{\pi(Q_{k-1})}Q^*\right], \\
\frac{\partial \mathrm{BR}^*(Q_k)}{\partial \kappa_d} &= -(\mathbf{I} - \gamma\mathcal{P}^{\pi(Q_k)})(Q_{k-1} - Q_{k-2}) = -(\mathbf{I} - \gamma\mathcal{P}^{\pi(Q_k)})(e_{k-1} - e_{k-2}), \\
\frac{\partial \mathrm{BR}^*(Q_k)}{\partial \kappa_I} &= -(\mathbf{I} - \gamma\mathcal{P}^{\pi(Q_k)})\left[\beta^{(k-1)}z_{k-1} + \alpha^{(k-1)}\mathrm{BR}^*(Q_{k-1})\right] = -(\mathbf{I} - \gamma\mathcal{P}^{\pi(Q_k)})z_k, \\
\frac{\partial \mathrm{BR}^*(Q_k)}{\partial \alpha} &= -\kappa_I(\mathbf{I} - \gamma\mathcal{P}^{\pi(Q_k)})\mathrm{BR}^*(Q_{k-1}) = \kappa_I(\mathbf{I} - \gamma\mathcal{P}^{\pi(Q_k)})\left[\left(\mathbf{I} - \gamma\mathcal{P}^{\pi(Q_{k-1})}\right)e_{k-1} - \gamma\Delta\mathcal{P}^{\pi(Q_{k-1})}Q^*\right], \\
\frac{\partial \mathrm{BR}^*(Q_k)}{\partial \beta} &= -\kappa_I(\mathbf{I} - \gamma\mathcal{P}^{\pi(Q_k)})z_{k-1}.
\end{aligned}
$$

We provide two sets of formulae, one based on the quantities that are easy to compute, and one based on those depending on the error e_k. The first set is reported in Table 1. The update rule would be the same as (22) with obvious modifications.

F.4 Justification for Normalized Gradient Descent

Performing gradient descent based on (44) might lead to a very slow convergence. To see this, after applying the Cauchy-Schwarz inequality to (44), we get that

$$
\left|\frac{\partial J_{\mathrm{BE}}(k)}{\partial \kappa_\cdot}\right| \leq \left\|\mathrm{BR}(V_k)\right\|_2 \left\|\frac{\partial \mathrm{BR}(V_k)}{\partial \kappa_\cdot}\right\|_2. \qquad (50)
$$

From (45), we have

$$
\begin{aligned}
\left\|\frac{\partial \mathrm{BR}(V_k)}{\partial \kappa_p}\right\|_2 &\leq \left\|\mathbf{I} - \gamma\mathcal{P}^\pi\right\|_2 \left\|\mathrm{BR}(V_{k-1})\right\|_2 \\
\left\|\frac{\partial \mathrm{BR}(V_k)}{\partial \kappa_d}\right\|_2 &= \left\|(\mathbf{I} - \gamma\mathcal{P}^\pi)(e_{k-1} - e_{k-2})\right\|_2 \\
&= \left\|(\mathbf{I} - \gamma\mathcal{P}^\pi)(\mathbf{I} - \gamma\mathcal{P}^\pi)^{-1}\left(\mathrm{BR}(V_{k-1}) - \mathrm{BR}(V_{k-2})\right)\right\|_2 \\
&= \left\|\mathrm{BR}(V_{k-1}) - \mathrm{BR}(V_{k-2})\right\|_2 \\
&\leq \left\|\mathrm{BR}(V_{k-1})\right\|_2 + \left\|\mathrm{BR}(V_{k-2})\right\|_2
\end{aligned}
$$

$$\left\| \frac{\partial \mathrm{BR}(V_k)}{\partial \kappa_I} \right\|_2 \le \left\| \mathbf{I} - \gamma \mathcal{P}^\pi \right\|_2 \left\| z_k \right\|_2 ,$$

$$\left\| \frac{\partial \mathrm{BR}(V_k)}{\partial \alpha} \right\|_2 = |\kappa_I| \left\| \mathbf{I} - \gamma \mathcal{P}^\pi \right\|_2 \left\| \mathrm{BR}(V_{k-1}) \right\|_2$$

$$\left\| \frac{\partial \mathrm{BR}(V_k)}{\partial \beta} \right\|_2 = |\kappa_I| \left\| \mathbf{I} - \gamma \mathcal{P}^\pi \right\|_2 \left\| z_{k-1} \right\|_2 .$$

These together with (50) show that the magnitude of the derivatives w.r.t. κ_p, κ_d, and α is approximately proportional to the Bellman error squared (more precisely, it is of $O(\|\mathrm{BR}(V_k)\|_2 \|\mathrm{BR}(V_{k-1})\|_2)$ for κ_p, etc.). For the derivatives w.r.t. κ_I and β, which have dependence on $\|z_k\|_2$, we have a similar, though not precisely the same, behaviour. This is because z_k is a low-pass filter on the Bellman residuals.

If we assume that the PID VI procedure is stable and e_k converges to zero with a rate of γ'^k (with $|\gamma'| < 1$), this implies that the $\mathrm{BR}(V_k)$ is also converging with the same rate. The consequence is that the derivative of these terms converge to zero with a rate of γ'^{2k}. Therefore, if the learning rate η is fixed, this implies that most of the variation in the controller gains occur in earlier iterations, as the change from iteration k_0 onward is proportional to

$$\frac{\eta \gamma'^{2k_0}}{1 - \gamma'^2},$$

which can be quite small.

F.5 Other Remarks

We collect some remarks regarding the gain adaptation procedure in this section.

We started the description of the gain adaptation procedure in this appendix by defining a multi-step loss function $\sum_{t=0}^{T-1} a_t \|e_{k+t}\|_2^2$ (37), and then focused on the case of $T = 1$ for simplicity of derivations and exposition. As the error e cannot be computed during the process, we used the Bellman error as a surrogate. One may consider a more general $T \ge 1$ case too, in which case the surrogate loss functions would be

$$J_{\mathrm{BE}}(\kappa_p, \kappa_I, \kappa_d; k, T) = \frac{1}{2} \sum_{t=0}^{T-1} a_t \left\| T^\pi V_{k+t} - V_{k+t} \right\|_2^2 ,$$

$$J^*_{\mathrm{BE}}(\kappa_p, \kappa_I, \kappa_d; k, T) = \frac{1}{2} \sum_{t=0}^{T-1} a_t \left\| T^* Q_{k+t} - Q_{k+t} \right\|_2^2 .$$

One would use these loss functions in a receding horizon manner by unrolling the PID VI procedure and finding the current best optimal value of the controller gains, or perhaps just move in the direction of its gradient. More concretely, at iteration k, we perform the PID VI for T steps with the current parameters of the controller to obtain $V_{k,k+1}, \dots, V_{k,k+T}$. Here k in $V_{k,k'}$ refers to the main iteration of PID VI, and k' refers to the index of the intermediate value functions generated based on the V_k for the purpose of gain adaptation. Using these intermediate value functions, we compute the Bellman errors $\|T^\pi V_{k,k+t} - V_{k,k+t}\|$ for $t = 0, \dots, T-1$. The gradient of these Bellman errors w.r.t. the controller parameters can be obtained similar to how we derived for the simpler $T = 1$ case. We use the gradient to update the controller parameters. We then keep $V_{k+1} = V_{k,k+1}$ as the starting point of another T steps of the PID VI with the updated controller parameters, and repeat the process.

A potential benefit of this procedure is that it is less myopic compared to what we presented in Section 5. Instead of optimizing for the reduction of the Bellman error at the next step, this procedure incorporates the long-term effect of the parameter change. Nevertheless, we do not currently know whether choosing a large T leads to much better results or not. An evidence that this myopia might have a significant negative effect is studied in the context of optimization of DNN by Wu et al. [2018]. In one of their results, they considered optimizing a noisy quadratic function using a momentum variant of SGD. When the quadratic function is ill-conditioned and the observations are noisy, a myopic solution leads to a selection of a small learning rate compared to a non-myopic one. This consequently prevents the optimizer to find a good minimizer fast enough. This is not an issue when the observations are not noisy, or the quadratic function is not ill-conditioned. They also have similar empirical results for training of a DNN. It is not clear whether that result translates to this context or not. In any case, studying the effect of T in the gain adaptation procedure is an interesting topic, which deserves further study. As this is the first work on this line, we do not pursue studying $T > 1$ case in this paper.

The downside of this procedure, however, is that it becomes computationally expensive as T increases. At some point, the extra computation for the adaptation procedure may not worth the benefit of having a PID VI procedure with a faster convergence rate. Another disadvantage of this procedure is that it requires the knowledge of the model. Even though this is assumed throughout this paper, it becomes a barrier when we want to adopt this gain adaptation procedure to the RL context, in which the model is not known a priori. For $T = 1$, we can still compute the Bellman error using data (e.g., using the empirical Bellman error), albeit possibly in a biased fashion [Lagoudakis and Parr, 2003, Antos et al., 2008], and perform the gain adaptation procedure. This is more complicated for $T > 1$. Extension of the gain adaptation to the RL setting, especially considering the case of $T > 1$, is an interesting research direction.

We remark that even though performing gradient descent on hyper-parameters is a reasonable idea and makes the problem of hyper-parameter selection much easier, it does not completely eliminate it as we still require to choose the meta-learning rate η. Although the choice of one parameter is often much simpler, and perhaps even less sensitive to changes (as our experiments suggested), it still requires some attention. Choosing a large meta-learning rate might lead to instability, as has already been observed decades ago in the context of the so-called MIT rule for adaptive control [Aström and Wittenmark, 1994]. This should not be considered as a criticism specific to our method. This is a criticism for most gradient-based approaches for adaptation of hyper-parameters of a learning procedure, including most methods mentioned earlier in the beginning of Section 5. Designing controller gain (or learning rate) adaptation method with stability guarantee is an interesting topic for future research.

G Related Work (Detailed)

This paper has brought ideas from control theory to design methods for accelerated computation of the value function. In this section, we discuss some relevant bodies of research, which we only briefly mentioned before. Appendix G.1 reviews some algorithms that are designed to accelerate RL methods. We focus on those that modify basic building blocks, such as value iteration or policy iteration. We particularly discuss some recently proposed methods such as the work of Goyal and Grand-Clement [2020] (Appendix G.1.1) and Momentum Value Iteration and Anderson Accelerated-based algorithms (Appendix G.1.2) in some detail, as some of them are either similar to some of the proposed methods or have high-level structural similarities. We briefly review a few other RL methods that can be interpreted as acceleration methods (Appendix G.1.3), even though they have quite different flavour than ours.

There are some work in the continuous optimization literature that find the connection between concepts in control theory and optimization methods. Even though VI is not a conventional optimization method, we would like to acknowledge this connection (Appendix G.2). We also mention some papers for learning rate adaptation (Appendix G.3).

G.1 Acceleration in RL

There have been some recent work for accelerating RL algorithms [Geist and Scherrer, 2018, Shi et al., 2019, Vieillard et al., 2020b, Goyal and Grand-Clement, 2020, Devraj and Meyn, 2017]. Similar to this work, they all modify basic RL/Planning algorithms with the goal of accelerating them. They often bring a commonly used acceleration technique in other areas of numerical analysis, such as Anderson acceleration [Anderson, 1965] in the fixed-point approximation literature or momentum gradient descent in the optimization literature, in order to design new RL/Planning algorithms. In contrast to this work, none of them are motivated by tools in the control theory. Let us describe them more closely.

G.1.1 Comparison with Goyal and Grand-Clement [2020]

Goyal and Grand-Clement [2020] design accelerated VI methods based on insights from the first-order convex optimization. They observe the similarity between gradient and the Bellman residual $\text{BR}(V) = T^{\pi}V - V$:

the gradient is zero at the stationary point of an optimization problem; the Bellman residual is zero at the correct value function. They also show a deeper connection: the Bellman residual has properties similar to a $(1-\gamma)$-strongly convex and $(1+\gamma)$-Lipschitz function. In particular, they show that

$$(1-\gamma)\left\|V_1 - V_2\right\|_\infty \le \left\|(\mathbf{I}-T)(V_1) - (\mathbf{I}-T)(V_2)\right\|_\infty \le (1+\gamma)\left\|V_1 - V_2\right\|_\infty .$$

These two inequalities are analogous to the following inequalities for μ-strong convex, L-Lipschitz continuous function $f : \mathbb{R}^d \to \mathbb{R}$:

$$\mu\left\|x_1 - x_2\right\|_2 \le \left\|\nabla f(x_1) - \nabla f(x_2)\right\|_2 \le L\left\|x_1 - x_2\right\|_2 .$$

These connections suggest that one may use various optimization methods to design new VI-like algorithms. In particular, they use accelerated optimization methods. Based on the analogy to the gradient descent (i.e., $x_{k+1} = x_k - \alpha_k \nabla f(x_k)$), we get

$$V_{k+1} = V_k + \alpha_k \mathrm{BR}(V_k).$$

They call this method Relaxed Value Iteration. This is the same as the P-variant of VI (9), which we obtained by interpreting the VI procedure as a specific choice of controller. This is also the same algorithm as the Accelerated Jacobi procedure, as introduced by Kushner and Kleinman [1971].

Based on the connection with the Nesterov's Accelerated Gradient Descent, they suggest the following procedure:

$$\begin{aligned} U_k &= V_k + \beta_k (V_k - V_{k-1}), \\ V_{k+1} &= U_k + \alpha_k (TU_k - U_k), \end{aligned} \tag{51}$$

where T is either T^* (control) or T^π (PE). Based on the similarity with $(1-\gamma)$-strongly convex and $(1+\gamma)$-Lipschitz function, they prescribe a fixed $\alpha_k = \frac{1}{1+\gamma}$ and $\beta_k = \frac{1-\sqrt{1-\gamma^2}}{\gamma}$. They call this method Accelerated Value Iteration (A-VI) (for Control) or Accelerated Value Computation (A-VC) (for PE). This method is not the same as any of the PID variants of the VI introduced in our work.

This approach had been introduced before in the fixed-point approximation literature [Berinde, 2007]. Maingé [2008] proposes the inertia Krasnoselskii-Mann (KM) procedure for the computation of a fixed point of a sequence of operators. The A-VI/VC of Goyal and Grand-Clement [2020] are essentially the same as the inertia KM method when the operator is the Bellman operator. Inertia KM method is defined as follows (with notation similar to ours): Consider a Hilbert space $\mathcal{H}$ and a sequence of self-mapping operators (L_k) on that space. For an operator L, denote the set of fixed points by $\mathrm{Fix}(L) = \{\, V \,:\, LV = V \,\}$. The inertia Krasnoselskii-Mann (KM) procedure is

$$\begin{aligned} U_k &= V_k + \theta_k (V_k - V_{k-1}), \\ V_{k+1} &= \left[(1-w_k)\mathbf{I} + w_k L_k\right] U_k, \end{aligned} \tag{52}$$

with $(V_0, V_1) \in \mathcal{H} \times \mathcal{H}$, $(\theta_k) \subset [0,1]$, and $(\theta_k) \subset (0,2)$. The common fixed-point set is denoted by $\overline{\mathrm{Fix}} = \cap_k \mathrm{Fix}(T_k)$. Maingé [2008] then shows that under certain conditions, V_k converges to a $\bar{V} \in \overline{\mathrm{Fix}}$ for a large class of operators, e.g., set of non-expansive operators, which satisfy $\|TV_1 - TV_2\| \le \|V_1 - V_2\|$ with the norm being the Hilbert space norm. Comparing (51) and (52) shows that A-VI/VC is a special case of inertia KM.

