Control and Reinforcement Learning through the Lens of Optimization: An Algorithmic Perspective

Tolga Ok[1], Arman Sharifi Kolarijani[1], Mohamad Amin Sharifi Kolarijani[1], and Peyman Mohajerin Esfahani[1,2]

[1]Delft Center for Systems and Control, Delft University of Technology, The Netherlands
[2]Department of Mechanical and Industrial Engineering, University of Toronto, Canada

ABSTRACT. The connection between control algorithms for Markov decision processes and optimization algorithms has been implicitly and explicitly exploited since the introduction of dynamic programming algorithm by Bellman in the 1950s. Recently, this connection has attracted a lot of attention for developing new control algorithms inspired by well-established optimization algorithms. In this paper, we make this analogy explicit across four problem classes with a unified solution characterization. This novel framework, in turn, allows for a systematic transformation of algorithms from one domain to the other. In particular, we identify equivalent optimization and control algorithms that have already been pointed out in the existing literature, but mostly in a scattered way. We also discuss the issues arising in providing theoretical convergence guarantees for these new control algorithms and provide simple yet effective techniques to solve them. The provided framework and techniques then lay out a concrete methodology for developing new convergent control algorithms.

KEYWORDS: Markov decision processes, stochastic optimal control, reinforcement learning, optimization, accelerated algorithms, quasi-Newton methods

1. Introduction

Markov decision processes (MDPs) have become a standard mathematical framework for the formulation of stochastic optimal control problems, that is, the problem of dynamic decision-making under uncertainty. This popularity can be to a large extent attributed to the Bellman principle of optimality and the dynamic programming algorithm [9]. In particular, the fixed-point characterization of the optimal value function of an MDP has led to the development of a large class of iterative value-based algorithms, such as value iteration and policy iteration. This fixed-point characterization has also been used for identifying the fundamental connection between algorithms for optimal control of MDPs and those for optimization. A classic example is the policy iteration algorithm for MDPs which is an instance of the Newton method [88, 13]. Recently, the connection between optimization and control problems has attracted a lot of attention for developing new control algorithms, with faster convergence and/or lower complexity, inspired by their counterparts for solving

optimization problems. For instance, the modifications to the value iteration algorithm in [42] and the Q-learning algorithm in [106] are inspired by momentum-based acceleration in optimization.

The implicit connection between optimization algorithms and control algorithms for MDPs with a finite state-action space has also been studied more systematically. In [101], the authors look at the connection between *constrained* convex optimization algorithms and control algorithms such as Frank-Wolfe algorithm [37] and conservative policy iteration [50]. A detailed comparison between *deterministic* optimization algorithms and *model-based*[1] control algorithm is also provided in [43], where the author looks at a wide range of optimization algorithms including gradient descent, accelerated gradient descent, Newton method, and quasi-Newton method and their counterparts for solving control problems.

Motivated by these observations, this study provides an explicit framework unifying the tight link between optimization (stochastic and deterministic, respectively) and optimal control (model-based dynamic programming and model-free reinforcement learning, respectively). Specifically, this goal is achieved by exploiting the (expected) root-finding characterization of optimization problems and the (expected) fixed-point characterization of control problems. The framework yields an explicit transformation of deterministic (stochastic) convex optimization problems to model-based (model-free) control problems, and vice versa (Section 2, Table 1). This explicit relationship, in turn, allows for a systematic transformation of algorithms from one domain to the other, which we will use to identify existing (and mostly known) equivalent algorithms for optimization and control (Section 3, Table 2). Unfortunately, the common formulation for control and optimization algorithms does not allow us to borrow existing theoretical results from one field to another. Indeed, the similarity in the update rule does not carry over to the analysis, particularly when it comes to the convergence of these algorithms. Therefore, we also examine general tools for the convergence analysis of control algorithms, as well as some simple yet effective remedies for ensuring the convergence of novel control algorithms (Section 4, Theorems 4.1, 4.2, 4.3). The provided framework and techniques then lay out a concrete methodology for developing new algorithms in one domain based on the existing algorithms in the other, along with theoretical convergence guarantees.

In this paper, we restrict attention to control algorithms for *finite* state–action MDPs. For infinite (continuous) state–action spaces, aside from special cases like linear–quadratic regulators (LQR), practical computation typically requires finite-dimensional approximations. One approach is to approximate at the model level by aggregating (discretizing) the state and action spaces, thereby reducing the problem to a finite MDP [14, 87]. Another approach is to approximate the value function directly with a finite parameterization by minimizing (a proxy for) the residual of its Bellman fixed-point equation [16, 98]. Examples include linear parameterizations [15, 100], nonlinear parameterizations using neural networks [17, 93, 96], and max-plus methods [12, 41, 57,

---

[1]In this paper, the terminologies of "model-free" and "model-based" indicate the *available information (oracle)*, i.e., whether we have access to the model or only the system trajectory (samples); see Section 2.2 for more details. We note that this is different from the common terminologies in the RL literature where these terms refer to the *solution approach*, i.e., whether we identify the model along the way (model-based RL) or directly solve the Bellman equation to find the value function (model-free RL).

Optimization		Control	
$\min_x\{f(x) := \mathbb{E}_\xi[\widehat{f}(x,\xi)]\}$		$\min_{\pi:\mathcal{S}\to\mathcal{A}} \mathbb{E}_{(s_t)_{t=0}^\infty}\left[\sum_{t=0}^\infty \gamma^t c(s_t,a_t)\,\middle|\, s_0=s,\ a_t=\pi(s_t)\right],\ \forall s \in \mathcal{S}$	
Function $\widehat{f} : \mathbb{R}^\ell \times \mathbf{\Xi} \to \mathbb{R}$,		State $s \in \mathcal{S}$, Control $a \in \mathcal{A}$, Cost $c : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$	
Random variable $\xi \sim \mathbb{P}$		Dynamics $s^+ \sim \mathbb{P}(\cdot|s,a)$	
Deterministic	**Stochastic**	**Model-based**	**Model-free**
Equivalent characterization			
$\nabla f(x^\star) = 0$	$\mathbb{E}_\xi[\nabla \widehat{f}(x^\star,\xi)] = 0$	$v^\star = T(v^\star)$	$q^\star = \mathbb{E}_{\varsigma^+}\left[\widehat{T}(q^\star,\varsigma^+)\right]$
Available oracle/information			
$\nabla f(x)$	$\nabla \widehat{f}(x,\xi)$	$T(v)$	$\widehat{T}(q,\varsigma^+)$
(Prob. distribution $\mathbb{P}$)	(Samples ξ)	(Prob. kernel $\mathbb{P}$, Cost c)	(Samples $\{s,a,c(s,a),s^+\}$)
Transformation			
x	$\longleftrightarrow$	v	
∇f	$\longleftrightarrow$	$\mathrm{Id} - T$	
$\mathrm{Id} - \nabla f$	$\longleftrightarrow$	T	
$\nabla^2 f$	$\longleftrightarrow$	$I - \gamma P$	
$I - \nabla^2 f$	$\longleftrightarrow$	γP	
	(x,ξ)	$\longleftrightarrow$	(q,ς^+)
	$\nabla \widehat{f}$	$\longleftrightarrow$	$\mathrm{Id} - \widehat{T}$
	$\mathrm{Id} - \nabla \widehat{f}$	$\longleftrightarrow$	$\widehat{T}$
	$\nabla^2 \widehat{f}$	$\longleftrightarrow$	$I - \gamma \widehat{P}$
	$I - \nabla^2 \widehat{f}$	$\longleftrightarrow$	$\gamma \widehat{P}$

TABLE 1. Equivalence transformations. See Section 2 for details.

56, 77, 72]. There is also an alternative formulation in which the target function solves an infinite-dimensional linear program [46], enabling approximation via finite, tractable convex programs [29, 47, 78]. Viewed this way, most of these approximation strategies can be framed as solving a finite-dimensional fixed-point problem or a convex optimization problem.

Notations. For a vector $v \in \mathbb{R}^n$, we use $v(i)$ and $[v](i)$ to denote its i-th element. Similarly, $M(i,j)$ and $[M](i,j)$ denote the element in row i and column j of the matrix $M \in \mathbb{R}^{m \times n}$. We use $\cdot^\top$ to denote the transpose of a vector/matrix. $\|\cdot\|_2$ and $\|\cdot\|_\infty$ denote the 2-norm and ∞-norm of a vector, respectively. $\|\cdot\|_2$ and $\|\cdot\|_F$ denote the induced 2-norm and the Frobenius norm of a matrix, respectively. The identity operator is denoted by Id. We use $\mathbf{1}$ and I to denote the all-ones vector and the identity matrix, respectively. We denote the i-th unit vector by e_i, that is, the vector with its i-th element equal to 1 and all other elements equal to 0.

2. Problems: Optimization vs. Control

In this section, we provide the generic framework that connects optimization problems to control problems. In particular, we provide the explicit transformations between different variables and operations in these problems. Table 1 provides a condensed summary of this framework.

2.1. Optimization problem

We first look at the root-finding characterization of the solution to convex optimization problems. Consider the minimization problem

$$\min_{x\in\mathbb{R}^\ell}\left\{f(x)=\mathbb{E}_\xi[\widehat{f}(x,\xi)]\right\},\tag{1}$$

where $\widehat{f}:\mathbb{R}^\ell\times\Xi\to\mathbb{R}$ is a sample-wise function and ξ is random vector with a fixed probability distribution $\mathbb{P}$ over Ξ. We assume that the resulting function $f:\mathbb{R}^\ell\to\mathbb{R}$ is twice continuously differentiable and strongly convex. Much like the control problem, this problem can be considered in two settings:

(i) **Deterministic** optimization: Assume that $\mathbb{P}$ in (1) is known and that the corresponding expectation can be computed. The unique minimizer $x^\star$ is the root of the gradient operator, i.e.,

$$\nabla f(x^\star)=0.\tag{2}$$

(ii) **Stochastic** optimization: Now assume that $\mathbb{P}$ in (1) is unknown but can be sampled from. In this case, the minimizer $x^\star$ satisfies the expected root-finding problem

$$\mathbb{E}_\xi[\nabla\widehat{f}(x^\star,\xi)]=0,\tag{3}$$

where ∇ now denotes the partial derivative w.r.t. x. We note that, above, there is an underlying assumption that the differentiation w.r.t. x and expectation w.r.t. ξ can be operated in any order.

2.2. Control problem

The standard modeling framework for stochastic optimal control problems is the *Markov decision process* (MDP). An MDP is characterized by the tuple $(\mathcal{S},\mathcal{A},\mathbb{P},c,\gamma)$, where

- $\mathcal{S}$ is the state space,
- $\mathcal{A}$ is the action space,
- $\mathbb{P}$ is the transition probability kernel such that $\mathbb{P}(s^+|s,a)$ is the probability of the transition to state s^+ given that the system is in state s and the chosen control is a for each $(s,a,s^+)\in\mathcal{S}\times\mathcal{A}\times\mathcal{S}$,
- $c\in\mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ is the stage cost function such that $c(s,a)$ is the cost of taking the control action a while the system is in state s, and,
- $\gamma\in(0,1)$ is the discount factor acting as a trade-off between short- and long-term costs.