Momentum Value Iteration/Computation (M-VI/VC) is another method suggested by Goyal and Grand-Clement [2020].[5] This method is inspired by the Polyak's momentum method (or heavy ball) in optimization (e.g., Polyak, 1964 and Polyak, 1987, Section 3.2). It is

$$\begin{aligned} V_{k+1} &= V_k - \alpha_k \mathrm{BR}(V_k) + \beta_k (V_k - V_{k-1}) \\ &= (1-\alpha_k)V_k + \alpha_k TV_k + \beta_k (V_k - V_{k-1}), \end{aligned} \tag{53}$$

[5] Note that the Momentum VI was not mentioned in the original submission in May 2019. It is first mentioned in v5 (December 2019) as a remark, and further analyzed in v6 (March 2020).

where T is either T^{π} (PE) or T^{*} (Control). Based on the analogy to convex optimization, they prescribe choosing $\alpha = 2/(1+\sqrt{1-\gamma^2})$ and $\beta = (1-\sqrt{1-\gamma^2})/(1+\sqrt{1-\gamma^2})$.

Interestingly, the Momentum VI/VC is the same as the PD variant of VI (11), proposed in this work, with a specific choice of controller parameters $\kappa_p = \alpha$ and $\kappa_d = \beta$. This is interesting because these methods are derived from two different perspectives: using the PD controller to modify VI vs. bringing a convex optimization method to modify VI.

They study the convergence rate of A-VC and M-VC (PE) for reversible Markov chains with their specific choice of step sizes. The convergence rates they obtain are $\frac{\sqrt{1-\gamma}-\sqrt{1+\gamma}}{\sqrt{1-\gamma}+\sqrt{1+\gamma}}$ for M-VC, and $1-\sqrt{\frac{1-\gamma}{1+\gamma}}$ for A-VC. Both of these are faster than γ of the conventional VI. One can show that the rate of M-VC is faster than A-VC's too.

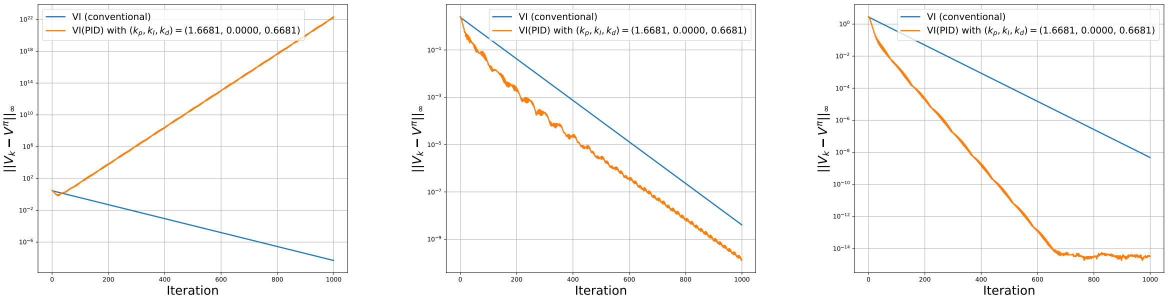
We note that under the same reversibility assumption, we derive the parameters for the PD VI variant that leads to the same choice of parameters and the same rate (Proposition 6 in Appendix E).

Even though having a parameter set that guarantees to accelerate the computation of the value function for reversible Markov chains is nice, we point out that the reversibility assumption is restrictive in the MDP/RL context. Recall that a reversible Markov chain satisfies the detailed balance equation (3) and has a transition matrix with real-valued eigenvalues. The detailed balance is a reasonable assumption when we deal with the Markov chain induced by an MCMC method; in fact, we design the proposal distribution in MCMC to satisfy the detailed balance equation [MacKay, 2003, Chapter 29]. But it is not a reasonable assumption in the MDP/RL context. For example, a problem as simple as the chain walk is not reversible, neither is the Garnet problem. They both have complex-valued eigenvalues. Although this does not mean that A-VC and M-VC would diverge for more general problems, as reversibility is only a sufficient condition, we have empirically observed that M-VC (which is the same as PD VI with parameters selected for the reversible Markov chain according to Proposition 6) does diverge, as we shall explain soon. It seems that optimizing the parameters under such assumption may lead to "aggressive" parameter choice that leads to divergence, even though there might exist parameters that lead to a stable condition. In our description of PID VI, we did not initially prescribe how the controller parameters should be selected because we either have to make restrictive assumption on the Markov chain (e.g., reversibility), or the controller parameters should be selected very conservatively. In Appendix E we show how a result under restrictive assumption would look like. But we believe that the right way to approach the problem of choosing controller gains is not to select them a priori, but to choose them problem-dependently, for example by following a gain adaptation procedure, as described in Section 5.

We would like to mention that M-VC/VI diverges for the chain walk problem for large enough discount factor. For the PE problem (M-VC), it diverges for any $\gamma \geq 0.86$. For the control problem (M-VI), it diverges for any $\gamma \geq 0.93$. We also tried it on our variant of the Garnet problem. When the branching factor is 3 and the number of non-zero rewards is 5, as in our experiments, for the PE problem M-VC often diverges for any discount factor $\gamma \geq 0.85$ (since the MDP is random, the behaviour of M-VC is different for each MDP instance). Through some initial experiments, not reported here, we empirically observed that for these relatively sparsely connected MDPs, the complex eigenvalues have large imaginary components. When we increase the branching factor, however, the eigenvalues get closer to the real line, and the problem becomes more "similar" to a reversible Markov chain. As an example, we show the behaviour of M-VC for three branching factors of 20, 30, and 40 for $\gamma = 0.98$ in Figure 5 and for $\gamma = 0.99$ in Figure 6 (we used the same fixed seed to generate them). We observe that for $\gamma = 0.98$, M-VC leads to acceleration for the branching factors of 30 and 40. It is unstable for $\gamma = 0.99$.

G.1.2 Comparison with Momentum Value Iteration and Anderson Accelerated Algorithms

Vieillard et al. [2020b] introduce Momentum Value Iteration (MoVI). Again, it is motivated by making an analogy to the optimization literature, as opposed to the control theory literature, as was done in this work. They make an informal analogy between the action-value function and the gradient: the same way that the gradient is the direction that a function changes the most, the greedy policy $\pi_k(\cdot|x) = \text{argmax}_{\pi(\cdot|x)\in\Delta_{|\mathcal{A}|}} \langle \pi(\cdot|x), Q(x,\cdot) \rangle$, where $\Delta_{|\mathcal{A}|}$ is $|\mathcal{A}|$-dimensional simplex, is the direction the policy is most linearly aligned with an action-value function. This interpretation is different from the connection between the Bellmen Residual and the gradient in the work of Goyal and Grand-Clement [2020]. Based on this connection, they argue that the same way that the momentum can be used to stabilize the gradient descent,

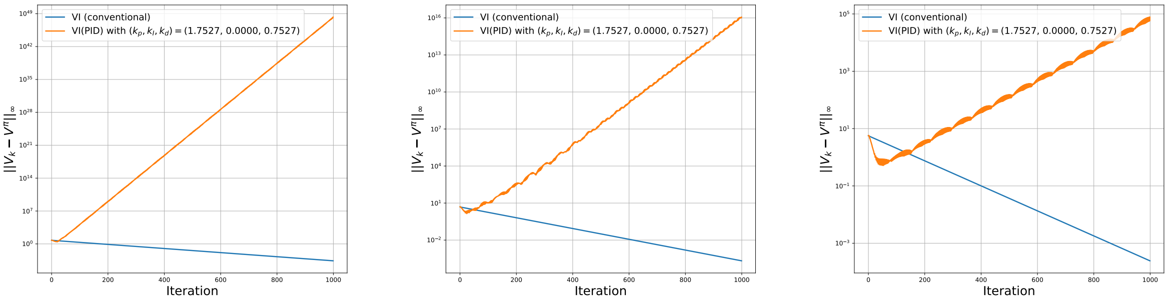


(a) Branching factor = 20

(b) Branching factor = 30

(c) Branching factor = 40

Figure 5: (Garnet - Policy Evaluation) Behaviour of M-VC for a 50-state Garnet problem with $\gamma = 0.98$.



(a) Branching factor = 20

(b) Branching factor = 30

(c) Branching factor = 40

Figure 6: (Garnet - Policy Evaluation) Behaviour of M-VC for a 50-state Garnet problem with $\gamma = 0.99$.

one may use the momentum over action-value function too. Instead of computing the greedy policy w.r.t. the most recent action-value function, MoVI computes the greedy policy w.r.t. the moving average of the previous action-value functions. The moving average is essentially the result of passing the past action-value functions through an integrator. To be more concrete, MoVI is defined by

$$\begin{aligned}
\pi_{k+1}(x) &= \operatorname*{argmax}_a z_k(x, \cdot), \qquad \forall x \in \mathcal{X} \\
Q_{k+1} &= T^{\pi_{k+1}} Q_k + \varepsilon_{k+1}, \\
z_{k+1} &= (1 - \beta_{k+1}) z_k + \beta_{k+1} Q_{k+1}.
\end{aligned}$$

The function $z : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ is the smoothed version of the previous action-value functions. The policy π_{k+1} is the greedy policy w.r.t. the smoothed version of Q_k, instead of Q_k itself. And the Bellman operator is w.r.t. this policy, instead of the greedy policy w.r.t. Q_k. Here, the function ε_{k+1} denotes the error in performing this iteration. In the case that we can do it exactly, it would be zero.

PI VI (12) and MoVI both use an integrator to construct a smoothed estimate of certain quantities. The quantity is the Bellman Residual for PI VI and is the action-value function for MoVI. The main feature of MoVI is the use of a greedy policy w.r.t. this smoothed action-value function instead of the current action-value function. MoVI does not directly use the smoothed value function to obtain a new value function, but uses it to perform the policy improvement step w.r.t. a smoothed version of the action-value function. In comparison, PI VI directly uses the integrated value function in the construction of the new value function.

The main motivation of MoVI is that by smoothing the action-value functions, the effect of errors ε_k at each iteration might be reduced. In the case of having no error (i.e., $\varepsilon_k = 0$ for all k), we get from Corollary 1 of Vieillard et al. [2020b] that

$$\|Q^{\pi_{k+1}} - Q^*\|_{1,\mu} \leq \frac{C(\mu) 2 Q_{\max}}{(1-\gamma)(k+1)},$$

where $C(\mu)$ is a concentrability coefficient. It is noticeable that we have $O(\frac{1}{k})$ behaviour, which is much slower than the exponential $O(\gamma^k)$ behaviour of the conventional VI or the PID variants of VI (upon proper choice of controller gains). Therefore, MoVI does not accelerate the VI process itself, but potentially reduces the effect of errors at each iteration of VI.

More recently, Vieillard et al. [2020a] propose and analyze the class of Dual Averaging Modified Policy Iteration (DA-MPI) algorithms, in the context of regularized MDPs [Geist et al., 2019]. MoVI would be a limiting case of DA-MPI. As in the case of MoVI, the use of smoothing in DA-MPI is for the computation of the greedy policy (or its KL regularized variant), and is not directly used in the computation of the value function as in PI VI. It is imaginable that one might combine ideas of DA-MPI or MoVI with PID VI.

A different approach to acceleration is through Anderson acceleration (AA) [Anderson, 1965]. AA is an acceleration method for solving fixed-point equations and was originally developed for solving integral equations. It has recently been used in the context of MDPs and RL. Geist and Scherrer [2018] propose the Anderson acceleration variant of VI. Their method finds the linear combination of previous value functions that minimizes the sum of weighted Bellman residuals and uses the best combination to construct a new value function. More concretely, for a given $m \geq 0$, it first computes the vector w

$$\begin{aligned} w_{k+1} = & \operatorname*{argmin}_{w \in \mathbb{R}^m} \left\| \sum_{i=0}^{m} w_i \mathrm{BR}(V_{k-i}) \right\| \\ \text{s.t.} \qquad & \sum_{i=0}^{m} w_i = 1, \end{aligned}$$

and then set the new value function as

$$Q_{k+1} = \sum_{i=0}^{m} w_i T V_{k-i}.$$

Shi et al. [2019] propose to use Anderson acceleration for policy iteration. Their method finds the best linear combination of previous action-value functions and uses the linearly combined action-value function for policy improvement.

G.1.3 Other RL Acceleration Methods

There are several other approaches to acceleration in RL and DP, which we briefly mention here. These methods are less similar to the ideas mentioned in this paper.

Some techniques for acceleration are based on changing the order or frequency of state updates in an asynchronous VI scheme [Bertsekas and Tsitsiklis, 1996, Section 2.2.2]. This is in contrast with this work, in which all states are synchronously updated and acceleration is potentially achieved by changing the dynamics at the iteration level. The main idea is that not all states are required to be updated at the same time, and some of them might be prioritized based on the impact of their update on the value function at other states. This idea is explored in variants of prioritized sweeping algorithms and other similar methods, which have been used in both RL and DP settings [Moore and Atkeson, 1993, Peng and Williams, 1993, Andre et al., 1997, McMahan and Gordon, 2005, Wingate and Seppi, 2005, Dai et al., 2011]. The prioritized sweeping-like algorithms, and more generally asynchronous updates, are orthogonal to what we proposed here. It is plausible that one can achieve even further acceleration by combining both ideas and have methods such as prioritized sweeping PID VI. This is an interesting research direction, which is beyond the scope of this work.

Another method is Speedy Q-Learning (SQL), which is an accelerated variant of Q-Learning [Azar et al., 2011]. It decomposes the update rule of Q-Learning in a particular way and uses a more aggressive learning rate, compared to Q-Learning, on one of its terms. SQL is an online algorithm, so is not directly comparable with the proposed methods. Looking at its underlying DP operator, however, is instructive. At iteration k of the algorithm, its operator is $L_k : \mathcal{B}(\mathcal{X} \times \mathcal{A}) \times \mathcal{B}(\mathcal{X} \times \mathcal{A}) \to \mathcal{B}(\mathcal{X} \times \mathcal{A})$

$$L_k[Q_k, Q_{k-1}] = T^* Q_k + (k-1) \left[T^* Q_k - T^* Q_{k-1} \right].$$

This has some high-level similarities to PD VI. The first term is the same as in the conventional VI (or the Proportional term in PD). The second term computes the difference between the effect of the Bellman optimality operators applied to Q_k and Q_{k-1}. This is similar to the Derivative term of PD VI. But there are two differences. The first is that instead of $Q_k - Q_{k-1}$ of PD VI, it uses $T^* Q_k - T^* Q_{k-1}$.[6] The second difference is that its corresponding D gain is $\kappa_d^{(k)} = (k-1)$ and grows as a function of the iteration, as opposed to the constant gain κ_d in our formulation (if we do not perform the gain adaptation).

We may consider acceleration in the context of sample-efficient RL methods such as LSTD and LSPI [Boyan, 1999, Lagoudakis and Parr, 2003, Lazaric et al., 2012]. LSTD is a sample-efficient algorithm, but is computationally more expensive compared to methods such as TD (or VI in the DP setting). The original fixed-point formulation of the LSTD and LSPI requires solving a linear system of equations, which is especially expensive if performed online. There are methods to improve the computational complexity of LSTD. These methods might be considered as acceleration methods, though with a slightly different interpretation of acceleration as our previous usage. Geramifard et al. [2006] introduce an incremental version of LSTD that benefits from the sparsity of features to improve its computational efficiency.

There are some other second-order methods that can improve the convergence of the TD method, which can be considered as a first-order method. Pan et al. [2017] use quasi-second-order gradient descent on the mean squared projected Bellman error, whose minimizer is the same as the solution obtained by the LSTD algorithm. Yao and Liu [2008] propose a class of preconditioned TD methods. Romoff et al. [2021] study a diagonal preconditioner in order to improve the convergence behaviour of TD-like update.