Let us now fix a *control policy* $\pi:\mathcal{S}\to\mathcal{A}$, i.e., a mapping from states to actions. The stage cost of the policy π is denoted by $c^\pi\in\mathbb{R}^\mathcal{S}$, where $c^\pi(s):=c\big(s,\pi(s)\big)$ for $s\in\mathcal{S}$. The transition probability kernel of the resulting Markov chain under the policy π is denoted by $\mathbb{P}^\pi$, where $\mathbb{P}^\pi(s^+|s)=\mathbb{P}\big(s^+|s,\pi(s)\big)$ for $s,s^+\in\mathcal{S}$. We also define $P^\pi\in[0,1]^{\mathcal{S}\times\mathcal{S}}$, with $P^\pi(s,s^+):=\mathbb{P}^\pi(s^+|s)$ for each $(s,s^+)\in\mathcal{S}\times\mathcal{S}$, to be the corresponding transition probability *matrix*. The value $v^\pi\in\mathbb{R}^\mathcal{S}$ of

the policy π is then the expected, discounted, accumulative cost of following this policy over an infinite-horizon trajectory, that is,

$$v^\pi(s) := \mathbb{E}_{(s_t)_{t=0}^\infty}\left[\sum_{t=0}^\infty \gamma^t c(s_t, a_t) \mid s_0 = s, \ a_t = \pi(s_t)\right], \quad \forall s \in \mathcal{S}.$$

Let us also define the *action-value function* (a.k.a. the *Q-function*) $q^\pi \in \mathbb{R}^{\mathcal{S} \times \mathcal{A}}$ for the policy π by

$$q^\pi(s, a) := c(s, a) + \gamma \ \mathbb{E}_{s^+}[v^\pi(s^+) \mid (s, a)], \quad \forall (s, a) \in \mathcal{S} \times \mathcal{A},$$

so that $v^\pi(s) = q^\pi(s, \pi(s))$ for each $s \in \mathcal{S}$. Given a value function $v \in \mathbb{R}^{\mathcal{S}}$, let us also define the *greedy policy* $\pi_v : \mathcal{S} \to \mathcal{A}$ *w.r.t.* v denoted and given by

$$\pi_v(s) \in \operatorname*{argmin}_{a \in \mathcal{A}} \left\{c(s, a) + \gamma \ \mathbb{E}_{s^+}\left[v(s^+) \mid (s, a)\right]\right\}, \quad \forall s \in \mathcal{S}.$$

Similarly, for a Q-function $q \in \mathbb{R}^{\mathcal{S} \times \mathcal{A}}$, we define the *greedy policy* $\pi_q : \mathcal{S} \to \mathcal{A}$ *w.r.t.* q by

$$\pi_q(s) \in \operatorname*{argmin}_{a \in \mathcal{A}} q(s, a), \quad \forall (s, a) \in \mathcal{S} \times \mathcal{A}.$$

The problem of interest is to control the MDP optimally, that is, to find an optimal policy π^* with the optimal (action-)value functions

$$v^\star = \min_\pi v^\pi \quad \text{and} \quad q^\star = \min_\pi q^\pi, \tag{4}$$

so that the expected, discounted, infinite-horizon cost is minimized. Let us also note that the optimal policy, that is a minimizer of the preceding optimization problems, is greedy w.r.t. $v^\star$ and $q^\star$, i.e., $\pi^\star = \pi_{v^\star} = \pi_{q^\star}$.

Interestingly, the optimal (action-)value functions introduced in (4) can be equivalently characterized as the fixed-point of two different operators each of which is useful depending on the available information (oracle):

(i) **Model-based** control: When we have access to the transition kernel and the cost function, the problem is usually characterized by the fixed-point problem. Indeed, defining the *Bellman optimality operator* $T : \mathbb{R}^{\mathcal{S}} \to \mathbb{R}^{\mathcal{S}}$ by

$$[T(v)](s) := \min_{a \in \mathcal{A}} \left\{c(s, a) + \gamma \ \mathbb{E}_{s^+}\left[v(s^+) \mid (s, a)\right]\right\}, \quad \forall s \in \mathcal{S},$$

we have

$$v^\star(s) = [T(v^\star)](s), \quad \forall s \in \mathcal{S}, \tag{5}$$

or, equivalently, $v^\star = T(v^\star)$, that is, the optimal value function $v^\star$ is the *unique* fixed-point of T. The uniqueness follows from the fact that the operator T is a γ-contraction in ∞-norm. Observe that, in this case, T can be exactly computed given the model of the underlying MDP.

(ii) **Model-free** control: In real applications, the model is often unknown, and instead, one can generate samples. Examples of this are very large systems where identifying the model is prohibitively expensive but transitions between states can be observed and recorded, such as those in video games. This problem has been studied extensively in the reinforcement

learning community and is often formulated as the *expected* fixed-point problem. To provide this characterization, let us define the *vector of sampled next states* $\varsigma^+ \in \mathcal{S}^{\mathcal{S} \times \mathcal{A}}$ where

$$\varsigma^+(s, a) \coloneqq s^+ \sim \mathbb{P}(\cdot | s, a), \tag{6}$$

is a sample of the next state drawn from the distribution $\mathbb{P}(\cdot | s, a)$ for each $(s, a) \in \mathcal{S} \times \mathcal{A}$. Then, defining the *sampled* Bellman optimality operator $\widehat{T} : \mathbb{R}^{\mathcal{S} \times \mathcal{A}} \times \mathcal{S}^{\mathcal{S} \times \mathcal{A}} \to \mathbb{R}^{\mathcal{S} \times \mathcal{A}}$ by[2]

$$[\widehat{T}(q, \varsigma^+)](s, a) \coloneqq c(s, a) + \gamma \min_{a^+ \in \mathcal{A}} q(\varsigma^+(s, a), a^+), \quad \forall (s, a) \in \mathcal{S} \times \mathcal{A},$$

we have

$$q^\star(s, a) = \mathbb{E}_{\varsigma^+(s,a)} \left[[\widehat{T}(q^\star, \varsigma^+)](s, a) \right], \quad \forall (s, a) \in \mathcal{S} \times \mathcal{A}, \tag{7}$$

or, equivalently, $q^\star = \mathbb{E}_{\varsigma^+}[\widehat{T}(q^\star, \varsigma^+)]$.

2.3. **Transformation**

In what follows, we focus on *tabular MDPs with a finite state-action space*. We thus consider $\mathcal{S} = \{1, 2, \ldots, n\}$ and $\mathcal{A} = \{1, 2, \ldots, m\}$. Recall the Bellman optimality operator $T : \mathbb{R}^{\mathcal{S}} \to \mathbb{R}^{\mathcal{S}}$ with

$$T(v) = \sum_{s \in \mathcal{S}} [T(v)](s) \cdot e_s,$$

where $e_s \in \mathbb{R}^{\mathcal{S}}$ is the unit vector for the state $s \in \mathcal{S}$, and the sampled Bellman optimality operator $\widehat{T} : \mathbb{R}^{\mathcal{S} \times \mathcal{A}} \times \mathcal{S}^{\mathcal{S} \times \mathcal{A}} \to \mathbb{R}^{\mathcal{S} \times \mathcal{A}}$ with

$$\widehat{T}(q, \varsigma^+) = \sum_{(s,a) \in \mathcal{S} \times \mathcal{A}} [\widehat{T}(q, \varsigma^+)](s, a) \cdot e_{(s,a)},$$

where $e_{(s,a)} \in \mathbb{R}^{\mathcal{S} \times \mathcal{A}}$ is the unit vector for the state-action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$, and $\varsigma^+ \in \mathcal{S}^{\mathcal{S} \times \mathcal{A}}$ is the vector of sampled next states defined in (6).

Before providing the equivalence relations between optimization and control problems, let us provide an important result for the Bellman optimality operator.

Lemma 2.1 (Jacobian of T)**.** *Suppose that T is differentiable at v. Then, $\partial T(v) = \gamma P^{\pi_v}$, where π_v is a greedy policy w.r.t. v.*

Proof. Let us define the matrix $P^a \in \mathbb{R}^{n \times n}$ with entries $P^a(s, s^+) = \mathbb{P}(s^+ | s, a)$, for every control action $a \in \mathcal{A}$. Fix $s \in \mathcal{S}$, and observe that

$$\partial_v \big([T(v)](s) \big) = \partial_v \left(\min_{a \in \mathcal{A}} \{ c(s, a) + \gamma \, \mathbb{E}_{s^+}[v(s^+) \mid (s, a)] \} \right) = \partial_v \left(\min_{a \in \mathcal{A}} \{ c(s, a) + \gamma \cdot P^a(s, \cdot) \cdot v \} \right),$$

where $P^a(s, \cdot)$ is the s-th row of P^a. Then, using the envelope theorem [92], we have

$$\partial_v \big([T(v)](s) \big) = \partial_v \left(c\big(s, \pi_v(s)\big) + \gamma \cdot P^{\pi_v}(s, \cdot) \cdot v \right) = \gamma \cdot P^{\pi_v}(s, \cdot).$$

Hence, $\partial T(v) = \gamma P^{\pi_v}$, and the claim follows. $\qquad\square$

---

[2] Strictly speaking, the provided sampled Bellman optimality operator is the empirical version of the Bellman optimality operator for the Q-function, given by $[\bar{T}(q)](s, a) \coloneqq c(s, a) + \gamma \mathbb{E}_{s^+} \left[\min_{a^+ \in \mathcal{A}} q(s^+, a^+) \mid (s, a) \right]$ for each $(s, a) \in \mathcal{S} \times \mathcal{A}$.