Devraj and Meyn [2017] propose a second-order stochastic approximation method, called Zap Q-Learning, with a matrix gain that minimizes the asymptotic variance of the value function estimate. Zap Q-Learning, an application of the Zap Stochastic Approximation (SA) algorithm to the RL problem, is accelerated in the sense that its asymptotic variance is superior to Q-Learning's, but it is computationally more expensive as it requires matrix inversion at every time step. We describe it and its extensions [Devraj et al., 2019] in some more detail as they have some connections with the P and PD variants of VI. Zap Q-Learning can be seen as a stochastic Newton-Raphson method. By ignoring the sample-based aspect of Zap Q-Learning, which admittedly is central to its appeal, and writing down its deterministic dynamics for PE with an exact value function representation (no function approximation), we have

$$V_{k+1} = V_k + \alpha(\mathbf{I} - \gamma \mathcal{P}^\pi)^{-1} \mathrm{BR}(V_k).$$

Comparing with $V_{k+1} = V_k + K_p \mathrm{BR}(V_k)$ of (9) (with a matrix gain K_p instead of κ_p), we see that the deterministic version of Zap Q-Learning corresponds to P VI with a matrix gain $K_p = \alpha(\mathbf{I} - \gamma \mathcal{P}^\pi)^{-1}$. The Zap Q-Learning algorithm estimates $\mathrm{BR}(V_k)$ using the TD error and constructs an estimate of $(\mathbf{I} - \gamma \mathcal{P}^\pi)$ online.

The matrix inversion required in Zap Q-Learning makes it a computationally expensive algorithm. Devraj et al. [2019] and Devraj [2019, Chapter 4] propose Polyak Stochastic Approximation (PolSA) and Nesterov's Stochastic Approximation (NeSA) as computationally cheaper SA algorithms. It can be shown that the idealized version of PolSA and Zap SA *couples*, i.e., their estimated parameters converge to each other with a fast rate (Proposition 4.2 of Devraj 2019).

PolSA is an extension of Polyak's heavy-ball momentum for stochastic approximation. The deterministic variant of PolSA (see Eq. (4-17) of Devraj 2019), used for computing the value function V^π with an exact representation of the value function (no function approximation), would be

$$V_{k+1} = (1-\eta)V_k + \eta T^\pi V_k + \left[(1-\eta)\mathbf{I} + \eta\gamma\mathcal{P}^\pi\right](V_k - V_{k-1}).$$

This is PD VI (11) with $\kappa_p = \eta$ and $K_d = (1-\eta)\mathbf{I} + \eta\gamma\mathcal{P}^\pi$. When $\eta = 1$, this simplifies to $\kappa_p = 1$ and $K_d = \gamma \mathcal{P}^\pi$:

$$V_{k+1} = V_k + \mathrm{BR}(V_k) + \gamma \mathcal{P}^\pi (V_k - V_{k-1}).$$

PolSA-based approach to estimate the value function uses the TD error to estimate $\mathrm{BR}(V_k)$ and constructs an estimate of $\mathcal{P}^\pi$ for the last term.

[6] We could also define the PD VI differently and have $Q_{k+1} = T^* Q_k + \kappa_d (T^* Q_k - T^* Q_{k-1})$ or $Q_{k+1} = T^* Q_k + \kappa_d (\mathrm{BR}^*(Q_k) - \mathrm{BR}^*(Q_{k-1}))$ instead of $Q_{k+1} = T^* Q_k + \kappa_d (Q_k - Q_{k-1})$ in (10), though we did not study these variations in this work.

Proposition 2 describes the dynamics with a matrix gain, though the simplified Corollary 3, as well as experiments, are for the scalar κ_d. It is also notable that even though Momentum VC of Goyal and Grand-Clement [2020] (53) is inspired from the heavy-ball method of Polyak, the resulting algorithm is different from the deterministic version of PolSA and has a scalar κ_d, instead of a matrix gain K_d that is a function of $\mathcal{P}^\pi$. We emphasize that Zap Q-Learning and PolSA are designed as SA algorithms with a concern to have a small asymptotic variance of the estimated parameters, and their starting point was not the acceleration of computation of the value function in the known model setting, as is in this paper. Comparing their deterministic counterparts and PID VI, however, might shed some light on how we can design RL algorithms based on PID VI.

G.2 Control Theoretic Interpretation of Optimization Techniques

Our work made a connection between VI and a simple dynamical system, and suggested using simple controllers to modify the dynamics. We would like to note that the connection between control theory/dynamical system viewpoint and the gradient-based optimization literature has been recognized. Although VI is not an optimization algorithm in the same sense, revealing the connection between similar fields might be instructive. As a few recent examples, Hu and Lessard [2017] show that optimizers such as heavy-ball and Nesterov's accelerated method can be interpreted as Lag controller or Lag + PID controller. Lessard et al. [2016] use a method from robust control (Integral Quadratic Constraints (IQC)) to provide upper bounds on the convergence rate of several optimization methods. Muehlebach and Jordan [2019] show that the Nesterov's acceleration can be interpreted as the semi-implicit Euler discretization of a continuous-time mass-spring-damper system with a curvature-dependent damping. Benosman et al. [2020] derive optimization algorithms based on discretization of continuous-time rescaled and signed gradient flow, and analyze their convergence using results from hybrid dynamical control theory.

G.3 Learning Rate Adaptation

Recall that the basic idea of the gain adaptation was to compute the gradient of an appropriately-defined loss function w.r.t. the controller gains, and to update them based on the gradient throughout the iterations of the accelerated VI algorithm. Tuning the learning rate by computing the gradient of loss w.r.t. the learning rate is not a new idea, and has been suggested several times in the machine learning community, see e.g., the 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]. Interestingly, the idea of tuning the hyper-parameters of an algorithm using a gradient descent-like procedure has been explored much earlier in late 1950s and early 1960s in the adaptive control literature, and is known as the MIT rule [Osburn et al., 1961].[7]

H Detail of Experiments

H.1 Chain Walk Problem

The chain walk problem with $\mathcal{X} = \{0, \dots, N-1\}$ and $\mathcal{A} = \{\text{Right}, \text{Left}\}$ is similar to the problem used by Lagoudakis and Parr [2003]. It is a circular chain, where the state 0 and $N-1$ are connected. Upon taking action Right at state $x \in \mathcal{X}$, the agent goes to its right state $(x+1) \bmod N$ with probability 0.7, stays in the same state x with probability 0.2, and goes to its left state $(x-1) \bmod N$ with probability of 0.1. The Left action is similar, with a reversed probability. The reward function is state-dependent and is

$$r(x) = \begin{cases} -1 & x = 10 \\ +1 & x = N - 10 \\ 0 & \text{otherwise} \end{cases}$$

[7] We could not find and read the paper by Osburn et al. [1961] in our search, so we are relying on secondary sources for this reference, e.g., Mareels et al. [1987] and Chapter 5 (Model-Reference Adaptive Systems) of Aström and Wittenmark [1994].

(assuming that $N > 10$). For our experiments, the number of states are $N = 50$, so the non-zero rewards are at states 10 and 40.

H.2 Random MDP (Garnet)

We define a variation of the class of Garnet problems here. The Garnet problems are a class of randomly generated finite MDPs [Bhatnagar et al., 2009]. The name is an acronym for Generic Average Reward Non-stationary Environment Testbed. Even though our formulation is neither for the average reward nor non-stationary problems, we refer to the resulting MDP as Garnet anyway.

Our Garnet problem is described by parameters $(|\mathcal{X}|, |\mathcal{A}|, b_P, b_r)$. Here $|\mathcal{X}|$ is the number of states and $|\mathcal{A}|$ is the number of actions. The parameter b_P describes the branching factor for each state-action pairs. That is, for each pair $(x, a) \in \mathcal{X} \times \mathcal{A}$, the transition vector $\mathcal{P}(\cdot|x, a)$ has b_P non-zero values. The probability of going to the next-state is determined as follows: The possible b_P next-states are chosen from $\mathcal{X}$ randomly without replacement. We then uniformly randomly choose $b_P - 1$ values in $(0, 1)$; these points act as cuts of the unit interval, which leads to b_P partitions. The size of each partition is the probability for the selected b_P next-states. The rest of $|\mathcal{X}| - b_P$ next-states have the probability of zero. We repeat this process for all state-action pairs. The reward function $r : \mathcal{X} \to \mathbb{R}$ is only state dependent. We randomly pick b_r states without replacement. For each selected state x, we set $r(x)$ as a random number drawn from a uniform distribution between $(0, 1)$.

I Experiments (Extended)

This section follows a similar structure to Section 6 and complements its results with more comprehensive empirical results and discussions.

I.1 Experiments with Controller Gains

The error dynamics of the new variants of the VI algorithm depends on the eigenvalues of matrices A_{PD}, A_{PI}, and A_{PID}. If the eigenvalues are strictly within the unit circle, the dynamical system is asymptotically exponentially stable, and the procedure is convergent. The convergence rate depends on how close the eigenvalues are to zero. To accelerate the convergence of the modified VI algorithm, we have to choose the controller gains such that the eigenvalues move closer to 0. In this section, we use some simple experiments to illustrate how these eigenvalues move around as the controller gains are changed, and we observe their effect on the error behaviour of the modified variants of VI. These experiments complement the results of Proposition 4 and Corollary 5 (Appendix B), which indicate the dependence of the location of the eigenvalues of the PD, PI, and PID variants as a function of the gains and the eigenvalues λ_i of $\mathcal{P}^\pi$. We would like to note that even though the modulus of the dominant eigenvalue determines the asymptotic behaviour of the dynamical system, the transient behaviour depends on the whole eigenstructure of the matrix.

We use a chain walk problem with 50 states in this section. The details of the dynamics are described in Appendix H. We consider both policy evaluation (using T^π) and control (using T^*) cases. In all experiments, we set $\gamma = 0.99$, unless stated otherwise.

In the first experiment in Section 6, we showed a typical behaviour of VI and accelerated VI with different controller gains by reporting $\log_{10} \|V_k - V^\pi\|_\infty$ (PE) and $\log_{10} \|V_k - V^*\|_\infty$ (control) as a function of iteration k (Figure 1). The norm of error provides an overall indicator of the error behaviour. We also take a closer look at the error of each state to get a better sense on how the value function evolves as a function of iteration.

Figure 7 and Figure 8 depict how the error $e_k(x) = V_k(x) - V^\pi(x)$ (or $e_k(x) = V_k(x) - V^*(x)$ for the control case) converge to zero for a select number of states (since we have 50 states in this problem, we only show a few regularly space states in order to avoid excessive clutter) for the same gains as in Figure 1.

Figure 7 presents the result for the PE problem. We observe the oscillatory behaviour of the errors $e_k(x) = V_k(x) - V^\pi(x)$ for all methods, including the conventional VI. Even though the standard norm contraction argument shows that the norm of the error should monotonically decrease and behave as

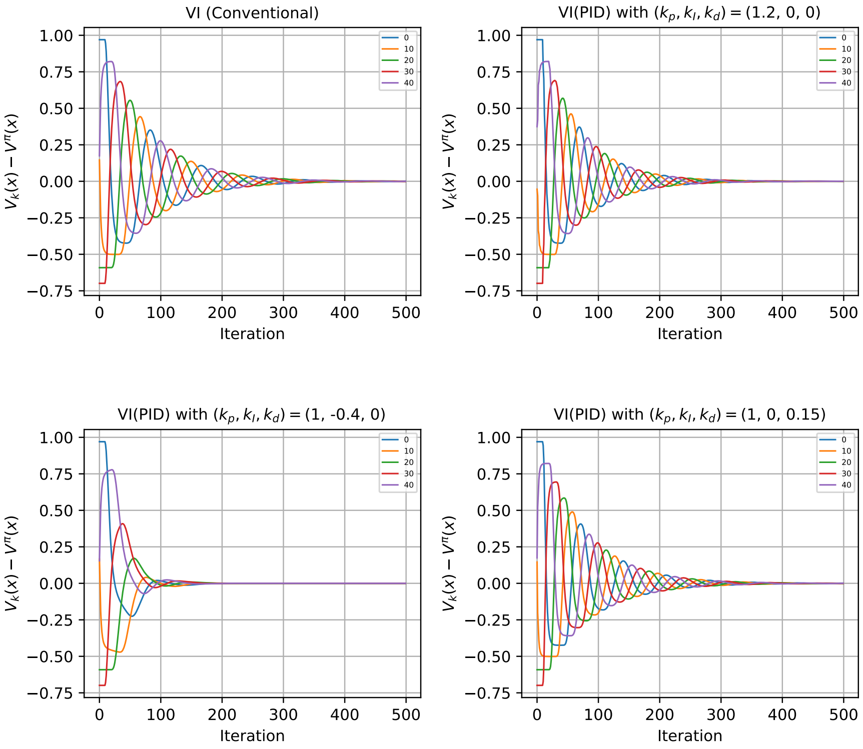


Figure 7: (Chain Walk - Policy Evaluation) Error behaviour of the value function for a 50-state chain walk problem for various accelerated variants of VI and the conventional one.

$\|e_k\|_\infty \leq \gamma^k \|e_0\|_\infty$, it does not give much insight about the fine behaviour of the error at each state.[8] We see that it is possible, as in this problem, that the error at each state have both increasing and decreasing periods, while the norm over the whole state space is decreasing. The decrease in norm is apparent in the envelop of these curves. Comparing the conventional VI and the accelerated variants, we observe the modified behaviour, most noticeably in the PI case, in which the errors dampens quickly and the oscillatory behaviour is reduced too.

Figure 8 presents the result for the control case. No oscillatory behaviour is observed here anymore, and the error is monotonically reducing for most cases. The accelerated variants reduce the dampening time. For the $(\kappa_p, \kappa_I, \kappa_d) = (1, 0.75, 0.4)$ case, we observe some temporarily increase in error of some states.

In Section 6, we also studied the effect of changing each of the controller gains on the performance for both PE and Control (Figure 2). Here we complement those results by (1) looking at how the eigenvalues changes as we change gains, and (2) studying the change of two gains at the same time (visualizing the effect of changing three gains at the same time is difficult).

Figure 9 shows the root locus diagram, the location of the eigenvalues of the error dynamics matrices as the gain changes. This is only shown for PE as the eigenvalues of a dynamics matrix as a descriptor of the dynamics is most meaningful for linear time-invariant system. For the control case, the dynamical system is linear, but time-varying. See Appendix C for more discussion. We observe that as the controller gains change, the location of the eigenvalues change too. The behaviour is quite complicated in the case of PD controller. We observe that the change in the locations of some eigenvalues are more significant than others,

[8] Recall that we have $\|e_k\|_\infty = \|V_k - V^\pi\|_\infty = \|T^\pi V_{k-1} - T^\pi V^\pi\|_\infty \leq \gamma \|V_{k-1} - V^\pi\|_\infty = \gamma \|e_{k-1}\|_\infty$ by the contraction of the Bellman operator. This shows that $\|e_k\|_\infty \leq \gamma^k \|e_0\|_\infty$.