Using the preceding result, we have

$$\partial(\mathrm{Id}\,-T)(v) = I - \gamma P(v),$$

where $P(v) \coloneqq P^{\pi_v}$ is the state transition probability matrix of the greedy policy π_{v_k} w.r.t. v_k. Then, comparing the characterizations (5) and (2), we can draw the following equivalence relations between deterministic optimization and model-based control:

$$x \leftrightarrow v, \quad \begin{cases} \nabla f & \leftrightarrow \quad \mathrm{Id}\,-T \\ \mathrm{Id}\,-\nabla f & \leftrightarrow \quad T \end{cases}, \quad \begin{cases} \nabla^2 f & \leftrightarrow \quad I - \partial T = I - \gamma P \\ I - \nabla^2 f & \leftrightarrow \quad \partial T = \gamma P \end{cases},$$

where Id is the identity operator, I is the identity matrix, and $P = P(v)$ is the transition probability matrix of the Markov chain under the greedy policy w.r.t. v. Similarly, for stochastic optimization and model-free control, the characterizations (7) and (3) point to the following equivalence relations:

$$(x, \xi) \leftrightarrow (q, \varsigma^+), \quad \begin{cases} \nabla \widehat{f} & \leftrightarrow \quad \mathrm{Id}\,-\widehat{T} \\ \mathrm{Id}\,-\nabla \widehat{f} & \leftrightarrow \quad \widehat{T} \end{cases}, \quad \begin{cases} \nabla^2 \widehat{f} & \leftrightarrow \quad I - \partial \widehat{T} = I - \gamma \widehat{P} \\ I - \nabla^2 \widehat{f} & \leftrightarrow \quad \partial \widehat{T} = \gamma \widehat{P} \end{cases},$$

where $\widehat{P} = \widehat{P}(q, \varsigma^+) \in [0,1]^{(\mathcal{S} \times \mathcal{A}) \times (\mathcal{S} \times \mathcal{A})}$ is the *synchronously sampled* transition probability matrix of the Markov chain under π_q with elements[3]

$$[\widehat{P}(q, \varsigma^+)]\big((s,a),(s',a')\big) = \begin{cases} 1, & s' = \varsigma^+(s,a), \ a' = \pi_q\big(\varsigma^+(s,a)\big), \\ 0, & \text{otherwise}, \end{cases}$$

for each $\big((s,a),(s',a')\big) \in (\mathcal{S} \times \mathcal{A}) \times (\mathcal{S} \times \mathcal{A})$.

3. Algorithms: Optimization vs. Control

We now look at existing algorithms for optimization and control and their equivalence within the proposed framework. We start with the well-known baseline first-order and second-order algorithms and show how the application of the proposed transformations to well-established optimization algorithms, such as gradient descent and Newton method, leads to well-known control algorithms such as value iteration and policy iteration. We then cover a wide range of modifications to optimization algorithms, from momentum to spectral decomposition of the Hessian, and their counterpart control algorithms. We note that most of these equivalences have already been pointed out in the existing literature, however, mostly in a scattered way. An exception is [43], which studies the relation between *deterministic* optimization algorithms and *model-based* control algorithms.

To ease the exposition, we unify iterative algorithms by

$$y_{k+1} = y_k + d_k, \quad k = 0, 1, \ldots,$$

where $y_k = x_k$, v_k or q_k based on the context. In all considered settings, d_k represents the update vector between iterations k and $k+1$. This form allows us to characterize algorithms in terms of d_k. Moreover, we use Greek letters $\alpha_k, \beta_k, \ldots$ as scalar coefficients (e.g., step-size, learning rate,

---

[3]Once again, strictly speaking, $\widehat{P}(q, \varsigma^+)$ is the empirical version of the *state-action* transition probability matrix $\bar{P}(q) \in [0,1]^{(\mathcal{S} \times \mathcal{A}) \times (\mathcal{S} \times \mathcal{A})}$ of the Markov chain under the greedy policy π_q w.r.t. q, with elements $[\bar{P}(q)]\big((s,a),(s',a')\big) = \mathbb{P}(s'|s,a)$ if $a' = \pi_q(s')$ and $= 0$ otherwise, for $(s,a),(s',a') \in \mathcal{S} \times \mathcal{A}$.

Deterministic optimization	Model-based control	Stochastic optimization	Model-free control
$(y = x)$	$(y = v)$	$(y = x)$	$(y = q)$
$g(x) := \nabla f(x)$	$g(v) := v - T(v)$	$\widehat{g}_k(x) := \nabla \widehat{f}(x, \xi_k)$	$\widehat{g}_k(q) := q - \widehat{T}(q, \varsigma_k^+)$
$H(x) := \nabla^2 f(x)$	$H(v) := I - \gamma P(v)$	$\widehat{H}_k(x) := \nabla^2 \widehat{f}(x, \xi_k)$	$\widehat{H}_k(q) := I - \gamma \widehat{P}(q, \varsigma_k^+)$

First Order Methods (access to gradient oracle)

Vanilla First Order Method

Deterministic optimization	Model-based control	Stochastic optimization	Model-free control
GD [66]	Rel-VI [9, 59]	SGD [89]	QL [104]
$d_k = -\alpha_k g(y_k)$		$d_k = -\alpha_k \widehat{g}_k(y_k)$	

Acceleration via Momentum

Deterministic optimization	Model-based control	Stochastic optimization	Model-free control
Mom-GD [84]	Mom-VI [42]	Mom-SGD [107]	SQL [40], NeSA [30], Mom-QL [106]
$d_k = -\alpha_k g(y_k) + \beta_k d_{k-1}$		$\begin{cases} d'_{k-1} = \widehat{g}_k(y_k) - \widehat{g}_k(y_{k-1}) \\ d_k = -\alpha_k \widehat{g}_k(y_k) - \beta_k d'_{k-1} + \delta_k d_{k-1} \end{cases}$	
Acc-GD [79]	Acc-VI [42]		
$d_k = -\alpha_k g(y_k + \beta_k d_{k-1}) + \beta_k d_{k-1}$			

Acceleration via Anchoring

Deterministic optimization	Model-based control	Stochastic optimization	Model-free control
Anc-GD [108]	Anc-VI [65]	SHI [24]	HQL [21], SAVID [63]
$d_k = \beta_k(y_0 - y_k) - \alpha_k g(y_k)$		$d_k = \beta_k(y_0 - y_k) + \alpha_k \widehat{g}_k(y_k)$	

Acceleration via PID Control

Deterministic optimization	Model-based control	Stochastic optimization	Model-free control
PID-Opt (Full-batch) [4]	PID-VI [35]	PID-Opt (Mini-batch) [4]	PID-QL [8]
$\begin{cases} d_k^I = -\alpha_k g(y_k) + \beta_k d_{k-1}^I \\ d'_{k-1} = g(y_k) - g(y_{k-1}) \\ d_k^D = \delta_k d_{k-1} + (1 - \delta_k) d'_{k-1} \\ d_k = -\kappa_k^P g(y_k) + \kappa_k^I d_k^I + \kappa_k^D d_k^D \end{cases}$		$\begin{cases} d_k^I = -\alpha_k \widehat{g}_k(y_k) + \beta_k d_{k-1}^I \\ d'_{k-1} = \widehat{g}_k(y_k) - \widehat{g}_{k-1}(y_{k-1}) \\ y'_{k-1} = (1 - \eta_k) y'_{k-2} + \eta_k y_{k-1} \\ d_k^D = \delta_k(y_k - y'_{k-1}) + (1 - \delta_k) d'_{k-1} \\ d_k = -\kappa_k^P \widehat{g}_k(y_k) + \kappa_k^I d_{k+1}^I + \kappa_k^d d_k^D \end{cases}$	

QNM via Anderson Mixing[†]

Deterministic optimization	Model-based control	Stochastic optimization	Model-free control
AA-GD [76]	AA-VI [39]	SAM [105]	SAA-QL [95]
$\begin{cases} Y_k = [y_k, \ldots, y_{k-m_k}] \\ G_k = [g(y_k), \ldots, g(y_{k-m_k})] \\ w_k = (\mathbf{1}^\top (G_k^\top G_k)^{-1} \mathbf{1})^{-1}(G_k^\top G_k)^{-1} \mathbf{1} \\ d_k = -y_k + (Y_k - \alpha_k G_k) w_k \end{cases}$		$\begin{cases} D_k^y = [d_{k-m_k}^y, \ldots, d_{k-1}^y] \quad \text{with} \quad d_i^y = y_{i+1} - y_i \\ D_k^g = [d_{k-m_k}^g, \ldots, d_{k-1}^g] \quad \text{with} \quad d_i^g = \widehat{g}_{i+1}(y_{i+1}) - \widehat{g}_i(y_i) \\ D_k = \beta_k I - \alpha_k(D_k^y + \beta_k D_k^g)\big((D_k^g)^\top D_k^g + R_k\big)^{-1}(D_k^g)^\top \\ d_k = -D_k \widehat{g}_k(y_k) \end{cases}$	

Second-Order Methods (access to gradient and Hessian oracles)

QNM via Spectral Preconditioning[‡]

Deterministic optimization	Model-based control	Stochastic optimization	Model-free control
(SP-GD)[32]	R1-VI [55]	(NewSamp) [33]	R1-QL [55]
$\begin{cases} D_k = \alpha_k I + \beta_k U_k W_k^\top, \\ d_k = -D_k^{-1} g(y_k), \end{cases}$		$\begin{cases} D_k = \alpha_k I + \beta_k \widehat{U}_k \widehat{W}_k^\top, \\ d_k = -D_k^{-1} \widehat{g}_k(y_k) \end{cases}$	

Vanilla Second-Order Method

Deterministic optimization	Model-based control	Stochastic optimization	Model-free control
NM	PI [48]	SNR [90, 31]	ZQL [31]
$d_k = -\alpha_k [H(y_k)]^{-1} g(y_k)$		$\begin{cases} D_k = (1 - \beta_k) D_{k-1} + \beta_k \widehat{H}_k(y_k) \\ d_k = -\alpha_k D_k^{-1} \widehat{g}_k(y_k) \end{cases}$	

TABLE 2. Equivalent algorithms: The vector d_k is the update vector in the iterative scheme $y_{k+1} = y_k + d_k$ for $k = 0, 1, \ldots$. The Greek letters $\alpha_k, \beta_k, \ldots$ are scalar coefficients that may take different values (even in equivalent algorithms). [†]R_k is a regularizer depending on D_k^y and D_k^g. [‡]U_k and W_k are tall matrices depending on the (sampled) second-order oracle. See Section 3 for details.

etc.) in the update vectors. We note that the specific values of these parameters may differ between equivalent algorithms. A compact summary of this can be found in Table 2.

Some remarks are in order regarding the following equivalence relationships. First, the following list of algorithms is not meant to be an exhaustive review of the literature. Instead, our aim is to provide illustrative examples of the application of transformations of Table 1 in identifying equivalent algorithms for optimization and control. Second, the provided update rules for the cited algorithms, although covering the main characteristic of interest, are occasionally not exact. This is particularly the case in stochastic/model-free settings where other techniques, such as batch estimation or variance reduction, are also used in the original algorithms. Finally, the equivalence between optimization and control algorithms does not imply that these algorithms enjoy similar theoretical guarantees, particularly when it comes to convergence. We will discuss this issue in more detail in Section 4.

3.1. Baseline methods

We start our discussion with the vanilla first-order and second-order algorithms.