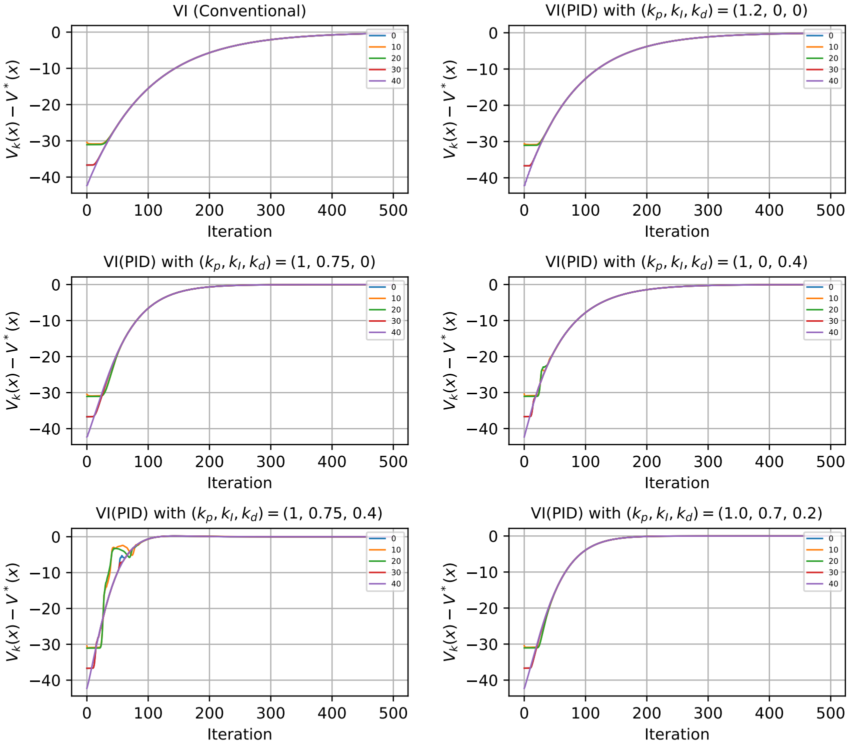


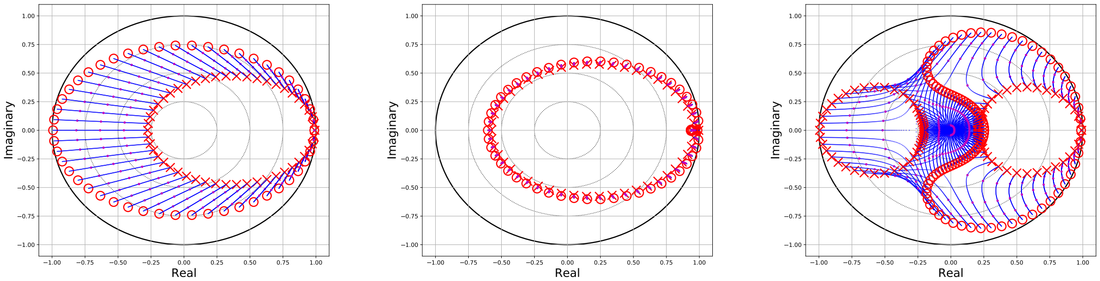
Figure 8: (Chain Walk - Control) Error behaviour of the value function for a 50-state chain walk problem for various accelerated variants of VI and the conventional one.

while for some others the change is barely noticeable. This is especially the case with eigenvalues close to +1. This, however, does not mean that the dynamics is not changing. Reducing the dominant eigenvalue from 1 to, say, 0.98 has a significant effect on the dynamics, as it changes the asymptotic behaviour from $O(\gamma^k)$ to $O((0.98\gamma)^k)$.

The previous experiments studied variation of one parameter only. Different gains of a PID controller, however, can have complementary effect on the dynamics. Together they might lead to better performance, an example of which is the PID controller $(\kappa_p, \kappa_I, \kappa_d) = (1, 0.75, 0.4)$ in Figure 1, which behaves better than the PD or PI controllers described with $(\kappa_p, \kappa_I, \kappa_d) = (1, 0.75, 0)$ or $(\kappa_p, \kappa_I, \kappa_d) = (1, 0, 0.4)$. To investigate this complementary effect, we now study changing two gains at the same time, while keeping the third constant. We sweep over (κ_p, κ_d) pairs (corresponding to PD controllers), (κ_p, κ_I) pairs (corresponding to PI controllers), and (κ_I, κ_d) (corresponding PID controllers). We use the default values of $\kappa_p = 1$, $\kappa_d = \kappa_I = 0$ for the constant gain. So we are effectively sweeping over a subset of PID controllers. Note that the integrator in PID also has two parameters β and α. We do not change them here, and leave them as before, i.e., $\beta = 0.95$ and $\alpha = 0.05$.

We use $V_k^{\text{PID}}(\kappa_p, \kappa_I, \kappa_d)$ to indicate the approximated value of such a method with a particular choices of gains at the k-th iteration. Here we use V^{PID}, even though when κ_d or κ_I are equal to zero, the corresponding controllers are in fact PI or PD, respectively. For the control case, we use $V_k^{\text{PID}} = \max_a Q_k^{\text{PID}}(\cdot, a)$ and the comparison is with the optimal value function V^*.

Instead of reporting the log error $\log_{10} \|V_k(\kappa_p, \kappa_I, \kappa_d) - V^\pi\|_\infty$ as a function of controller gains, which is



(a) P Controller

(b) PI Controller

(c) PD Controller

Figure 9: (Chain Walk) (Policy Evaluation) The eigenvalue location (root locus) of the error dynamics matrix as one of the controller gains are changed for a 50-state chain walk problem. The ranges are $\kappa_p \in [0.8, 1.2]$, $\kappa_I \in [-0.5, 0.3]$, and $\kappa_d \in [-0.2, 0.25]$, and their starts are indicated by × and their ends by ∘. The small purple dots indicate $[\frac{1}{4}, \frac{1}{2}, \frac{3}{4}]$ in the range.

somewhat difficult to inspect visually, we report the log of error relative to the conventional VI, i.e.,

$$\log_{10} \frac{\left\| V_k^{\text{PID}}(\kappa_p, \kappa_I, \kappa_d) - V^\pi \right\|_\infty}{\left\| V_k^{\text{conv}} - V^\pi \right\|_\infty}.$$

Figures 10 reports the result at the final iteration for the PE case. The log relative error is 0 when the performance of $V_k^{\text{PID}}(\kappa_p, \kappa_I, \kappa_d)$ is the same as the conventional VI (and is indicated by white pixels in the figure), is negative (hues of blue) when it outperforms, and positive (hues of red) when it underperforms. Some choices of gains might lead to unstable behaviour, and hence large relative error. In these figures, we cap the value at +1, which means that if the error is 10-times worse than the conventional VI, we use the same colour (red) to indicate it. Figures 11, 12, and 13 report the result for each pairs of sweeping dimensions at $k = [\frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1] \times K$ iterations, where K is the total number of iterations (so the last iteration is the same as the figures in Figures 10). In all figures, we see a parameter region that accelerates VI. As expected, these regions are different for different combinations of controller gains.

Figure 10a focuses on the PD controller. When $\kappa_p = 1$, and κ_d varies, the colours are close to white, showing that the performance is not better than the conventional VI. Interestingly, there is a region with much better performance when $\kappa_p \approx 1.2$. The best relative error is around $10^{-0.6} \approx 0.25$. We observe a high performing ridge in Figures 10b, surrounded by a large region of improved performance. Figure 10c show less sensitivity to changes in κ_d. The best performance of PI and PID controllers are better than the PD one, which is aligned with our previous observation.

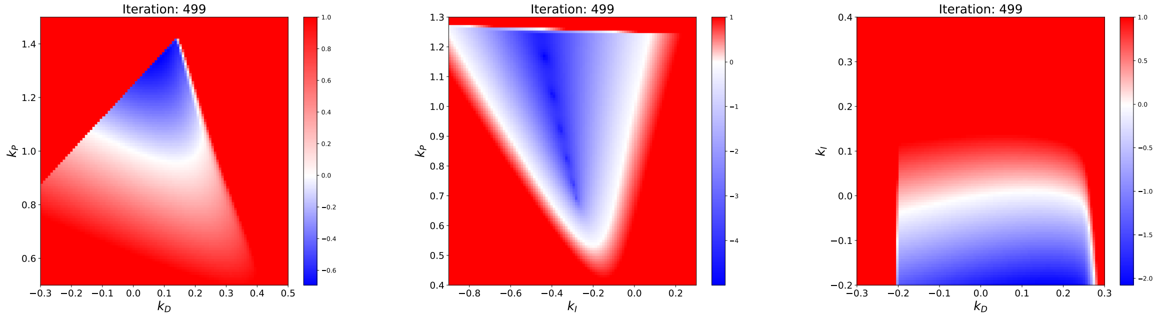
Figures 11, 12, and 13 show that log relative error at different iterations. The general shape of the regions are the same, with some slight variations between iterations.

Figure 14 shows the effective planning horizon. The effective planning horizon is defined as $\frac{1}{1-\gamma_{\text{eff}}}$, where γ_{eff} is the effective discount factor for a particular value of controller gains. The effective discount factor is computed as a fit of γ_{eff}^K to the error after K iteration, i.e.,

$$\gamma_{\text{eff}} = \exp\left(\frac{\ln\left(\frac{\| V_{K-1} - V^\pi \|_\infty}{\| V_0 - V^\pi \|_\infty} \right)}{K} \right),$$

or similar for the control case.

The same results for the control case appear in Figure 15 (the last iteration), and Figures 16, 17, 18 (for several iterations), and Figure 19 for the effective planning horizon. The general shape of regions are different compared to the PE case. Also we observe sudden changes in the colour, which corresponds to sudden changes in performance. This is similar to the large drop in error in the PI case (Figure 2e). We do not currently have a good explanation for it.



(a) PD Controllers ($\kappa_p \times \kappa_d$)

(b) PI Controllers ($\kappa_p \times \kappa_I$)

(c) PID Controllers ($\kappa_I \times \kappa_d$)

Figure 10: (Chain Walk - Policy Evaluation) Log relative error as two of the controller gains are changed for a 50-state chain walk problem.

The takeaway message of these experiments on simultaneous change of two gains is that we may achieve even faster acceleration compared to when we only change one of the controller gains. In many cases, the region that leads to a faster convergence is not small, indicating that the dynamics is not overly sensitive to the right choice of controller gains. As discussed in Appendix D, there are several ways to choose these gains. One is to consider them as hyper-parameters of the algorithm and tune them from an outer loop. Another is to select them through the gain adaptation procedure of Section 5. This is what we empirically evaluate in some detail in the next section.

I.2 Experiments with Gain Adaptation

In this section, we report some empirical results on using the gain adaptation method of Section 5. The results of this section are more comprehensive compared to Section 6.2. We study two problems. The first is the same 50-state chain walk problem as in Appendix I.1. We would like to see if the gain adaptation procedure can lead to acceleration or not. The second is the Garnet problem, a class of random MDPs [Bhatnagar et al., 2009] (see detail in Appendix H.2). We adapt κ_p, κ_I, and κ_d by (22), or similar 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.

I.2.1 Gain Adaptation for Random Walk

Let us start with the discount factor $\gamma = 0.99$. We compare the conventional VI and an accelerated PID VI with adaptive gains. For the adaptive gain, we start from $(\kappa_p, \kappa_I, \kappa_d) = (1, 0, 0)$, which makes it the conventional VI procedure at the beginning. The behaviour of the adaptive algorithm depends on the meta-learning rate η and the normalizing factor ε. For this experiment, we manually tried several values, but did not systematically optimize it. We choose $(\eta, \varepsilon) = (0.05, 10^{-20})$ for both the PE and control cases, even though their best values are not necessarily the same. We note that if η is too large, the procedure might become unstable. In the next section, we provide a systematic study of the effect of these meta-learning parameters on the performance.

Figure 20 reports the results. Figure 20a compares the error behaviour of conventional VI and the adaptive procedure. We see that the adaptive accelerated VI decreases error significantly faster than the conventional VI. Figure 20b shows how the gains evolve throughout iterations. We observe that κ_p becomes larger than 1, κ_I becomes negative, and κ_d becomes positive. Figure 21 reports the result for the control case. The performance improves significantly compared to the conventional VI. This figure is the same as Figure 3 in Section 6.2, which we repeat here for easier access.

We now repeat the same experiments with a larger discount factor of $\gamma = 0.999$. We use the same values of $(\eta, \varepsilon) = (0.05, 10^{-20})$. The results are reported in Figure 22 (PE) and Figure 23 (Control). The results show a significant speedup as a result of gain adaptation.

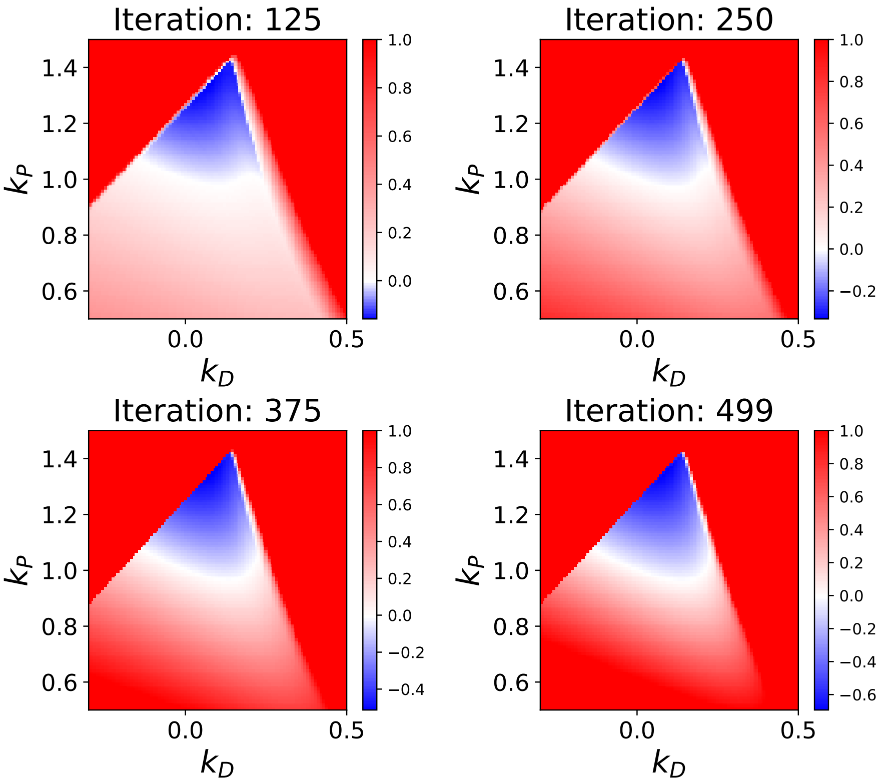


Figure 11: (Chain Walk - Policy Evaluation - $\kappa_d \times \kappa_p$) Log relative error as two of the controller gains are changed for a 50-state chain walk problem at several iterations.

I.2.2 Gain Adaptation for Random MDPs

The use of a gain adaptation procedure is most useful when we are facing an unknown problem for which we do not know a good choice of controller gains. The results on the random walk problem were encouraging as they showed that the gain adaptation procedure could lead to acceleration for that particular problem. To show the real benefit of gain adaptation, however, it is better to evaluate it on a wide range of MDPs. We use the Garnet problem, which is a randomly generated MDP, as the testbed. For the result of this section, we consider a Garnet problem with 50 states, a branching factor of 3, and 5 non-zero rewards throughout the state space. For the PE problem, we effectively have 1 action per state, and for the control problem we have 4 actions per state. The branching factor is per state-action pair, i.e., each state-action pair is connected to 3 other states in this setup. We consider the discount factor of $\gamma = 0.99$ in these experiments. We report the mean and standard error of performance over 20 runs.

Figures 24 and 25 compare the performance of the gain adaptation procedure for the PE problem. Each subfigure is for a fixed meta-learning rate η and varying normalizing coefficient ε. We also present the performance of the conventional VI. We present the mean and the standard error (shaded area around each curve) for each combination. As the Y-axis is logarithmic, the standard error is asymmetric. We also remark that an apparently wide standard error in a very small range of values can actually be much smaller than an apparently narrower one in larger range.

We observe that most combinations of η and ε lead to acceleration compared to the conventional VI. For each η, there is a wide range of ε (several orders of magnitude) that leads to good performance. We also notice that whenever ε is large (0.1 or 0.01), the acceleration is less significant compared to when ε is a small value. This can be understood by noticing that when ε is much larger than $\|\mathrm{BR}(V_{k-1})\|_2^2$ in (22), the

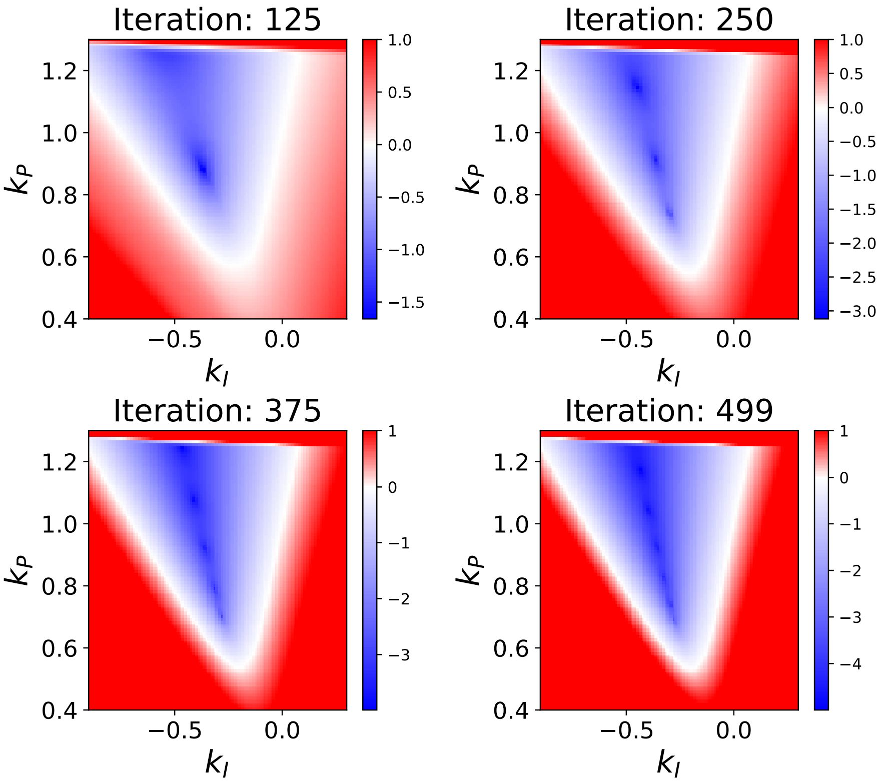


Figure 12: (Chain Walk - Policy Evaluation - $\kappa_I \times \kappa_p$) Log relative error as two of the controller gains are changed for a 50-state chain walk problem at several iterations.

normalized gradient would be too small to significantly change the controller gains. We also observe that smaller εs usually lead to better performance, though the relation is not always monotonic. For example, $\varepsilon = 10^{-20}$ or 10^{-16} are often one of the better performing ones for each η, but not always the best.

In these figures, we have a curve called "best". This refers to the average performance of the best choice of ε for each individual MDP. That is, for each random MDP, we select the best ε among the set $\{10^{-20}, 10^{-16}, 10^{-12}, 10^{-10}, 10^{-8}, 10^{-6}, 10^{-4}, 10^{-2}, 10^{-1}\}$. This shows the performance if we could do a model selection over the hyper-parameter ε of the gain adaptation procedure. The selection criteria is based on the average over the iterations of the logarithm of the error, i.e.,

$$\operatorname*{argmin}_{\theta} \frac{1}{K} \sum_{k=1}^{K} \log_{10} \left(\|V_k(\theta) - V^{\pi}\|_{\infty} + 10^{-20} \right),$$

where θ is the hyper-parameter of the gain adaptation procedure (ε in this case). We add a small number 10^{-20} in order to avoid numerical errors. For the control case, which we shall discuss soon, the optimal value function V^* replaces V^{π}. Note that this way of doing model selection is not practical, as it requires the knowledge of the true value function. But it shows that if we found the right hyper-parameter for each problem, what we could expect.

Figure 26 and 27 show the same experiment, but for the control case. The general observations are similar to the PE case, with a noticeable difference that when the meta-learning rate is large (e.g., 0.2 or 0.5), the gain adaptation procedure has a poor performance and does not lead to a procedure with decreasing error.

Next we report the result of similar experiment, but with the role of η and ε exchanged, i.e., we keep the normalizing constant ε fixed, and change the meta-learning rate η. Figure 28 depicts the result for the PE

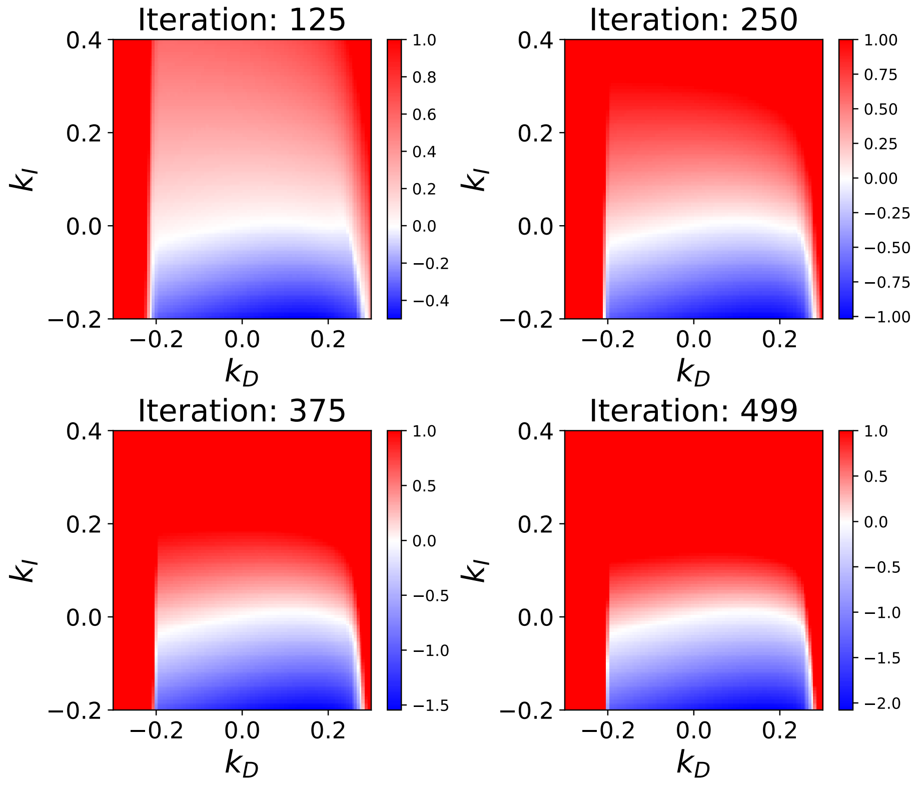


Figure 13: (Chain Walk - Policy Evaluation - $k_D \times \kappa_I$) Log relative error as two of the controller gains are changed for a 50-state chain walk problem at several iterations.

case, and Figure 29 depicts them for the control case. 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 meta-learning rates, e.g., $\eta = 10^{-3}$. For the control case, we also observe significant acceleration, except for the large value of the meta-learning rate $\eta = 0.1$.

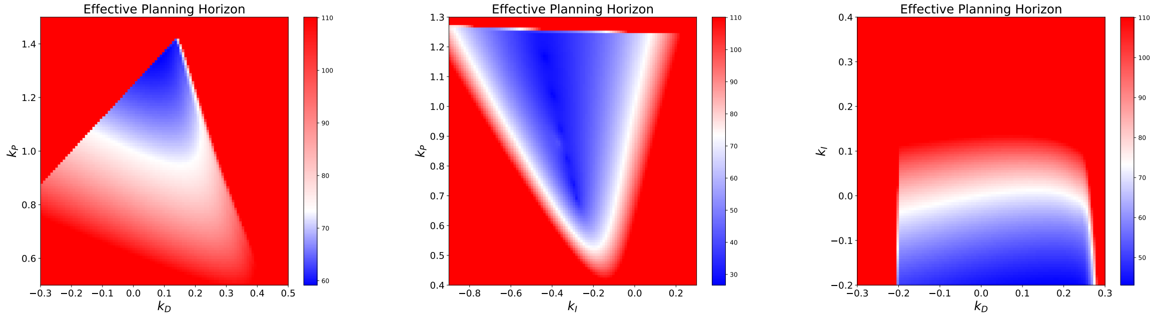
These results show that the gain adaptation procedure, with appropriately chosen parameters (η, ε), can lead to acceleration, and often a significant one. Although the gain adaptation procedure itself is not parameter-free, we observed that it is relatively insensitive to the choice of parameters. Our experiments suggest that a very small ε and small enough η usually work fine.

Acknowledgements

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

References

Luís B. Almeida, Thibault Langlois, Jose D. Amaral, and Alexander Plakhov. *Parameter Adaptation in Stochastic Optimization*, pages 111–134. Publications of the Newton Institute. Cambridge University Press, 1999. 7, 35

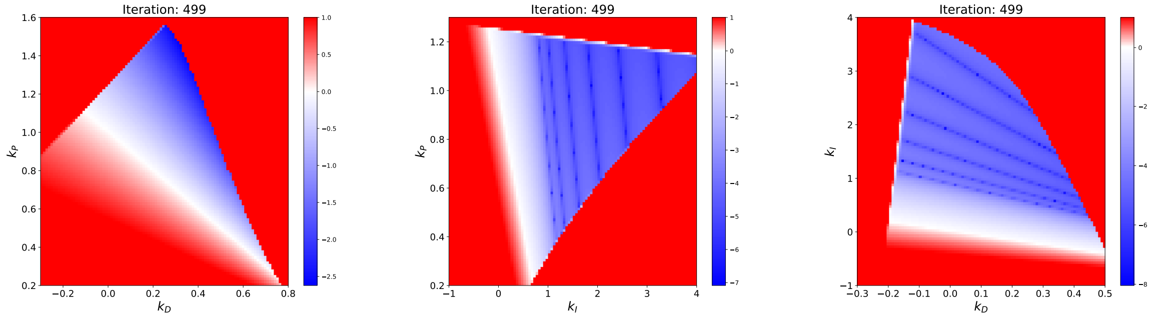


(a) PD Controllers ($\kappa_p \times \kappa_d$)

(b) PI Controllers ($\kappa_p \times \kappa_I$)

(c) PID Controllers ($\kappa_I \times \kappa_d$)

Figure 14: (Chain Walk - Policy Evaluation) Effective planning horizon as two of the controller gains are changed for a 50-state chain walk problem at several iterations.



(a) PD Controllers ($\kappa_p \times \kappa_d$)

(b) PI Controllers ($\kappa_p \times \kappa_I$)

(c) PID Controllers ($\kappa_I \times \kappa_d$)

Figure 15: (Chain Walk - Control) Log relative error as two of the controller gains are changed for a 50-state chain walk problem.

Donald G Anderson. Iterative procedures for nonlinear integral equations. *Journal of the ACM*, 12(4): 547–560, 1965. 12, 29, 33

David Andre, Nir Friedman, and Ronald Parr. Generalized prioritized sweeping. *Advances in Neural Information Processing Systems (NeurIPS)*, 10:1001–1007, 1997. 33

András Antos, Csaba Szepesvári, and Rémi Munos. Learning near-optimal policies with Bellman-residual minimization based fitted policy iteration and a single sample path. *Machine Learning*, 71:89–129, 2008. 29

Karl Johan Aström and Björn Wittenmark. *Adaptive Control*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 1994. 4, 29, 35

Mohammad Azar, Rémi Munos, Mohammad Ghavamzadeh, and Hilbert Kappen. Speedy Q-learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2411–2419, 2011. 13, 33

Atilim Gunes Baydin, Robert Cornish, David Martinez Rubio, Mark Schmidt, and Frank Wood. Online learning rate adaptation with hypergradient descent. In *International Conference on Learning Representations (ICLR)*, 2018. 7, 35

Mouhacine Benosman, Orlando Romero, and Anoop Cherian. Optimizing deep neural networks via discretization of finite-time convergent flows. *arXiv:2010.02990*, October 2020. 35

Vasile Berinde. *Iterative approximation of fixed points*, volume 1912. Springer, 2007. 5, 30

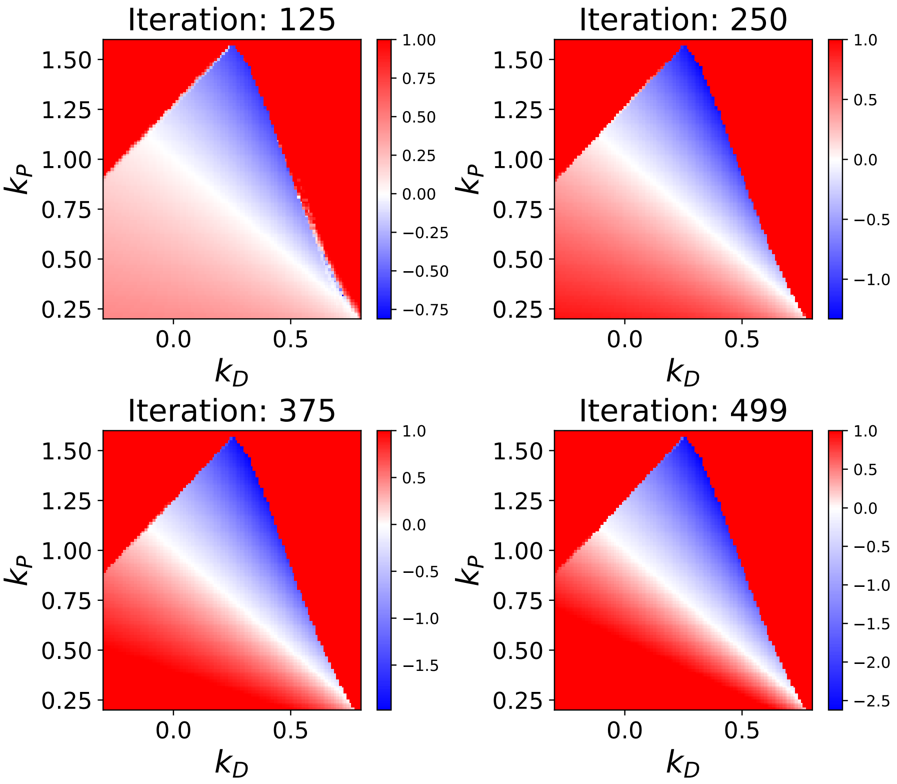


Figure 16: (Chain Walk - Control - $\kappa_d \times \kappa_p$) Log relative error as two of the controller gains are changed for a 50-state chain walk problem at several iterations.

Dimitri P. Bertsekas and Steven E. Shreve. *Stochastic Optimal Control: The Discrete-Time Case*. Academic Press, 1978. 4, 13

Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996. 2, 3, 13, 33

Shalabh Bhatnagar, Richard S. Sutton, Mohammad Ghavamzadeh, and Mark Lee. Natural actor–critic algorithms. *Automatica*, 45(11):2471–2482, 2009. 11, 36, 40

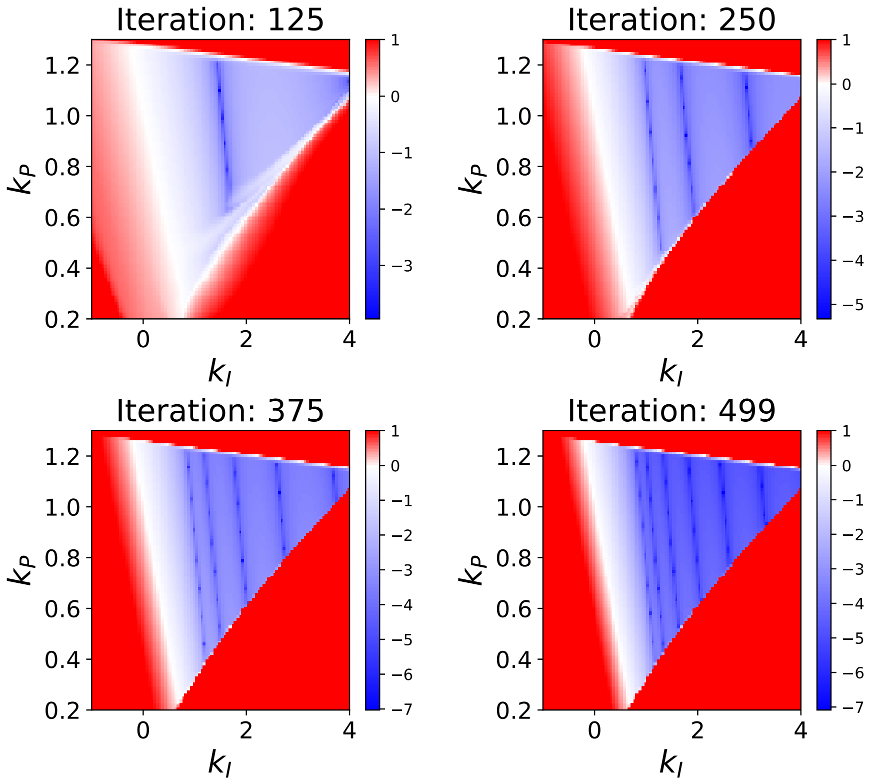
Vincent D. Blondel and John N. Tsitsiklis. The boundedness of all products of a pair of matrices is undecidable. *Systems & Control Letters*, 41(2):135–140, 2000. 19

Justin A. Boyan. Least-squares temporal difference learning. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML)*, pages 49–56. Morgan Kaufmann, 1999. 34

Jeffrey B. Burl. *Linear Optimal Control: H_2 and H_∞ Methods*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1998. 4

Jinglin Chen and Nan Jiang. Information-theoretic considerations in batch reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019. 2

Peng Dai, Mausam, Daniel S. Weld, and Judy Goldsmith. Topological value iteration algorithms. *Journal of Artificial Intelligence Research (JAIR)*, 42:181–209, 2011. 33


Iteration: 125

Iteration: 250

Iteration: 375

Iteration: 499

Figure 17: (Chain Walk - Control - $\kappa_I \times \kappa_p$) Log relative error as two of the controller gains are changed for a 50-state chain walk problem at several iterations.