Vanilla first-order methods. The celebrated *Gradient Descent (GD)* [66] method is characterized by

$$d_k = -\alpha_k \nabla f(x_k),$$

where α_k is a properly chosen step-size. Applying the transformations of Table 1 to GD, we derive the so-called *Relaxed Value Iteration (Rel-VI)* [59, 85, 42] with

$$d_k = -\alpha_k \big(v_k - T(v_k) \big),$$

for model-based control. In particular, for the constant step-size $\alpha_k = 1$, we have the standard VI algorithm $v_{k+1} = T(v_k)$ [9]. Correspondingly, *Stochastic GD (SGD)* [89] is characterized by

$$d_k = -\alpha_k \nabla \widehat{f}(x_k, \xi_k).$$

Under the transformations of Table 1, SGD leads to the *synchronous Q-Learning (QL)* algorithm [104, 52] with[4]

$$d_k = -\alpha_k \big(q_k - \widehat{T}(q_k, \varsigma_k^+) \big).$$

Vanilla second-order methods. In second-order algorithms, d_k is specified by both the gradient and the Hessian oracles. The *damped Newton Method (NM)* is one such algorithm with

$$d_k = -\alpha_k [\nabla^2 f(x_k)]^{-1} \nabla f(x_k).$$

The *pure* Newton step with $\alpha_k = 1$ has a *local* quadratic convergence if, in addition to f being strongly convex, the Hessian is Lipschitz-continuous [23, Thm. 5.3]. Globally, however, the pure Newton method can lead to divergence. This is the reason behind introducing the step-size $\alpha_k < 1$ in the damped version which can be used to guarantee a global linear convergence. We can use the

---

[4]This is the so-called *synchronous* update of the Q-function in *all* state-action pairs in each iteration, corresponding to the *parallel sampling model* introduced by [52].

transformations of Table 1 in order to transform the Newton method into a model-based control algorithm with

$$d_k = -\big(I - \gamma P(v_k)\big)^{-1}(v_k - T(v_k)),$$

where $P(v_k)$ is the transition probability matrix of the Markov chain under the greedy policy w.r.t. v_k. The derived model-based control algorithm then corresponds to the well-known *Policy Iteration (PI)* algorithm [88] with $v_{k+1} = \big(I - \gamma P(v_k)\big)^{-1} c^{\pi_{v_k}}$, where $c^{\pi_{v_k}}$ is the vector of stage costs corresponding to the greedy policy π_{v_k} w.r.t. v_k. Indeed, the PI algorithm is equivalent to the semi-smooth Newton method with a local quadratic convergence rate [38].

In the model-free case, the stochastic version of the Newton method [90] has been a source of inspiration for developing second-order-type Q-learning algorithms. In particular, the *Stochastic Newton-Raphson (SNR)* [90] algorithm with

$$\begin{cases} D_k = (1 - \beta_k)D_{k-1} + \beta_k \nabla^2 \widehat{f}(x_k, \xi_k), \\ d_k = -\alpha_k D_k^{-1} \nabla \widehat{f}(x_k, \xi_k), \end{cases}$$

was used for developing the *Zap QL (ZQL)* algorithm [31] with

$$\begin{cases} D_k = (1 - \beta_k)\, D_{k-1} + \beta_k\, e_{(s_k, a_k)}\, \big(e_{(s_k, a_k)} - \gamma\, e_{(s_k^+, \pi_k(s_k^+))}\big)^\top, \\ d_k = -\alpha_k\, D_k^{-1}\, \big(q_k(s_k, a_k) - [\widehat{T}(q_k, \varsigma_k^+)](s_k, a_k)\big)\, e_{(s_k, a_k)}, \end{cases}$$

where $\pi_k(s_k^+) \in \mathrm{argmin}_{a \in \mathcal{A}} q_k(s_k^+, a)$ is a greedy action w.r.t. q_k evaluated at the sampled next state $\varsigma_k^+(s_k, a_k) = s_k^+ \sim \mathbb{P}(\cdot | s_k, a_k)$. Note that the preceding algorithm involves updating one entry of the action-value function q_k at each iteration k, corresponding to the state-action pair (s_k, a_k) chosen at iteration k – recall that $e_{(s,a)} \in \mathbb{R}^{\mathcal{S} \times \mathcal{A}}$ is the unit vector corresponding to the state-action pair (s, a). Moreover, ZQL in [31] implements an eligibility trace with a decaying factor which we omit (i.e., set to 0) for simplicity of the representation. The implementation of zap QL algorithm with *synchronous* update of the Q-function in all state-action pairs in each iteration is then characterized by

$$\begin{cases} D_k = (1 - \beta_k)D_{k-1} + \beta_k\big(I - \gamma \widehat{P}(q_k, \varsigma_k^+)\big), \\ d_k = -\alpha_k D_k^{-1}\big(q_k - \widehat{T}(q_k, \varsigma_k^+)\big), \end{cases} \tag{8}$$

where $\widehat{P}(q, \varsigma^+)$ is the synchronously sampled state-action transition probability matrix of the Markov chain under the greedy policy w.r.t. q. Note that (8) is exactly the SNR algorithm under the transformations of Table 1.

3.2. Accelerated methods

We now focus on classic acceleration techniques such as momentum and anchoring in optimization. These techniques have recently attracted a lot of attention in developing control algorithms with improved convergence and complexity properties. We also look at a classic control engineering approach, namely, proportional-integral-derivative (PID) control, and its application in optimization and control algorithms.

Momentum methods. In the so-called momentum-based algorithms, the update vector d_k is specified by gradient oracles but also depends on d_{k-1}. One such algorithm is *GD with Polyak*

Momentum (Mom-GD) [84], a.k.a. heavy ball method, characterized by

$$d_k = -\alpha_k \nabla f(x_k) + \beta_k d_{k-1}.$$

Another well-known momentum-based algorithm is *GD with Nesterov Acceleration (Acc-GD)* [79] with update vector

$$d_k = -\alpha_k \nabla f(x_k + \beta_k d_{k-1}) + \beta_k d_{k-1}. \tag{9}$$

With a proper choice of the step-sizes α_k and β_k, these schemes can be shown to accelerate the convergence rate, compared to the standard GD, for particular classes of objective functions [84, 81]. The corresponding model-based control algorithms, using the transformations of Table 1, are *Momentum VI (Mom-VI)* [42] with

$$d_k = -\alpha_k \big(v_k - T(v_k)\big) + \beta_k d_{k-1}, \tag{10}$$

and *Accelerated VI (Acc-VI)* [42] with

$$d_k = -\alpha_k \big(v_k + \beta_k d_{k-1} - T(v_k + \beta_k d_{k-1})\big) + \beta_k d_{k-1}.$$

However, the convergence of the preceding accelerated schemes is in general not guaranteed. In [42], the authors address this issue by *safeguarding*, i.e., combining the accelerated VI with the standard VI; see Section 4 for more details.

For accelerating SGD, a direct combination of Polyak momentum or Nesterov acceleration with SGD has been shown to lead to no better (and even worse) performance in terms of convergence rate [107, 54]. At least for almost sure convergence, [71] reports the same rate of convergence for SGD with Polyak momentum and SGD with Nesterov acceleration as for standard SGD. Nevertheless, modifications of momentum-based acceleration methods have led to a range of accelerated SGD algorithms with faster convergence rates with specific assumptions on the problem data [54, 69, 3]. The idea of using momentum for accelerating QL has also attracted some interest. In particular, applying the transformations of Table 1 on a generic *Momentum SGD (Mom-SGD)* [107] with

$$\begin{cases} d'_{k-1} = \nabla \widehat{f}(x_k, \xi_k) - \nabla \widehat{f}(x_{k-1}, \xi_k), \\ d_k = -\alpha_k \nabla \widehat{f}(x_k, \xi_k) - \beta_k d'_{k-1} + \delta_k d_{k-1}, \end{cases}$$

and step-sizes $\alpha_k, \beta_k, \delta_k$, we obtain the *Speedy QL (SQL)* [40], *Nesterov Stochastic Approximation (NeSA)* [30], and *Momentum QL (Mom-QL)* [106] algorithms with

$$\begin{cases} d'_{k-1} = \big(q_k - \widehat{T}(q_k, \varsigma_k^+)\big) - \big(q_{k-1} - \widehat{T}(q_{k-1}, \varsigma_k^+)\big), \\ d_k = -\alpha_k \big(q_k - \widehat{T}(q_k, \varsigma_k^+)\big) - \beta_k d'_{k-1} + \delta_k d_{k-1}. \end{cases} \tag{11}$$

The difference between these three algorithms is in the choice of the step-sizes $\alpha_k, \beta_k, \delta_k$. For example, by setting $\beta_k = \delta_k = (1 - 2\alpha_k) = (k-1)/(k+1)$, we recover SQL.

Anchoring & Halpern iteration. Halpern iteration [45] is an acceleration scheme originally developed for fixed-point iterations involving *non-expansive* maps. In particular, it modifies the updates by gradually pulling the iterates toward a reference point, a.k.a. *anchor*. In the case

of deterministic optimization, modifying the standard GD using Halpern iteration, we derive the *Anchored GD (Anc-GD)* [99]

$$x_{k+1} = \beta_k x + (1 - \beta_k)\big(x_k - \delta_k \nabla f(x_k)\big),$$

where $x \in \mathbb{R}^n$ is the anchor, $\beta_k \in (0, 1)$ is a properly chosen decaying step-size, and δ_k is the step-size corresponding to the standard GD update. The canonical choices for the anchor and the step-size are $x = x_0$ and $\beta_k = \frac{1}{k+2}$ [68]. Using our standard characterization, Anc-GD (with the anchor $x = x_0$) corresponds to the update vector

$$d_k = \beta_k(x_0 - x_k) - \alpha_k \nabla f(x_k),$$

where $\alpha_k = \delta_k(1 - \beta_k)$. Under certain conditions on the objective function f and the step-sizes α_k and β_k, Anc-GD improves the rate of convergence compared to standard GD. Indeed, Anc-GD can be shown to be equivalent to Nesterov acceleration with a correction term [99]. However, when the operation corresponding to standard GD is a *contraction*, the modification via anchoring in Anc-GD does *not* improve the convergence rate [82]. The corresponding model-based control algorithm is *Anchored VI (Anc-VI)* [65] defined by the update vector

$$d_k = \beta_k(v_0 - v_k) - (1 - \beta_k)\big(v_k - T(v_k)\big),$$

where $\alpha_k = (1 - \beta_k)$. A similar, undesired situation arises for Anc-VI algorithm. Since the Bellman operator is a contraction, there is no improvement in the convergence rate for $\gamma < 1$ and the iterates of Anc-VI converge to the optimal value function $v^\star$ linearly with rate γ (i.e., similar to standard VI). However, of importance is the polynomial (i.e., with rate $\mathcal{O}(1/k)$) convergence of the Bellman residual $\|v_k - T(v_k)\|_\infty$ to zero in the *long-horizon* setting with $\gamma \to 1$ [65, Thm. 2]. In particular, even in the *undiscounted* setting (i.e., $\gamma = 1$), the Bellman residual in Anc-VI converges to zero with rate $\mathcal{O}(1/k)$, assuming the corresponding undiscounted Bellman operator has a fixed point [65, Thm. 3]. As a result, in long-horizon and undiscounted settings, Anc-VI maintains robust convergence guarantees, while the geometric rate of standard VI deteriorates.