Adithya M. Devraj. *Reinforcement Learning Design with Optimal Learning Rate.* PhD thesis, University of Florida, 2019. 34

Adithya M. Devraj and Sean P. Meyn. Zap Q-learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2235–2244, 2017. 13, 29, 34

Adithya M. Devraj, Ana Bušić, and Sean P. Meyn. Optimal matrix momentum stochastic approximation and applications to Q-learning. *arXiv:1809.06277v2*, February 2019. 34

Richard C. Dorf and Robert H. Bishop. *Modern control systems.* Prentice Hall, 2008. ISBN 9780132270281. 4

Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research (JMLR)*, 6:503–556, 2005. 2

Amir-massoud Farahmand, Mohammad Ghavamzadeh, Csaba Szepesvári, and Shie Mannor. Regularized fitted Q-iteration for planning in continuous-space Markovian Decision Problems. In *Proceedings of American Control Conference (ACC)*, pages 725–730, June 2009. 2

Amir-massoud Farahmand, Rémi Munos, and Csaba Szepesvári. Error propagation for approximate policy and value iteration. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems (NeurIPS - 23)*, pages 568–576. 2010. 22

Matthieu Geist and Bruno Scherrer. Anderson acceleration for reinforcement learning. In *European workshop on Reinforcement Learning (EWRL)*, 2018. 11, 12, 29, 33

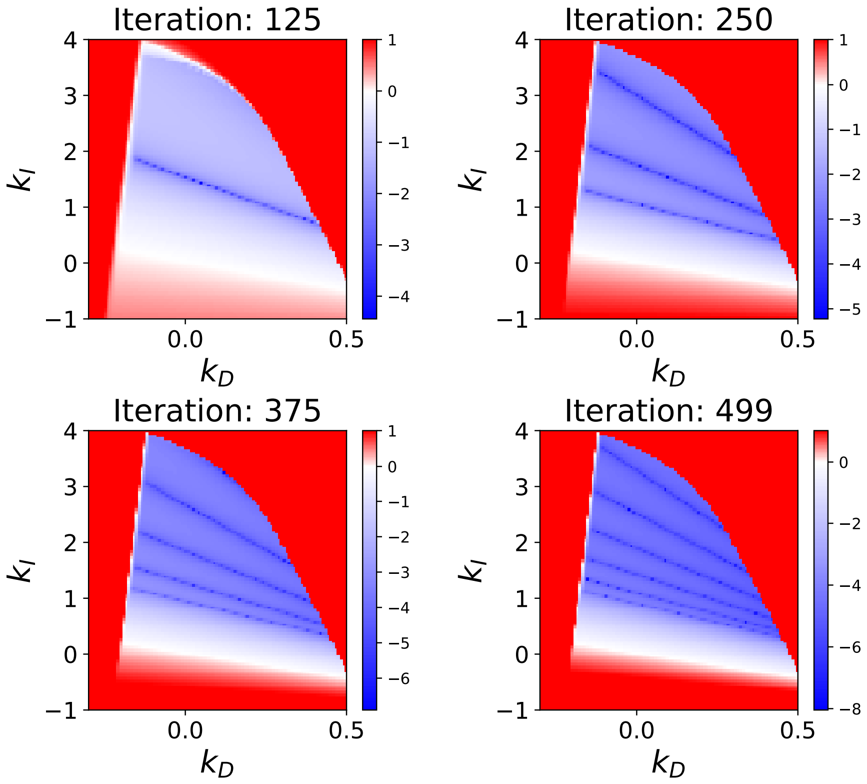


Figure 18: (Chain Walk - Control - $k_D \times \kappa_I$) Log relative error as two of the controller gains are changed for a 50-state chain walk problem at several iterations.

Matthieu Geist, Bruno Scherrer, and Olivier Pietquin. A theory of regularized Markov decision processes. In *International Conference on Machine Learning (ICML)*, 2019. 33

Alborz Geramifard, Michael Bowling, and Richard S. Sutton. Incremental least-squares temporal difference learning. In *Proceedings of the 21st American Association for Artificial Intelligence (AAAI)*, pages 356–361, 2006. 34

Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The John Hopkins University Press, 4th edition, 2013. 23

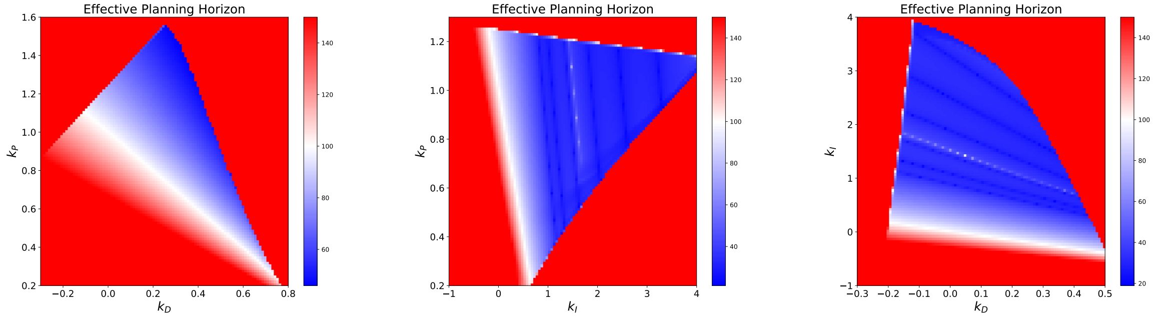
Vineet Goyal and Julien Grand-Clement. A first-order approach to accelerated value iteration. *arXiv:1905.09963v6*, March 2020. 2, 11, 12, 18, 21, 29, 30, 31, 35

Bin Hu and Laurent Lessard. Control interpretations for first-order optimization methods. In *American Control Conference (ACC)*, pages 3114–3119, May 2017. 35

Raphaël M. Jungers. *The Joint Spectral Radius: Theory and Applications*. Springer-Verlag Berlin Heidelberg, 2009. 7, 18, 19

Hassan K. Khalil. *Nonlinear Systems (3rd Edition)*. Prentice Hall, 2001. 4, 18

Miroslav Krstic, Ioannis Kanellakopoulos, and Petar V. Kokotovic. *Nonlinear and adaptive control design*. Wiley New York, 1995. 4



(a) PD Controllers ($\kappa_p \times \kappa_d$)

(b) PI Controllers ($\kappa_p \times \kappa_I$)

(c) PID Controllers ($\kappa_I \times \kappa_d$)

Figure 19: (Chain Walk - Control) Effective planning horizon as two of the controller gains are changed for a 50-state chain walk problem at several iterations.

Harold J. Kushner and Allan J. Kleinman. Accelerated procedures for the solution of discrete Markov control problems. *IEEE Transactions on Automatic Control*, 16(2):147–152, April 1971. 12, 30

Michail G. Lagoudakis and Ronald Parr. Least-squares policy iteration. *Journal of Machine Learning Research (JMLR)*, 4:1107–1149, 2003. 9, 29, 34, 35

Alessandro Lazaric, Mohammad Ghavamzadeh, and Rémi Munos. Finite-sample analysis of least-squares policy iteration. *Journal of Machine Learning Research (JMLR)*, 13:3041–3074, October 2012. 34

Laurent Lessard, Benjamin Recht, and Andrew Packard. Analysis and design of optimization algorithms via integral quadratic constraints. *SIAM Journal on Optimization*, 26(1):57–95, 2016. 35

David J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003. 21, 31

Ashique Rupam Mahmood, Richard S. Sutton, Thomas Degris, and Patrick M. Pilarski. Tuning-free step-size adaptation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2121–2124, 2012. 7, 35

Paul-Emile Maingé. Convergence theorems for inertial KM-type algorithms. *Journal of Computational and Applied Mathematics*, 219(1):223–236, 2008. 30

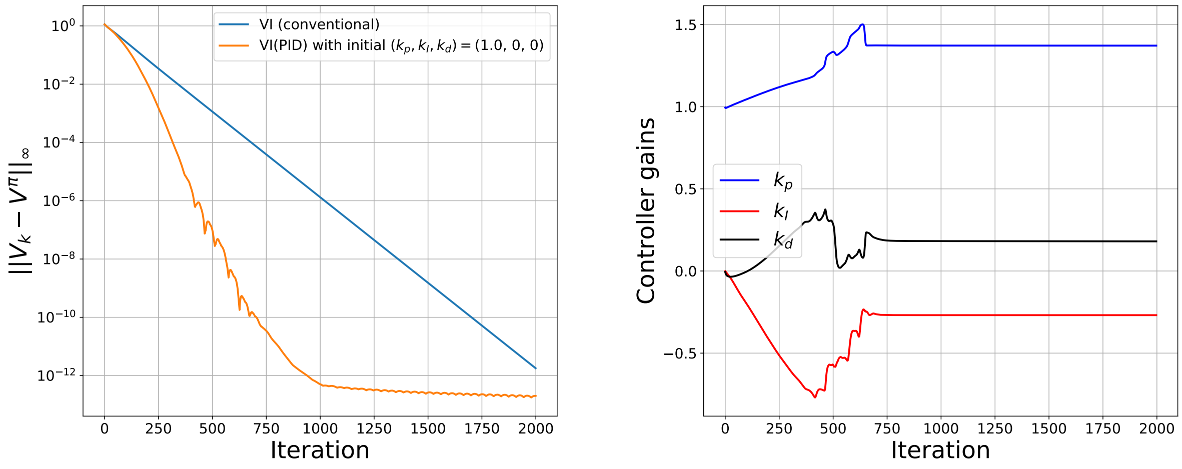
Iven M.Y. Mareels, Brian D.O. Anderson, Robert R. Bitmead, Marc Bodson, and Shankar S. Sastry. Revisiting the MIT rule for adaptive contro rule for adaptive control. In *Adaptive Systems in Control and Signal Processing 1986*, IFAC Workshop Series, pages 161–166. Pergamon, Oxford, 1987. 35

H Brendan McMahan and Geoffrey J Gordon. Fast exact planning in Markov decision processes. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 2005. 13, 33

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. 2

Andrew W Moore and Christopher G Atkeson. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine learning*, 13(1):103–130, 1993. 13, 33

Michael Muehlebach and Michael Jordan. A dynamical systems perspective on Nesterov acceleration. In *International Conference on Machine Learning (ICML)*, pages 4656–4662, 2019. 35



(a) Error behaviour

(b) Gains

Figure 20: (Chain Walk - Policy Evaluation) Gain adaptation for a 50-state chain walk problem with $\gamma = 0.99$. Here we use $(\eta, \varepsilon) = (0.05, 10^{-20})$.

Rémi Munos. Performance bounds in L_p norm for approximate value iteration. *SIAM Journal on Control and Optimization*, pages 541–561, 2007. 8, 22, 24

Rémi Munos and Csaba Szepesvári. Finite-time bounds for fitted value iteration. *Journal of Machine Learning Research (JMLR)*, 9:815–857, 2008. 2

Katsuhiko Ogata. *Modern Control Engineering*. Prentice hall Upper Saddle River, NJ, fifth edition, 2010. 4

P. V. Osburn, H. S. Whitaker, and Allen Kezer. New developments in the design of model reference adaptive control systems. In *Annual Meeting of Institute of Aeronautical Sciences (Paper No. 61-39)*, February 1961. 35

Michael L. Overton and Robert S. Womersley. On minimizing the spectral radius of a nonsymmetric matrix function: Optimality conditions and duality theory. *SIAM Journal on Matrix Analysis and Applications*, 9 (4):473–498, 1988. 19

Yangchen Pan, Adam White, and Martha White. Accelerated gradient temporal difference learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017. 34

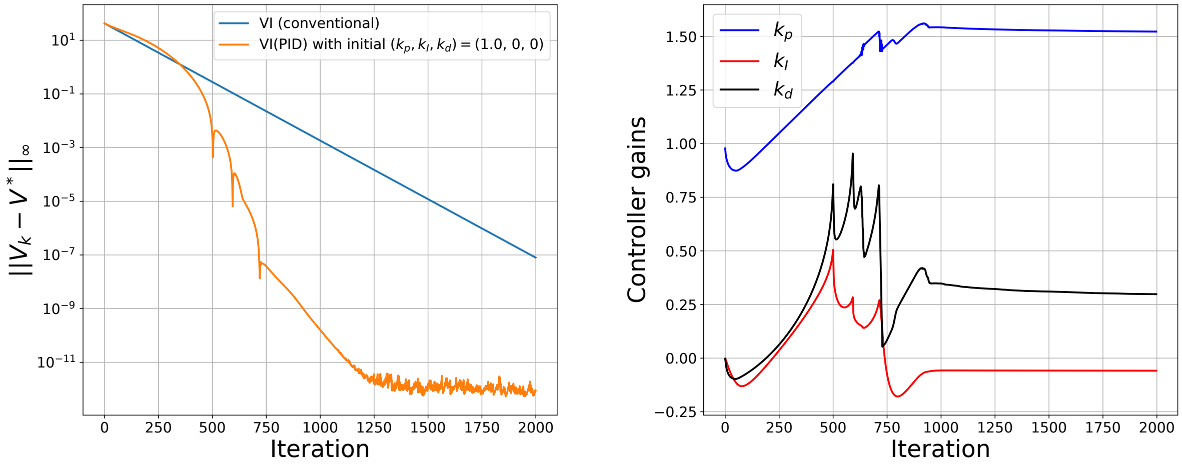
Jing Peng and Ronald J. Williams. Efficient learning and planning within the Dyna framework. *Adaptive Behavior*, 1(4):437–454, 1993. 13, 33

Boris T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964. 30

Boris T. Polyak. *Introduction to optimization*. Optimization Software, Inc., 1987. 12, 30

Joshua Romoff, Peter Henderson, David Kanaa, Emmanuel Bengio, Ahmed Touati, Pierre-Luc Bacon, and Joelle Pineau. TDprop: Does Jacobi preconditioning help temporal difference learning? In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 1082–1090, 2021. 34

Gian-Carlo Rota and W. Gilbert Strang. A note on the joint spectral radius. In *Proceedings of the Netherlands Academy*, volume 22, pages 379–381, 1960. 18



(a) Error behaviour

(b) Gains

Figure 21: (Chain Walk - Control) Gain adaptation for a 50-state chain walk problem with $\gamma = 0.99$. Here we use $(\eta, \varepsilon) = (0.05, 10^{-20})$.