The corresponding *Stochastic Halpern Iteration (SHI)* [24, 2], characterized by

$$d_k = \beta_k(x_0 - x_k) - \alpha_k \nabla \widehat{f}(x_k, \xi_k),$$

with step-sizes α_k, β_k, has been recently studied extensively in the stochastic optimization setting. In particular, under certain conditions and in combination with other techniques such as variance reduction and restarting, anchoring leads to improved sample complexity [24]. Moreover, anchoring has been shown to be instrumental in providing guarantees for the last iterate, as opposed to a weighted average of iterates, under relaxed bounded-variance assumptions [2]. Applying the transformations of Table 1 on SHI, we derive the *Halpern QL (HQL)* [21] and *Stochastic Anchored Value Iteration for Discounted MDPs (SAVID)* [63] algorithms with

$$d_k = \beta_k(q_0 - q_k) - (1 - \beta_k)\big(q_k - \widehat{T}(q_k, \varsigma_k^+)\big),$$

where $\alpha_k = 1 - \beta_k$. To be precise, HQL uses batch estimation with multiple samples for the sampled Bellman operation in this update rule to reduce the variance, while SAVID incorporates the recursive sampling technique [49] to reduce the sample complexity. Of importance is again that both HQL

and SAVID algorithms (with minor modifications) can also be used in the *averaged cost* setting in which the corresponding Bellman optimality operator is non-expansive but *not* contractive.

PID control. An alternative to standard acceleration methods is to use proportional-integral-derivative (PID) controllers for designing the update rule. This is a well-established technique in automatic control for set-point tracking. The basic idea is to treat the iterates x_k as the "state" of the system and define the "error" signal in terms of the gradient $\nabla f(x_k)$ and possibly the finite difference of the state, i.e., d_k. This idea has recently been explored in training deep networks in [4, 27]. In particular, the *full batch* version of the *PID Optimizer (PID-Opt)* [4] algorithm is a deterministic optimization algorithm characterized by

$$\begin{cases} d_k^I = -\alpha_k \nabla f(x_k) + \beta_k d_{k-1}^I, \\ d_k^D = \nabla f(x_k) - \nabla f(x_{k-1}), \\ d_k = d_k^I + d_k^D, \end{cases}$$

where α_k is the step-size and β_k is the decay coefficient in the integral term. The corresponding model-based control algorithm is *PID accelerated VI (PID-VI)* [35] with

$$\begin{cases} d_k^I = -\alpha_k\big(v_k - T(v_k)\big) + \beta_k d_{k-1}^I, \\ d_k^D = d_{k-1}, \\ d_k = -\kappa_k^P\big(v_k - T(v_k)\big) + \kappa_k^I d_k^I + \kappa_k^D d_k^D. \end{cases}$$

Above, we see all three terms of the PID controller with the corresponding parameters κ_k^P, κ_k^I, κ_k^D. In particular, observe that, compared to PID-Opt, PID-VI uses the finite difference of the state (i.e., d_{k-1}) as opposed to the finite difference of the gradient (i.e., $d'_{k-1} = \nabla f(x_k) - \nabla f(x_{k-1})$) in its derivative term. Moreover, in [35], the authors provide an adaptive scheme for tuning the PID parameters. The close connection between the PID acceleration schemes above and the momentum-based algorithms can be readily seen in their update rules. Indeed, PID-Opt and PID-VI provided above can recover standard momentum-based algorithms for specific values of the PID parameters. For instance, Acc-GD (9) can be derived by removing the derivative term in PID-Opt (i.e., setting $\kappa_k^D = 0$) [4]. Similarly, Mom-VI (10) is an instance of PID-VI without the integral term (i.e., with $\kappa_k^I = 0$) [35].

For stochastic optimization, we can use the *mini-batch* version of PID-Opt [4], characterized by

$$\begin{cases} d_k^I = -\alpha_k \nabla \widehat{f}(x_k) + \beta_k d_{k-1}^I, \\ d_k^D = \beta_k d_{k-1}^D + (1-\beta_k)\big(\nabla \widehat{f}(x_k) - \nabla \widehat{f}(x_{k-1})\big), \\ d_k = d_{k+1}^I + \kappa_k^d d_k^D, \end{cases}$$

where the stochastic gradient operation is evaluated using a mini-batch of samples. In particular, observe that the derivative term also includes averaging to reduce the effect of noise in gradient evaluations. The corresponding model-free control algorithm is *PID accelerated QL (PID-QL)* [8]

with

$$\begin{cases} d_k^I = -\alpha_k\big(q_k - \widehat{T}(q_k, \varsigma_k^+)\big) + \beta_k d_{k-1}^I, \\ q_{k-1}' = (1 - \eta_k)q_{k-2}' + \eta_k q_{k-1}, \\ d_k^D = q_k - q_{k-1}', \\ d_k = -\kappa_k^P\big(q_k - \widehat{T}(q_k, \varsigma_k^+)\big) + \kappa_k^I d_{k+1}^I + \kappa_k^d d_k^D, \end{cases}$$

In particular, observe that PID-QL also uses a stochastic approximation q_{k-1}' of q_{k-1} in forming the derivative term based on the finite difference of the state q_k. Moreover, a scheme similar to that used in PID-VI [35] is used to tune the PID parameters in PID-QL adaptively. We also note that PID-QL can recover Mom-QL [106] and SQL [40] in (11) by setting the PID parameters accordingly [8]. Finally, while the empirical results reported for PID-Opt, PID-VI, and PID-QL show a significant improvement in the convergence rate compared to the corresponding standard methods, theoretical guarantees for the convergence of these algorithms are still lacking.

3.3. **Quasi-Newton methods**

Quasi-Newton methods (QNMs) are a class of methods that allow for a trade-off between computational complexity and (local) convergence rate in solving optimization problems. To do so, these methods use a Newton-type update vector

$$d_k = -\alpha_k \widetilde{H}_k^{-1}\nabla f(x_k),$$

where $\widetilde{H}_k$ (or $\widetilde{H}_k^{-1}$) is a computationally efficient approximation of the true Hessian $\nabla^2 f(x_k)$ (or its inverse) at iteration k. Different QNMs use different approximations of the Hessian or its inverse. Some "classical" examples of QNMs include Broyden approximation [22], Power symmetric Broyden approximation [86], Davidon-Fletcher-Powell approximation [28, 36]. All of these examples approximate the Hessian (or its inverse) as the "closest" matrix to a prior matrix within an affine set defined by (multi-)secant constraints (and possibly other structural constraints such as symmetry, sparsity, etc.). Under certain conditions, these methods lead to a local *superlinear* rate of convergence. Not surprisingly, different types of QNMs have been (directly or with some modifications) employed in solving model-based and model-free control problems. Some recent examples include the Sketched Newton VI [70] and Quasi-PI [58] algorithms. In what follows, we focus on two illustrative examples, namely, Anderson mixing and spectral preconditioning.

Anderson mixing. Anderson mixing, originally developed as an acceleration scheme for fixed-point iterations [5], provides an update rule in which the next iterate is constructed as a linear combination of a finite number of most recent iterates and their images. In particular, *Anderson Accelerated GD (AA-GD)* [76], with memory $m \geq 0$ and $m_k = \min\{k, m\}$, is characterized by the update vector

$$\begin{cases} X_k = [x_k, \ldots, x_{k-m_k}], \\ G_k = [\nabla f(x_k), \ldots, \nabla f(x_{k-m_k})], \\ w_k = (\mathbf{1}^\top(G_k^\top G_k)^{-1}\mathbf{1})^{-1}(G_k^\top G_k)^{-1}\mathbf{1}, \\ d_k = -x_k + (X_k - \alpha_k G_k)w_k, \end{cases}$$

where $\mathbf{1}$ is the all-ones vector, α_k is a properly chosen step-size, and G_k is assumed to be of full column rank (otherwise, we need to either use the pseudo-inverse or some form of regularization). Observe that for $m = 0$, we recover the standard GD. Applying the transformations of Table 1, we derive the *Anderson Accelerated VI (AA-VI)* [39] with

$$
\begin{cases}
V_k = [v_k, \ldots, v_{k-m_k}], \\
G_k = [v_k - T(v_k), \ldots, v_{k-m_k} - T(v_{k-m_k})], \\
w_k = (\mathbf{1}^\top (G_k^\top G_k)^{-1} \mathbf{1})^{-1} (G_k^\top G_k)^{-1} \mathbf{1}, \\
d_k = -v_k + (V_k - G_k) w_k,
\end{cases}
\tag{12}
$$

where $\alpha_k = 1$ and G_k is assumed to be of full column rank. The preceding update rules correspond to the so-called *type-II* Anderson mixing. To be precise, the update vector of AA-GD/VI admits a QNM characterization of the form $d_k = -D_k g(y_k)$, where D_k is an approximation of the *inverse* of the Hessian, i.e., H_k^{-1} [34]. In [109], the authors consider the *type-I* Anderson mixing scheme in combination with GD/VI in which the update vector is of the form $d_k = -D_k^{-1} g(y_k)$ with D_k being an approximation of the Hessian H_k. We note that the combination of Anderson mixing (type-I or type-II) with VI also lacks convergence guarantees. In [109], this issue is again addressed via *safeguarding*.

Anderson mixing has also been used in the stochastic optimization setting. In particular, in [105], the authors use (type-II) Anderson mixing to develop the *Stochastic Anderson Mixing (SAM)* algorithm with update vector

$$
\begin{cases}
D_k^x = [d_{k-m_k}^x, \ldots, d_{k-1}^x] \quad \text{with} \quad d_i^x = x_{i+1} - x_i, \\
D_k^g = [d_{k-m_k}^g, \ldots, d_{k-1}^g] \quad \text{with} \quad d_i^g = \nabla \widehat{f}(x_{i+1}, \xi_{i+1}) - \nabla \widehat{f}(x_i, \xi_i), \\
D_k = \beta_k I - \alpha_k (D_k^x + \beta_k D_k^g) \big((D_k^g)^\top D_k^g + R_k \big)^{-1} (D_k^g)^\top, \\
d_k = -D_k \nabla \widehat{f}(x_k, \xi_k),
\end{cases}
$$

where α_k, β_k are properly chosen step-sizes and $R_k = \delta_k (D_k^x)^\top D_k^x$, with an adaptive coefficient δ_k, acts as a *regularizer*. To be precise, SAM uses a batch of samples for estimation of the stochastic gradient operator. Note that the formulation above aligns with the QNM characterization of Anderson mixing. The corresponding model-free control algorithm is *QL with Stable Anderson Acceleration (SAA-QL)* [95] with

$$
\begin{cases}
D_k^q = [d_{k-m_k}^q, \ldots, d_{k-1}^q] \quad \text{with} \quad d_i^q = q_{i+1} - q_i, \\
D_k^t = [d_{k-m_k}^t, \ldots, d_{k-1}^t] \quad \text{with} \quad d_i^t = \widehat{T}(q_{i+1}, \varsigma_{i+1}^+) - \widehat{T}(q_i, \varsigma_i^+), \\
D_k^g = D_k^q - D_k^t, \\
D_k = \beta_k I - (D_k^q + \beta_k D_k^g) \big((D_k^g)^\top D_k^g + R_k \big)^{-1} (D_k^g)^\top, \\
d_k = -D_k \big(q_k - \widehat{T}(q_k, \varsigma_k^+) \big),
\end{cases}
\tag{13}
$$

where $\alpha_k = 1$ and $R_k = \delta_k (\|D_k^q\|_F^2 + \|D_k^g\|_F^2) I$ as the regularizer. To be precise, SAA-QL in [95] utilizes a *smoothed* version of the Bellman optimality operator based on the *mellow-max* operator;

see Section 4.2 for more details. Moreover, in [95], the proposed SAA is integrated within the Dueling Deep Q-Network (Duel-DQN) [103] algorithm for *continuous-state* model-free control. In particular, the SAA-Duel-DQN algorithm stores the last m_k target networks (i.e., their parameters) and uses (13) for computing the target values in each iteration. Finally, let us note that several works have incorporated the original form of the Anderson mixing similar to (12) for model-free control; see, e.g., [94, 110].