Schraudolph. Local gain adaptation in stochastic gradient descent. In *International Conference on Artificial Neural Networks (ICANN)*, 1999. 7, 35

Wenjie Shi, Shiji Song, Hui Wu, Ya-Chu Hsu, Cheng Wu, and Gao Huang. Regularized Anderson acceleration for off-policy deep reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 10231–10241. 2019. 11, 12, 29, 33

Sigurd Skogestad and Ian Postlethwaite. *Multivariable Feedback Control: Analysis and Design*. Wiley New York, 2nd edition, 2005. 4

Richard S. Sutton. Adapting bias by gradient descent: An incremental version of delta-bar-delta. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, 1992. 7, 35

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2019. 2, 3, 13

Csaba Szepesvári. *Algorithms for Reinforcement Learning*. Morgan Claypool Publishers, 2010. 3, 13

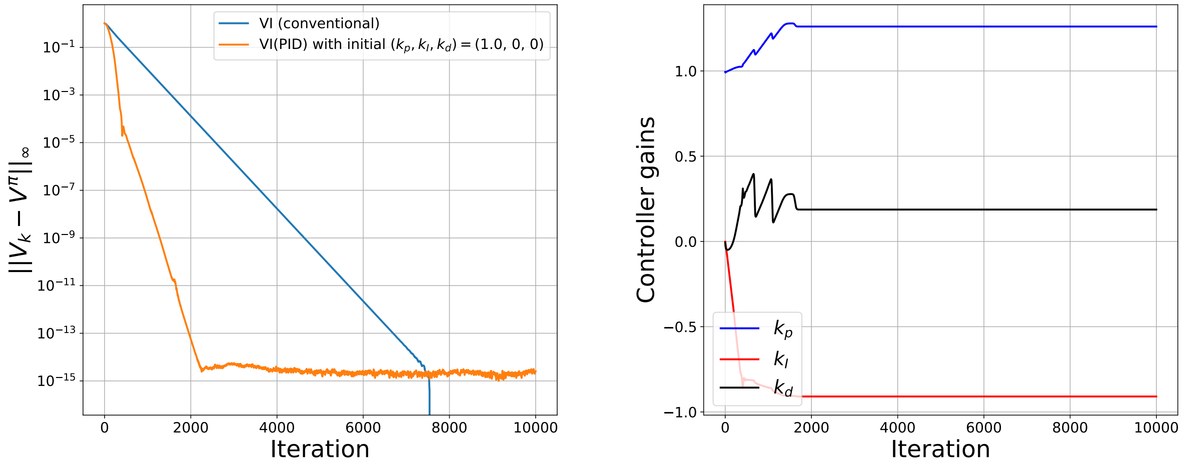
Samuele Tosatto, Matteo Pirotta, Carlo D'Eramo, and Marcello Restelli. Boosted fitted Q-iteration. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017. 2

John N. Tsitsiklis and Vincent D. Blondel. The Lyapunov exponent and joint spectral radius of pairs of matrices are hard—when not impossible—to compute and to approximate. *Mathematics of Control, Signals and Systems*, 10(1):31–40, 1997. 18

Nino Vieillard, Tadashi Kozuno, Bruno Scherrer, Olivier Pietquin, Rémi Munos, and Matthieu Geist. Leverage the average: an analysis of regularization in RL. In *Advances in Neural Information Processing Systems (NeurIPS)*, December 2020a. 33

Nino Vieillard, Bruno Scherrer, Olivier Pietquin, and Matthieu Geist. On momentum in reinforcement learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020b. 11, 12, 29, 31, 33

Ronald J. Williams and Leemon C. Baird. Tight performance bounds on greedy policies based on imperfect value functions. Technical report, Northeastern University, 1993. 22



(a) Error behaviour

(b) Gains

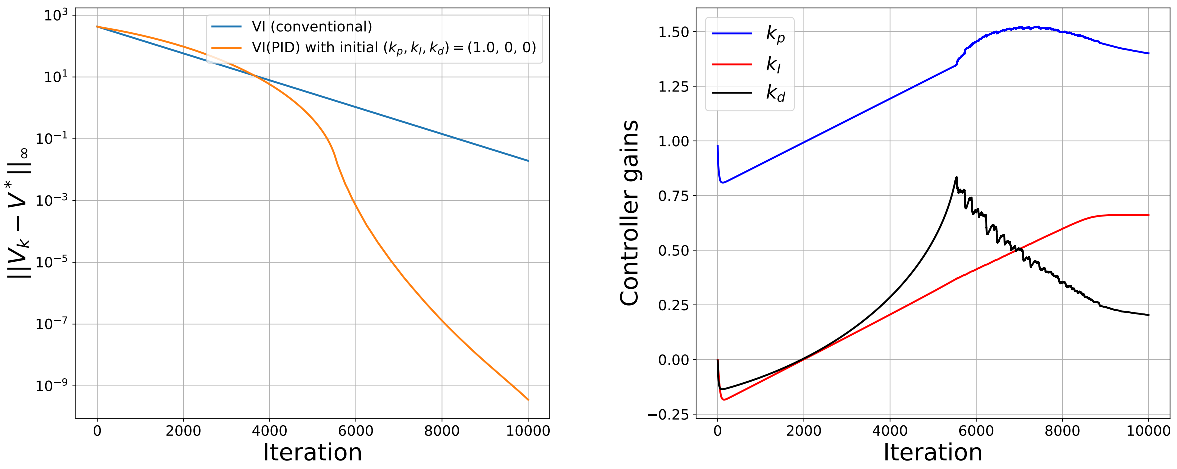
Figure 22: (Chain Walk - Policy Evaluation) Gain adaptation for a 50-state chain walk problem with $\gamma = 0.999$. Here we use $(\eta, \varepsilon) = (0.05, 10^{-20})$.

David Wingate and Kevin D. Seppi. Prioritization methods for accelerating MDP solvers. *Journal of Machine Learning Research (JMLR)*, 6(29):851–881, 2005. 13, 33

Yuhuai Wu, Mengye Ren, Renjie Liao, and Roger Grosse. Understanding short-horizon bias in stochastic meta-optimization. In *International Conference on Learning Representations (ICLR)*, 2018. 28

Hengshuai Yao and Zhi-Qiang Liu. Preconditioned temporal difference learning. In *International Conference on Machine Learning (ICML)*, 2008. 34

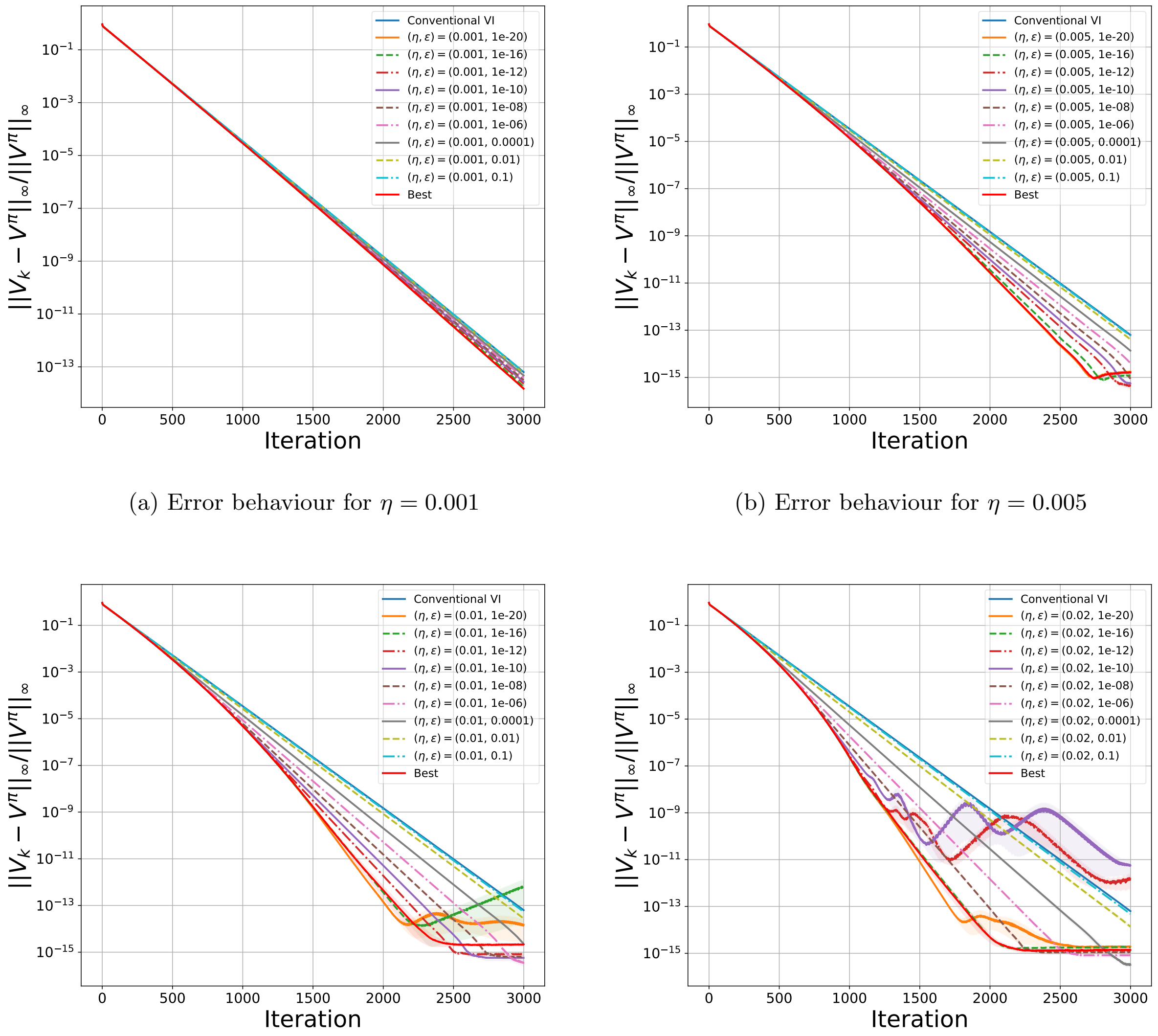
Kemin Zhou and John C. Doyle. *Essentials of robust control*. Prentice hall Upper Saddle River, NJ, 1998. 4



(a) Error behaviour

(b) Gains

Figure 23: (Chain Walk - Control) Gain adaptation for a 50-state chain walk problem with $\gamma = 0.999$. Here we use $(\eta, \varepsilon) = (0.05, 10^{-20})$.



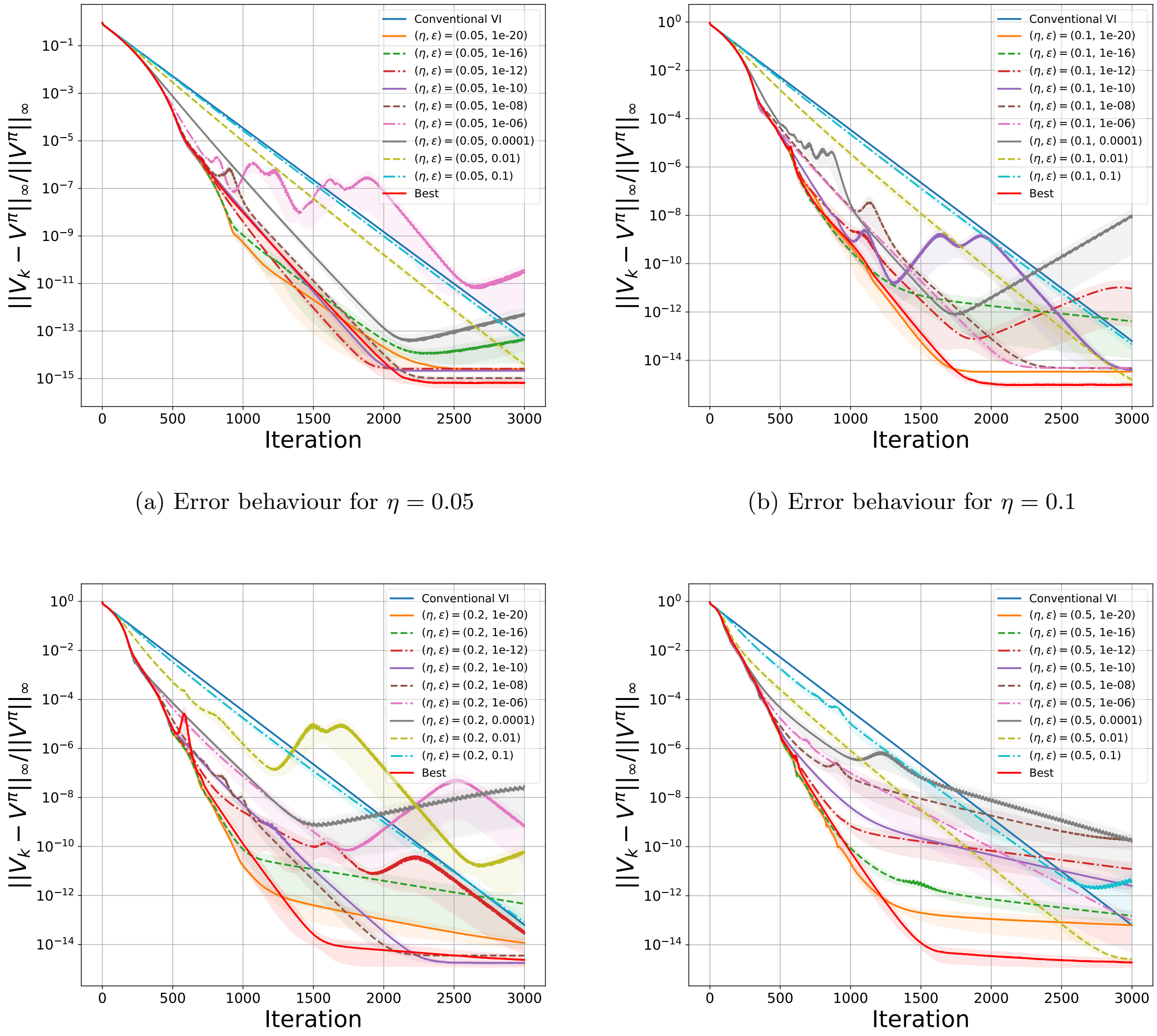
(a) Error behaviour for $\eta = 0.001$

(b) Error behaviour for $\eta = 0.005$

(c) Error behaviour for $\eta = 0.01$

(d) Error behaviour for $\eta = 0.02$

Figure 24: (Garnet - Policy Evaluation) Gain adaptation for a 50-state Garnet problem with $\gamma = 0.99$ for different meta-learning rates ($\eta \in \{0.001, 0.005, 0.01, 0.02\}$) over a range of normalizing factors ε. The mean and standard errors are evaluated based on 20 runs.