Spectral preconditioning. This class of QNMs involves another computationally efficient approximation of the Hessian. To be precise, they use an approximation of the form

$$\widetilde{H}_k = \alpha_k I + \beta_k U_k W_k^\top,$$

where $U_k, W_k \in \mathbb{R}^{\ell \times r}$ are tall matrices with $r \ll \ell$. Such approximations readily allow for application of the Woodbury matrix identity [44] for computing $\widetilde{H}_k^{-1}$ with a low computational cost. In spectral preconditioning techniques, the matrices U_k and W_k are particularly formed based on the spectral decomposition of the true Hessian. For instance, *GD with Spectral Preconditioning (SP-GD)* [32] is characterized by

$$\begin{cases} D_k = \alpha_k I + \sum_{i=1}^r \lambda_{k,i} u_{k,i} u_{k,i}^\top, \\ d_k = -D_k^{-1} \nabla f(x_k), \end{cases}$$

with α_k being a properly chosen regularizer parameter, $(\lambda_{k,i})_{i=1}^r$ the largest eigenvalues of the Hessian $\nabla^2 f(x_k)$ and $(u_{k,i})_{i=1}^r$ the corresponding eigenvectors. For model-based control, we can mention the recently developed *Rank-One Modified VI (R1-VI)* [55] with

$$\begin{cases} D_k = I - \gamma \mathbf{1} w_k^\top, \\ d_k = -D_k^{-1} \big(v_k - T(v_k) \big), \end{cases}$$

where w_k is the stationary distribution estimate of the Markov chain induced by the greedy policy w.r.t. v_k. The preceding update rule particularly exploits the structure $H(v) = I - \gamma P(v)$ of the "Hessian" in the control problem and employs the spectral decomposition of the matrix $P(v)$. In this regard, we note that w_k and the all-one vector $\mathbf{1}$ are, respectively, the left and right eigenvectors of $P(v_k)$ corresponding to the largest eigenvalue, i.e., 1.

A similar idea has also been used in stochastic optimization. In particular, *Newton-Sampling Method via Rank Thresholding (NewSamp)* [33] is characterized by

$$\begin{cases} D_k = \widehat{\lambda}_{k,r+1} I + \sum_{i=1}^r (\widehat{\lambda}_{k,i} - \widehat{\lambda}_{k,r+1}) \widehat{u}_{k,i} \widehat{u}_{k,i}^\top, \\ d_k = -D_k^{-1} \nabla \widehat{f}(x_k, \xi_k), \end{cases}$$

with eigenvalues $(\widehat{\lambda}_{k,i})_{i=1}^{r+1}$ and the corresponding eigenvectors $(\widehat{u}_{k,i})_{i=1}^r$ computed using eigenvalue decomposition of the *batch-sampled* Hessian, that is, $\frac{1}{B} \sum_{j=1}^B \nabla^2 \widehat{f}(x_k, \xi_{k,j})$. For model-free control, the same idea is used in *Rank-One Modified QL (R1-QL)* [55] with

$$\begin{cases} D_k = I - \gamma \mathbf{1} \widehat{w}_k^\top, \\ d_k = -\alpha_k D_k^{-1} \big(q_k - \widehat{T}(q_k, \varsigma_k^+) \big), \end{cases}$$

where $\widehat{w}_k$ is now the state-action stationary distribution of the Markov chain induced by the greedy policy w.r.t. q_k and estimated using the sample transitions. Once again, the preceding update rule exploits the structure of the "Hessian" in the control problem.

We note that the spectral preconditioning technique is closely related to the *matrix splitting* method for solving linear equations. In particular, Deflated Dynamics VI (DDVI) and Temporal Difference (DDTD) algorithms [64] also use a spectral decomposition of the transition probability matrix to form a low-rank approximation of it. They then use this low-rank approximation in combination with matrix splitting for the *policy evaluation* problem. In effect, the corresponding update rules follow the same structure as R1-VI/QL, except with rank-r approximation of the transition probability matrix where $r > 1$ (as opposed to $r = 1$).

4. Convergence of Control Algorithms

One of the promises of building a common formulation for control and optimization algorithms is to borrow existing theoretical results from one field to another. However, as several other studies have pointed out, the similarity in the update rule does not carry over to the analysis. One of the main reasons for this discrepancy is the *non-differentiability* of the Bellman optimality operator T. Another complicating factor is that the operator T is a contraction in ∞-norm, which is also *non-differentiable*. This forces the convergence analysis of the control algorithms to rely more on fixed-point theory and contraction mappings than on gradient-based smooth analysis.

In what follows, we examine general tools for the convergence analysis of control algorithms, as well as some simple yet effective remedies for ensuring the convergence of novel control algorithms.

4.1. Model-based algorithms

For *finite* state-action MDPs, the non-differentiability of the operator T is "manageable." Indeed, for such MDPs, T is piece-wise affine and we have

$$T(v) = \min_{\pi \in \mathcal{S}^{\mathcal{A}}} c^{\pi} + \gamma P^{\pi} v, \quad \forall v \in \mathbb{R}^{\mathcal{S}},$$

where the minimization is over the finite set of all deterministic control policies, each corresponding to a linear function $c^{\pi} + \gamma P^{\pi} v$. Equivalently, the finite family of policies admits an affine switching system [67], a connection leveraged by [62] to establish convergence guarantees for asynchronous Q-learning. Moreover, by Rademacher's Theorem [102, Thm. 10.8], T is differentiable almost everywhere. This property allows one to use tools from semismooth analysis for deriving local convergence results. In particular, in [38], the authors use the fact that the PI algorithm is an instance of the semi-smooth Newton method to show that it has a local quadratic convergence rate. Similarly, in [58], the semismoothness of T is used for deriving a local superlinear convergence rate for one of the proposed quasi-PI algorithms.

A more general approach, however, is to completely get rid of the non-differentiability by *smoothing* the Bellman optimality operator. Not surprisingly, a similar technique has been extensively used to develop algorithms for solving non-smooth optimization problems; see, e.g., the seminal work [80].

The smoothing of T is achieved by approximating the minimization operation by a differentiable function. As an example, by using the *log-sum-exp* (a.k.a. *soft-max*) function [91], we can define

$$[\bar{T}_\beta(q)](s,a) := c(s,a) + \gamma \, \mathbb{E}_{s^+}\big[-\tfrac{1}{\beta}\log\big(\textstyle\sum_{a^+\in\mathcal{A}} \mathrm{e}^{-\beta q(s^+,a^+)}\big) \,\big|\, (s,a)\big], \quad \forall (s,a)\in\mathcal{S}\times\mathcal{A},$$

where $\beta > 0$ is the so-called temperature parameter. The operator $\bar{T}_\beta$ is an over-approximation of the true Bellman optimality operator $\bar{T} = \mathbb{E}_{\varsigma^+}[\widehat{T}(\cdot,\varsigma^+)]$ and, in particular, $\bar{T}_\beta \to \bar{T}$ as $\beta \to \infty$. Accordingly, its fixed-point q^β is not the optimal Q-function, while the gap $\|q^\beta - q^\star\|_\infty$ can be controlled via the temperature parameter β. However, such smoothing allows for a direct application of the second-order scheme for finding the fixed-point of $\bar{T}_\beta$ [91]. This idea has been recently used to propose the generalized second-order VI with a quadratic convergence rate [51].

Besides the general tools discussed above, there are simple yet efficient "safeguarding" techniques for ensuring the convergence of novel model-based algorithms that lack convergence guarantees. These techniques specifically exploit the fact that the Bellman optimality operator T is a γ-contraction. To be precise, let us consider a generic model-based update rule

$$v_{k+1} = v_k + d_k,$$

characterized by the novel update vector d_k. One simple technique for ensuring the convergence of the preceding update rule is to safeguard it against VI [42]. For some $\gamma' \in [\gamma, 1)$, we consider the *safeguarded update rule*

$$
\begin{aligned}
&v_{k+1} = v_k + d_k; \\
&\textbf{if } \|v_{k+1} - T(v_{k+1})\|_\infty > (\gamma')^k \, \|v_0 - T(v_0)\|_\infty \\
&\qquad v_{k+1} = T(v_k); \\
&\textbf{endif}
\end{aligned}
\tag{14}
$$

The preceding update rule has the following convergence guarantee (for completeness, we provide the proof in Appendix A.1; see also [42, Thm. 4]):

Theorem 4.1 (Convergence via safeguarding against VI). *Let $\gamma' \in [\gamma, 1)$. The safeguarded update rule* (14), *with arbitrary update vector d_k, converges linearly to the optimal value function $v^\star = T(v^\star)$ with rate γ'.*

Alternatively, we can modify the update rule and use backtracking to ensure the convergence [58]. Specifically, we modify the update rule as

$$v_{k+1} = T(v_k) + \alpha_k\big(v_k - T(v_k) + \beta_k d_k\big),$$

by introducing the coefficients α_k and β_k. Observe that we have the original update rule by setting $\alpha_k = \beta_k = 1$. The coefficient α_k is determined via backtracking to ensure that the Bellman error $\|v_{k+1} - T(v_{k+1})\|_\infty$ is reduced in each iteration linearly with some rate $\gamma' \in (\gamma, 1)$. The coefficient β_k, on the other hand, is introduced to ensure that the novel update vectors are bounded and hence

the backtracking terminates in finitely many steps. To be precise, we consider the *update rule with backtracking*

$$(0) \quad \alpha_k = 1;$$

$$(1) \quad v_{k+1} = T(v_k) + \alpha_k\big(v_k - T(v_k) + \beta_k d_k\big);$$

$$(2) \quad \textbf{if } \|v_{k+1} - T(v_{k+1})\|_\infty > \gamma' \, \|v_k - T(v_k)\|_\infty \tag{15}$$

$$\alpha_k \leftarrow \lambda\alpha_k \text{ and go to step } (1) \text{ ;}$$

endif

The preceding update rule has the following convergence properties (for completeness, we provide the proof in Appendix A.2; see also [58, Sec. III.B.2]):

Theorem 4.2 (Convergence via backtracking against VI). *Let $\gamma' \in (\gamma, 1)$, $\lambda \in (0,1)$, and*

$$\beta_k = \frac{\min\{\|v_k - T(v_k)\|_\infty, \, \|d_k\|_\infty\}}{\|d_k\|_\infty}, \quad \forall k \geq 0.$$

Then, the backtracking scheme in (15) *is guaranteed to terminate in $\mathcal{O}\big(\ln(\gamma' - \gamma)/\ln(\lambda)\big)$ steps and the corresponding update rule, with arbitrary update vector d_k, converges linearly to the optimal value function $v^\star = T(v^\star)$ with rate γ'.*

Observe that the safeguarding and backtracking methods described above lead to a convergence behavior (i.e., linear with rate γ') and per-iteration computational complexity (i.e., $\mathcal{O}(n^2 m)$ discarding logarithmic factors) similar to the standard VI algorithm. This is not surprising considering the fact that these methods, similar to VI, exploit the contractivity of the Bellman optimality operator T. Nevertheless, these methods enable the safe integration of novel, optimization-inspired update rules (encapsulated in d_k) into a provably convergent control algorithm [42, 58].

4.2. Model-free algorithms

Similar to a model-based case, one can benefit from smoothing approaches in model-free scenarios. In simple words, smoothing out the kinks in the Bellman optimality operator or the optimal policies allows us to borrow well-developed tools for stochastic optimization of smooth functions. The recently proposed *mellow-max* operator [7] is another excellent example of such smoothing that can be used as a differentiable approximation of the minimization within the Bellman optimality operator. Besides being a smooth operator, the mellow-max operator is, more importantly, non-expansive in the ∞-norm. This property then streamlines the usage of standard convergence techniques seamlessly. The corresponding smoothed Bellman optimality operator using the mellow-max is defined as

$$[T_\omega(q)](s,a) := c(s,a) + \gamma \, \mathbb{E}_{s^+}\big[-\tfrac{1}{\omega} \log \big(\tfrac{1}{|\mathcal{A}|} \textstyle\sum_{a^+ \in \mathcal{A}} e^{-\omega q(s^+, a^+)}\big) \mid (s,a)\big], \quad \forall(s,a) \in \mathcal{S} \times \mathcal{A},$$

with $\omega > 0$ being the temperature parameter that can be used to control the gap $\|q^\omega - q^\star\|_\infty$ between the fixed-point q^ω of T_ω and the true optimal Q-function $q^\star = \bar{T}(q^\star)$. The smoothed operator T_ω is an under-approximation of the true Bellman optimality operator $\bar{T}$ and, in particular, we have

$T_\omega \to \bar{T}$ as $\omega \to \infty$. The operator T_ω has been recently used in combination with Anderson mixing to improve the convergence behavior in policy iteration [95].

In the model-free case, algorithmic updates are inherently noisy, being driven by samples of the dynamics and costs. This stochasticity naturally places these algorithms within the scope of *stochastic approximation* (*SA*). A standard SA iteration is

$$x_{k+1} = x_k + \alpha_k\big(h(x_k) + M_{k+1}\big), \tag{16}$$

where α_k is the step-size (or the so-called *learning rate*), $h : \mathbb{R}^\ell \to \mathbb{R}^\ell$ is the *mean field* (expected update), and M_k is a martingale-difference noise. A common approach to studying the limiting behavior of an SA method is the *ordinary differential equation* (*ODE*) method. The ODE method (with a dynamical-systems viewpoint), originating with Ljung [73], analyzes (16) by comparing it with the flow of

$$\dot{x}(t) = h\big(x(t)\big),$$

and drawing conclusions about the iterates from the asymptotics of this ODE. The interested reader is referred to [25, 61, 19] for comprehensive treatments. Invariance principles from nonlinear systems (e.g., LaSalle's invariance principle; see [53, Ch. 4]) then identify invariant/limit sets of the ODE toward which trajectories converge [10, 11]. Under standard SA assumptions, the SA iterates track these sets. To guarantee almost-sure boundedness (or Lyapunov stability of the iterates), one imposes verifiable drift/Lyapunov conditions. See [20] for the ODE method with drift at infinity and [6] for global Lyapunov constructions. It is worth mentioning that an important RL-specific deviation from otherwise standard SA conditions is *ergodicity of the state process* for policy evaluation and *sufficient exploration* for control.

Over the years, the ODE framework, in particular the framework introduced by [20], has become a standard tool for establishing stability and convergence of a broad class of model-free control/RL algorithms. Considering the *average cost problems*, in [1], the authors provided for the first time a complete analysis of two Q-learning algorithms by coupling the ODE style analysis with Kushner-Clark argumentation style [60]. Another major subclass of RL methods is the well-known *actor-critic* methods, which can be characterized as *coupled, two-timescale* stochastic iterations: the critic evolves on a *fast* scale to track the value of the current policy, and the actor evolves on a *slow* scale using the critic's signal to adjust the policy. The ODE viewpoint makes this decomposition explicit, which, under the usual SA conditions together with the RL ones, yields clean convergence guarantees to a stationary policy [18, 83]. Another example is the use of the ODE method to show the convergence of *gradient temporal difference* (*TD*) *approaches* under the general setup of off-policy learning and linear function approximation [97]. This work has then been extended to any smooth value function approximation in [74] and to a control setting (policy improvement) under a stationary behavioral policy in [75]. Recently, ODE-style arguments have been used not only to prove convergence but also to *design* RL algorithms. In [31], the authors introduce a *matrix preconditioning* for the TD direction, improving the transient behavior of the proposed algorithm. This matrix gain is learned on a faster timescale and tracks the inverse Jacobian of the mean field. This design extends to nonlinear function approximation under standard regularity assumptions [26].

As an example of application of results developed using ODE methods for SAs, namely [83], we finish this section by providing a simple and efficient "safeguarding" technique for ensuring the convergence of novel model-free algorithms that lack convergence guarantees. To this end, let us consider a generic update rule

$$q_{k+1} = q_k + \alpha_k b_k,$$

characterized by the novel update vector $d_k = \alpha_k b_k$. Here, α_k is the learning rate satisfying the standard Robbins–Monro condition (i.e., $\sum_k \alpha_k = \infty$ and $\sum_k \alpha_k^2 < \infty$) and b_k is a vector depending on the sampled Bellman error, a.k.a. temporal difference (i.e., $\widehat{T}(q_k, \varsigma_k^+) - q_k$). To address the convergence issue, one possibility is to consider a convex combination of the standard QL algorithm update rule (3.1) with this novel update rule with a diminishing effect [58]. To be precise, we modify the update rule as follows

$$q_{k+1} = q_k + \alpha_k \left((1 - \beta_k)\big(\widehat{T}(q_k, \varsigma_k^+) - q_k\big) + \beta_k b_k \right)$$
$$= q_k + \alpha_k \left(\widehat{T}(q_k, \varsigma_k^+) - q_k + \beta_k\big(b_k + q_k - \widehat{T}(q_k, \varsigma_k^+)\big) \right),$$

where β_k is a diminishing coefficient (i.e., $\beta_k \to 0$). Finally, to ensure convergence, we need to bound the effect of the extra term $p_k = b_k + q_k - \widehat{T}(q_k, \varsigma_k^+)$. For that, given some $\rho > 0$, we employ the operator $B_\rho : \mathbb{R}^{\mathcal{S} \times \mathcal{A}} \to \mathbb{R}^{\mathcal{S} \times \mathcal{A}}$ defined by

$$B_\rho(p) := \frac{\min\{\rho, \|p\|_\infty\}}{\|p\|_\infty} p.$$

This leads to the *safeguarded update rule*

$$
\begin{aligned}
p_k &= b_k - q_k + \widehat{T}(q_k, \varsigma_k^+), \\
q_{k+1} &= q_k + \alpha_k\big(\widehat{T}(q_k, \varsigma_k^+) - q_k + \beta_k B_\rho(p_k)\big),
\end{aligned}
\tag{17}
$$

with the following convergence property (for completeness, we provide the proof in Appendix A.3; see also [58, Sec. III.C]):

Theorem 4.3 (Convergence via safeguarding against QL)**.** *Consider the safeguarded update rule* (17). *Let $\rho > 0$ be given and assume the learning rates α_k and β_k are such that $\sum_k \alpha_k = \infty$, $\sum_k \alpha_k^2 < \infty$, and $\beta_k \to 0$. Then, the iterates q_k converge to $q^\star$ almost surely.*

We note that the safeguarding technique described above leads to the same asymptotic convergence behavior as QL due to the diminishing effect of the extra term p_k (which includes the novel direction b_k). However, it does not involve any extra sample complexity since the computation of the vector b_k already requires us to have access to the sampled Bellman error $\widehat{T}(q_k, \varsigma_k^+) - q_k$.

Appendix A. Technical Proofs

A.1. Proof of Theorem 4.1

We will use induction to show that

$$\|v_k - T(v_k)\|_\infty \le (\gamma')^k \|v_0 - T(v_0)\|_\infty, \quad \forall k \ge 0. \tag{18}$$

The base case $k = 0$ holds trivially. Let us now assume the inequality (18) holds for some $k \geq 0$ and recall that T is a γ-contraction in ∞-norm. Then, the safeguarding against standard VI as in (14) implies that

$$
\begin{aligned}
\|v_{k+1} - T(v_{k+1})\|_\infty &\leq \max\{(\gamma')^{k+1} \|v_0 - T(v_0)\|_\infty, \ \gamma \|v_k - T(v_k)\|_\infty\} \\
&\leq \max\{(\gamma')^{k+1} \|v_0 - T(v_0)\|_\infty, \ \gamma' \|v_k - T(v_k)\|_\infty\} \\
&\leq (\gamma')^{k+1} \|v_0 - T(v_0)\|_\infty,
\end{aligned}
$$

where we used $\gamma' \geq \gamma$ for the second inequality and the induction hypothesis for the last inequality. This completes the proof.