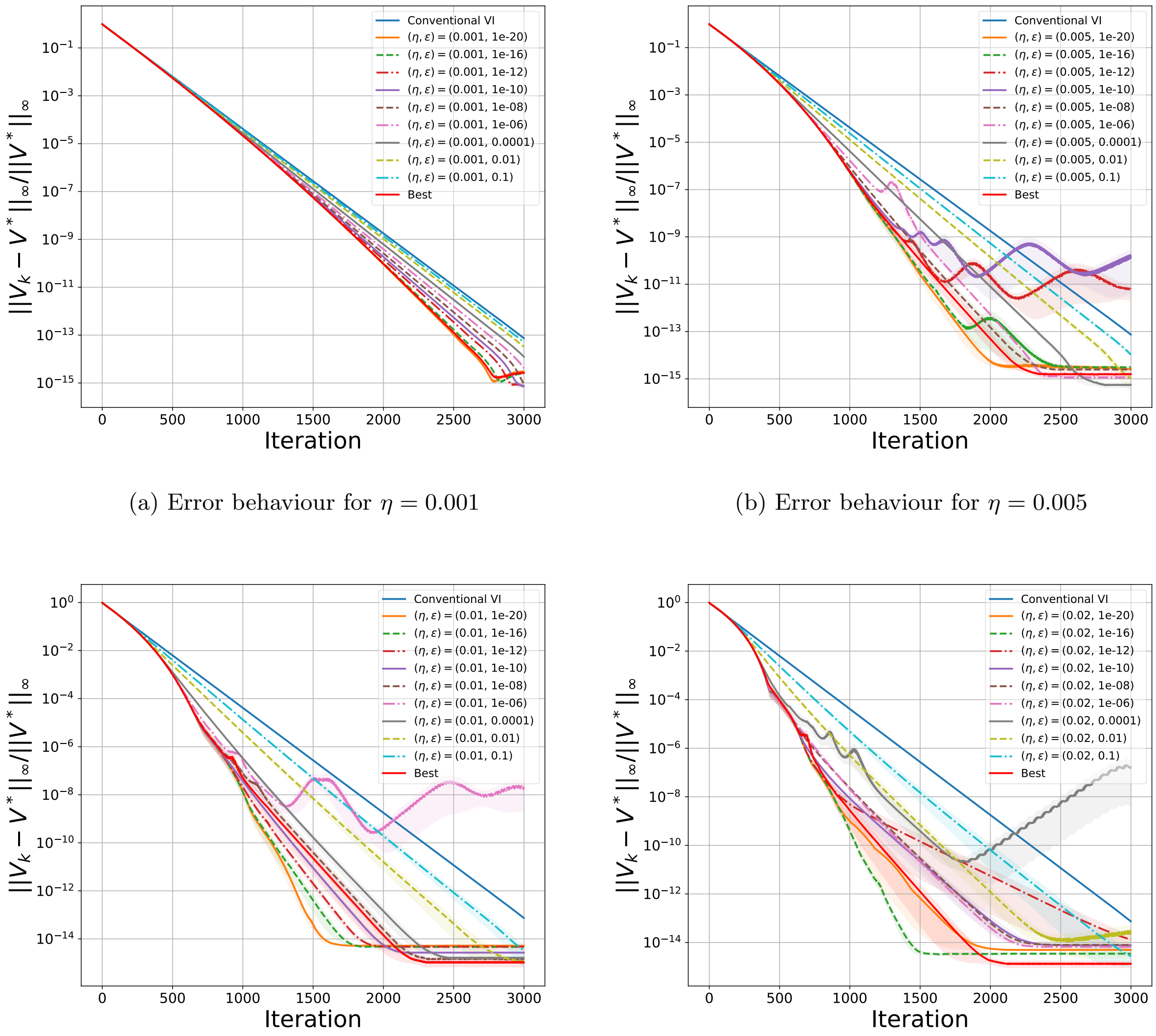
(a) Error behaviour for $\eta = 0.05$

(b) Error behaviour for $\eta = 0.1$

(c) Error behaviour for $\eta = 0.2$

(d) Error behaviour for $\eta = 0.5$

Figure 25: (Garnet - Policy Evaluation) Gain adaptation for a 50-state Garnet problem with $\gamma = 0.99$ for different meta-learning rates ($\eta \in \{0.05, 0.1, 0.2, 0.5\}$) over a range of normalizing factors ε. The mean and standard errors are evaluated based on 20 runs.



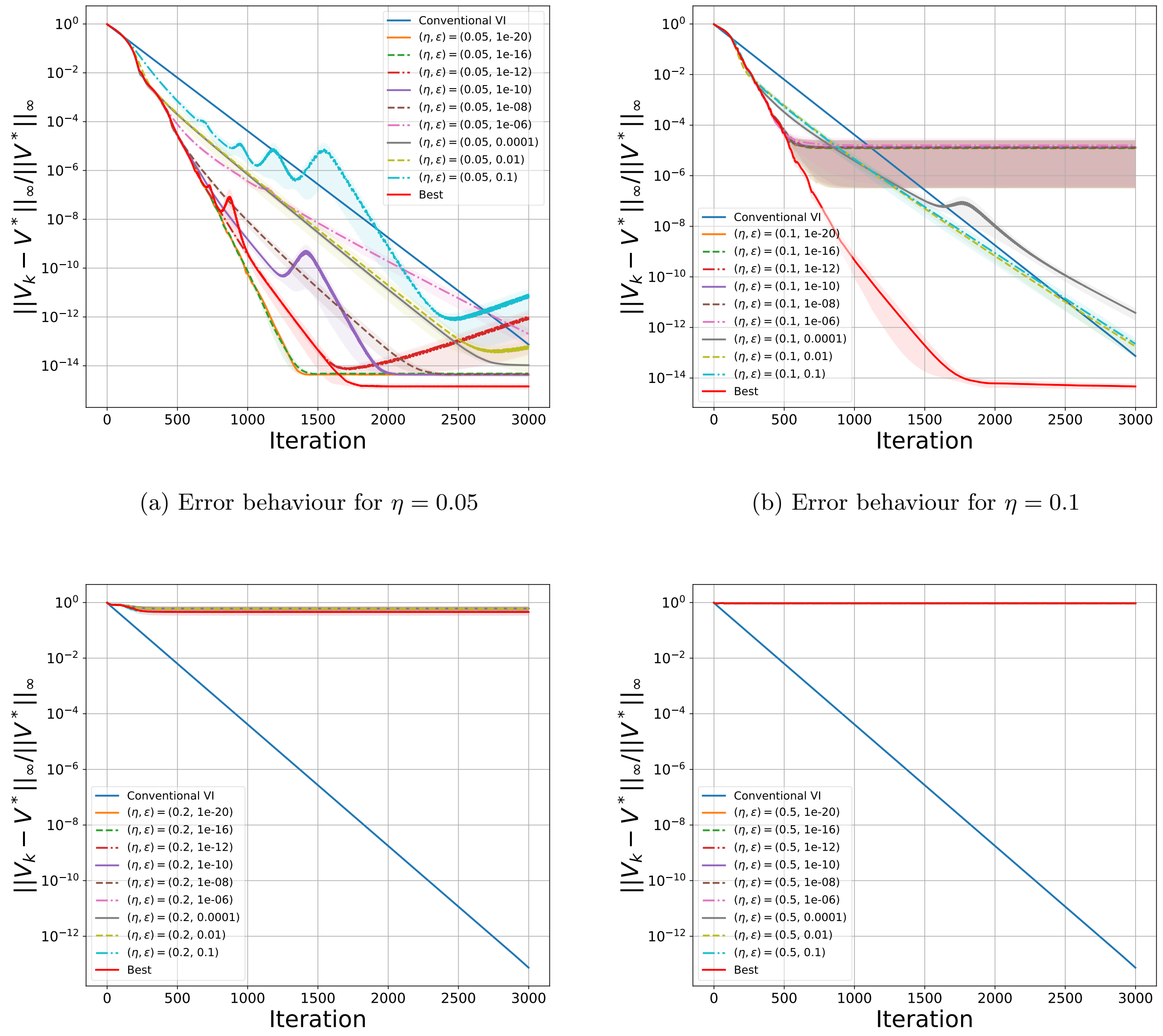
(a) Error behaviour for $\eta = 0.001$

(b) Error behaviour for $\eta = 0.005$

(c) Error behaviour for $\eta = 0.01$

(d) Error behaviour for $\eta = 0.02$

Figure 26: (Garnet - Control) Gain adaptation for a 50-state Garnet problem with $\gamma = 0.99$ for different meta-learning rates ($\eta \in \{0.001, 0.005, 0.01, 0.02\}$) over a range of normalizing factors ε. The mean and standard errors are evaluated based on 20 runs.



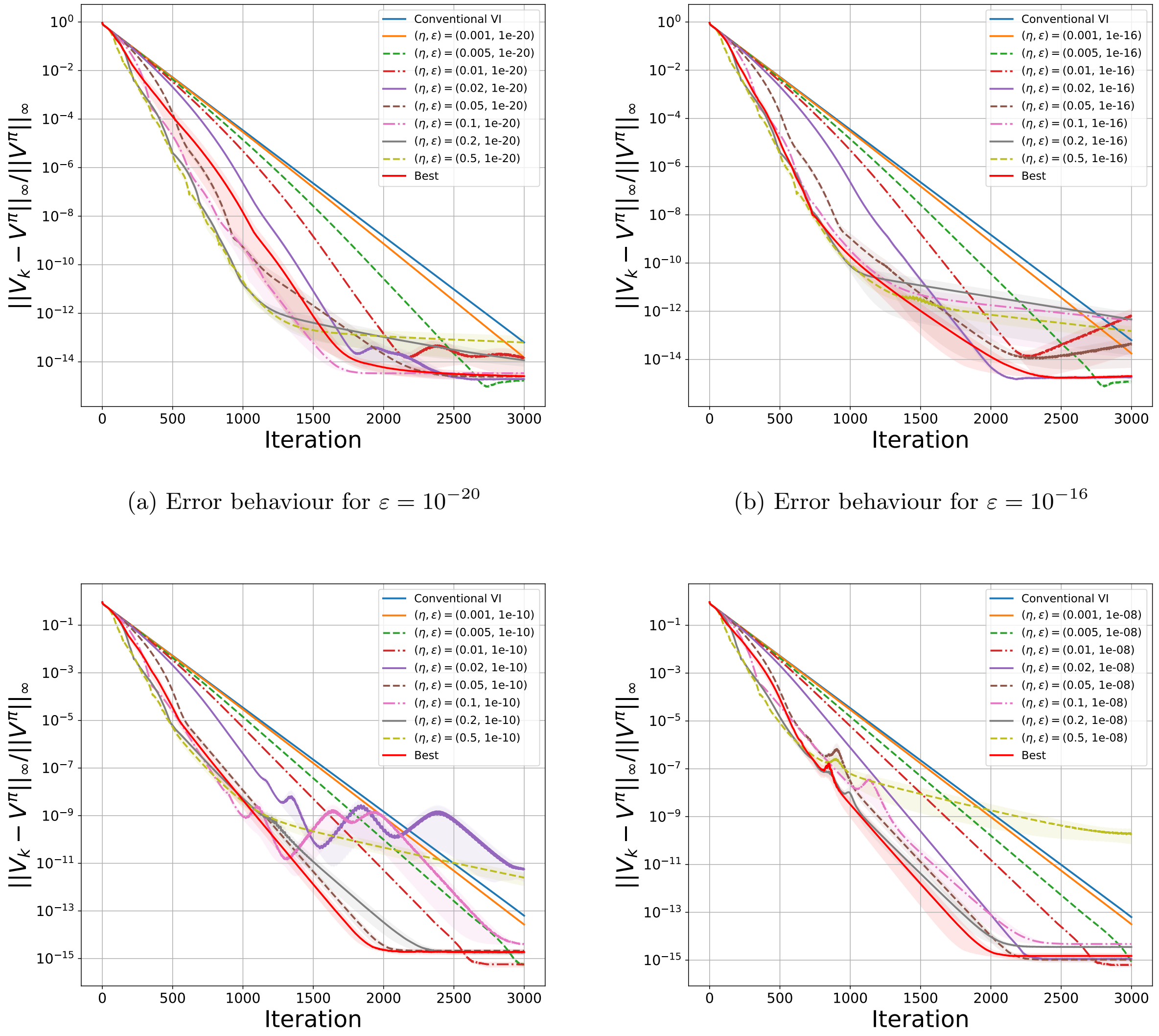
(a) Error behaviour for $\eta = 0.05$

(b) Error behaviour for $\eta = 0.1$

(c) Error behaviour for $\eta = 0.2$

(d) Error behaviour for $\eta = 0.5$

Figure 27: (Garnet - Control) Gain adaptation for a 50-state Garnet problem with $\gamma = 0.99$ for different meta-learning rates ($\eta \in \{0.05, 0.1, 0.2, 0.5\}$) over a range of normalizing factors ε. The mean and standard errors are evaluated based on 20 runs.



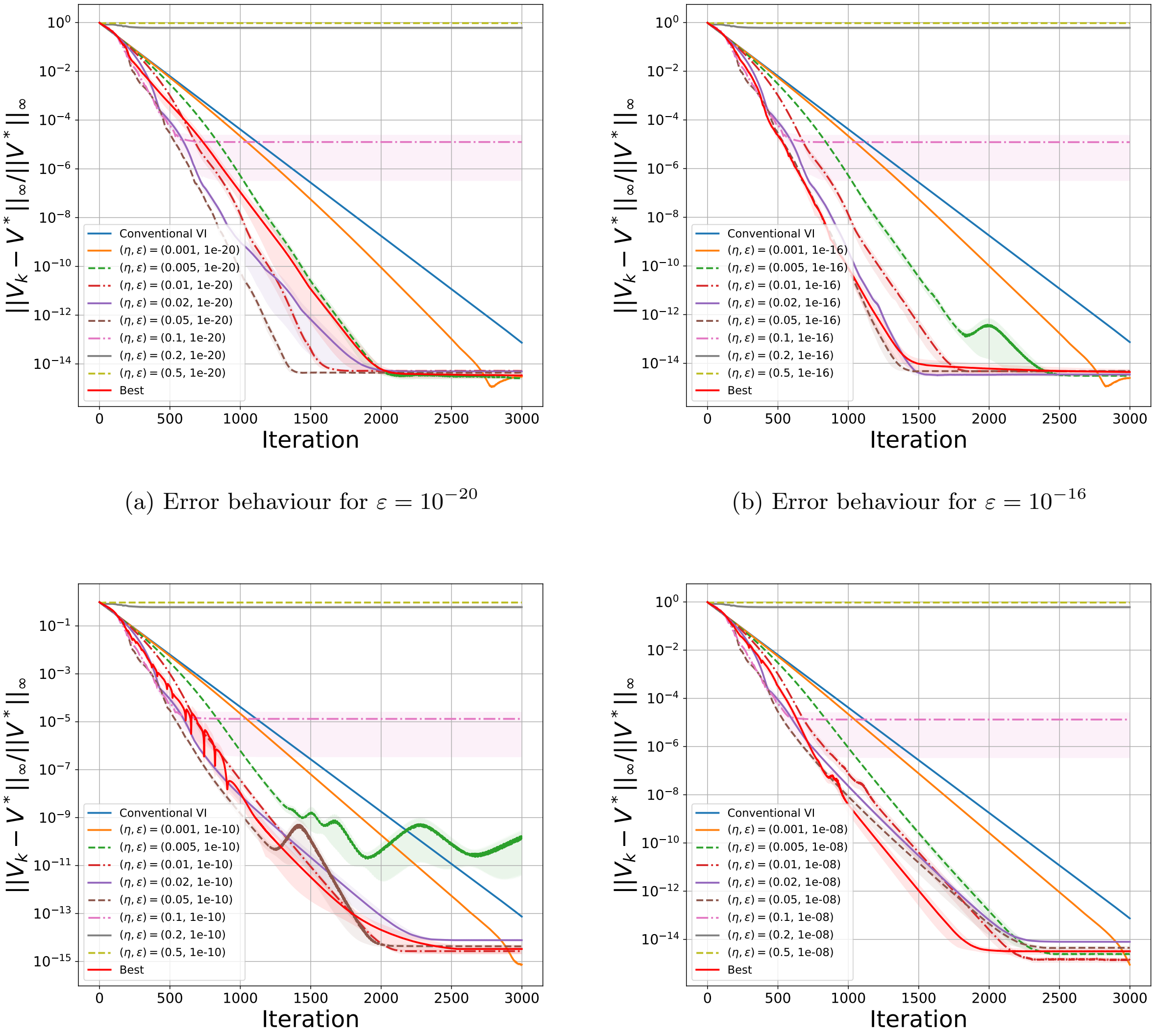
(a) Error behaviour for $\varepsilon = 10^{-20}$

(b) Error behaviour for $\varepsilon = 10^{-16}$

(c) Error behaviour for $\varepsilon = 10^{-10}$

(d) Error behaviour for $\varepsilon = 10^{-8}$

Figure 28: (Garnet - Policy Evaluation) Gain adaptation for a 50-state Garnet problem with $\gamma = 0.99$ for different normalizing factors ($\varepsilon \in \{10^{-20}, 10^{-16}, 10^{-10}, 10^{-8}\}$) over a range of meta-learning rate η. The mean and standard errors are evaluated based on 20 runs.



(a) Error behaviour for $\varepsilon = 10^{-20}$

(b) Error behaviour for $\varepsilon = 10^{-16}$

(c) Error behaviour for $\varepsilon = 10^{-10}$

(d) Error behaviour for $\varepsilon = 10^{-8}$

Figure 29: (Garnet - Control) Gain adaptation for a 50-state Garnet problem with $\gamma = 0.99$ for different normalizing factors ($\varepsilon \in \{10^{-20}, 10^{-16}, 10^{-10}, 10^{-8}\}$) over a range of meta-learning rate η. The mean and standard errors are evaluated based on 20 runs.