A.2. **Proof of Theorem 4.2**

Note that if $\|v_k - T(v_k)\|_\infty = 0$, we clearly have $v_k = v^\star$. Hence, we can assume $v_k \neq v^\star$. Then, using [58, Lem. 3.3], we have that $\|v_{k+1} - T(v_{k+1})\|_\infty \leq \gamma' \|v_k - T(v_k)\|_\infty$ if

$$
\|v_{k+1} - T(v_k)\|_\infty = \alpha_k \|v_k - T(v_k) + \beta_k d_k\|_\infty \leq \frac{\gamma' - \gamma}{1 + \gamma} \|v_k - T(v_k)\|_\infty
$$

That is, backtracking terminates after $\ell \geq 0$ steps if

$$
\alpha_k^{-1} = \lambda^{-\ell} \geq \frac{1 + \gamma}{\gamma' - \gamma} \cdot \frac{\|v_k - T(v_k) + \beta_k d_k\|_\infty}{\|v_k - T(v_k)\|_\infty}.
$$

Now, observe that $1 + \gamma \leq 2$ and the choice of β_k implies that

$$
\|v_k - T(v_k) + \beta_k d_k\|_\infty \leq \|v_k - T(v_k)\|_\infty + \beta_k \|d_k\|_\infty \leq 2 \|v_k - T(v_k)\|_\infty.
$$

This, in turn, implies that the backtracking terminates after $\ell \leq L$ steps, where

$$
\lambda^{-L} = \frac{4}{\gamma' - \gamma} \iff L = \mathcal{O}\left(\ln(\gamma' - \gamma)/\ln(\lambda)\right).
$$

The linear convergence of the iterates v_k of update rule with backtracking to $v^\star$ with rate γ' then immediately follows from the condition $\|v_{k+1} - T(v_{k+1})\|_\infty \leq \gamma' \|v_k - T(v_k)\|_\infty$, which implies

$$
\|v_k - T(v_k)\|_\infty \leq (\gamma')^k \|v_0 - T(v_0)\|_\infty, \quad \forall k \geq 0.
$$

A.3. **Proof of Theorem 4.3**

Define

$$
h(q) := \bar{T}(q) - q, \ M_k := \widehat{T}(q_k, \varsigma_k^+) - \bar{T}(q_k), \ r_k := \beta_k B_\rho(p_k),
$$

where $\bar{T}$ is the corresponding true Bellman optimality operator for Q-functions, that is, $\bar{T}(q) = \mathbb{E}_{\varsigma^+}[\widehat{T}(q, \varsigma^+)]$ for each q. Then, the convergence of q_k to the unique equilibrium $q^\star$ of the map h follows from [83, Cor. 3.2]. In particular, observe that $r_k \to 0$ since $\beta_k \to 0$ and $\|B_\rho(p_k)\|_\infty \leq \rho$ for all $k \geq 0$.

References

[1] Abounadi, J., Bertsekas, D. P., and Borkar, V. S. (2001). Learning algorithms for Markov decision processes with average cost. *SIAM Journal on Control and Optimization*, 40(3):681–698.

[2] Alacaoglu, A., Malitsky, Y., and Wright, S. J. (2025). Towards weaker variance assumptions for stochastic optimization. *arXiv preprint arXiv:2504.09951*.

[3] Allen-Zhu, Z. (2017). Katyusha: the first direct acceleration of stochastic gradient methods. *Journal of Machine Learning Research*, 18(1):8194–8244.

[4] An, W., Wang, H., Sun, Q., Xu, J., Dai, Q., and Zhang, L. (2018). A PID controller approach for stochastic optimization of deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8522–8531.

[5] Anderson, D. G. (1965). Iterative procedures for nonlinear integral equations. *Journal of the ACM (JACM)*, 12(4):547–560.

[6] Andrieu, C., Moulines, É., and Priouret, P. (2005). Stability of stochastic approximation under verifiable conditions. *SIAM Journal on Control and Optimization*, 44(1):283–312.

[7] Asadi, K. and Littman, M. L. (2017). An alternative softmax operator for reinforcement learning. In *International Conference on Machine Learning*, pages 243–252.

[8] Bedaywi, M., Rakhsha, A., and Farahmand, A.-m. (2025). PID accelerated temporal difference algorithms. *Reinforcement Learning Journal*, 5:2071–2095.

[9] Bellman, R. (1957). A Markovian decision process. *Journal of Mathematics and Mechanics*, 6(5):679–684.

[10] Benaïm, M. (1996). A dynamical system approach to stochastic approximations. *SIAM Journal on Control and Optimization*, 34(2):437–472.

[11] Benaïm, M. (1999). Dynamics of stochastic approximation algorithms. In *Séminaire de Probabilités XXXIII*, volume 1709, pages 1–68. Springer.

[12] Berthier, E. and Bach, F. (2020). Max-Plus linear approximations for deterministic continuous-state Markov decision processes. *IEEE Control Systems Letters*, 4(3):767–772.

[13] Bertsekas, D. (2022a). *Lessons from AlphaZero for optimal, model predictive, and adaptive control*. Athena Scientific.

[14] Bertsekas, D. P. (1975). Convergence of discretization procedures in dynamic programming. *IEEE Transactions on Automatic Control*, 20(3):415–419.

[15] Bertsekas, D. P. (2011). Temporal difference methods for general projected equations. *IEEE Transactions on Automatic Control*, 56(9):2128–2139.

[16] Bertsekas, D. P. (2022b). *Abstract dynamic programming*. Athena Scientific.

[17] Bertsekas, D. P. and Tsitsiklis, J. N. (1996). *Neuro-dynamic programming*. Athena Scientific.

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

[19] Borkar, V. S. (2008). *Stochastic approximation: a dynamical systems viewpoint*. Cambridge University Press.

[20] Borkar, V. S. and Meyn, S. P. (2000). The O.D.E. method for convergence of stochastic approximation and reinforcement learning. *SIAM Journal on Control and Optimization*, 38(2):447–469.

[21] Bravo, M. and Contreras, J. P. (2024). Stochastic Halpern iteration in normed spaces and applications to reinforcement learning. *arXiv preprint arXiv:2403.12338*.

[22] Broyden, C. G. (1965). A class of methods for solving nonlinear simultaneous equations. *Mathematics of Computation*, 19(92):577–593.

[23] Bubeck, S. (2015). Convex optimization: algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357.

[24] Cai, X., Song, C., Guzmán, C., and Diakonikolas, J. (2022). Stochastic Halpern iteration with variance reduction for stochastic monotone inclusions. In *Advances in Neural Information Processing Systems*, volume 35, pages 24766–24779.

[25] Chen, H.-F. (2002). *Stochastic approximation and its applications*, volume 64. Springer.

[26] Chen, S., Devraj, A. M., Lu, F., Busic, A., and Meyn, S. P. (2020). Zap Q-learning with nonlinear function approximation. In *Advances in Neural Information Processing Systems*, volume 33, pages 16879–16890.

[27] Chen, S., Liu, J., Wang, P., Xu, C., Cai, S., and Chu, J. (2024). Accelerated optimization in deep learning with a proportional-integral-derivative controller. *Nature Communications*, 15(1):10263.

[28] Davidon, W. C. (1991). Variable metric method for minimization. *SIAM Journal on Optimization*, 1(1):1–17.

[29] De Farias, D. P. and Van Roy, B. (2004). On constraint sampling in the linear programming approach to approximate dynamic programming. *Mathematics of Operations Research*, 29(3):462–478.

[30] Devraj, A. M., Bušić, A., and Meyn, S. P. (2019). On matrix momentum stochastic approximation and applications to Q-learning. In *Annual Allerton Conference on Communication, Control, and Computing*, pages 749–756.

[31] Devraj, A. M. and Meyn, S. P. (2017). Zap Q-learning. In *Advances in Neural Information Processing Systems*, volume 30.

[32] Doikov, N., Stich, S. U., and Jaggi, M. (2024). Spectral preconditioning for gradient methods on graded non-convex functions. In *International Conference on Machine Learning*, pages 11227–11252.

[33] Erdogdu, M. A. and Montanari, A. (2015). Convergence rates of sub-sampled Newton methods. In *Advances in Neural Information Processing Systems*, volume 28.

[34] Fang, H.-r. and Saad, Y. (2009). Two classes of multisecant methods for nonlinear acceleration. *Numerical Linear Algebra with Applications*, 16(3):197–221.

[35] Farahmand, A.-M. and Ghavamzadeh, M. (2021). PID accelerated value iteration algorithm. In *International Conference on Machine Learning*, pages 3143–3153.

[36] Fletcher, R. and Powell, M. J. (1963). A rapidly convergent descent method for minimization. *Computer Journal*, 6(2):163–168.

[37] Frank, M. and Wolfe, P. (1956). An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110.

[38] Gargiani, M., Zanelli, A., Liao-McPherson, D., Summers, T., and Lygeros, J. (2022). Dynamic programming through the lens of semismooth N ewton-type methods. *IEEE Control Systems Letters*, 6:2996–3001.

[39] Geist, M. and Scherrer, B. (2018). Anderson acceleration for reinforcement learning. *arXiv preprint arXiv:1809.09501*.

[40] Ghavamzadeh, M., Kappen, H., Azar, M., and Munos, R. (2011). Speedy Q-learning. In *Advances in Neural Information Processing Systems*, volume 24.

[41] Gonçalves, V. M. (2021). Max-Plus approximation for reinforcement learning. *Automatica*, 129:109623.

[42] Goyal, V. and Grand-Clément, J. (2022). A first-order approach to accelerated value iteration. *Operations Research*, 71(2):517–535.

[43] Grand-Clément, J. (2021). From convex optimization to MDPs: a review of first-order, second-order and quasi-Newton methods for MDPs. *arXiv preprint arXiv:2104.10677*.

[44] Hager, W. W. (1989). Updating the inverse of a matrix. *SIAM review*, 31(2):221–239.

[45] Halpern, B. (1967). Fixed points of nonexpanding maps. *Bulletin of the American Mathematical Society*, 73(6):957–961.

[46] Hernández-Lerma, O. and Lasserre, J. B. (2012a). *Discrete-time Markov control processes: basic optimality criteria*, volume 30. Springer Science & Business Media.

[47] Hernández-Lerma, O. and Lasserre, J. B. (2012b). *Further topics on discrete-time Markov control processes*, volume 42. Springer Science & Business Media.

[48] Howard, R. A. (1960). *Dynamic programming and Markov processes*. John Wiley.

[49] Jin, Y., Karmarkar, I., Sidford, A., and Wang, J. (2024). Truncated variance reduced value iteration. In *Advances in Neural Information Processing Systems*, volume 37, pages 117481–117508.

[50] Kakade, S. and Langford, J. (2002). Approximately optimal approximate reinforcement learning. In *International Conference on Machine Learning*, pages 267–274.

[51] Kamanchi, C., Diddigi, R. B., and Bhatnagar, S. (2022). Generalized second order value iteration in Markov decision processes. *IEEE Transactions on Automatic Control*, 67(8):4241–4247.

[52] Kearns, M. and Singh, S. (1998). Finite-sample convergence rates for Q-learning and indirect algorithms. In *Advances in Neural Information Processing Systems*, volume 11.

[53] Khalil, H. K. (2002). *Nonlinear systems*. Prentice Hall.

[54] Kidambi, R., Netrapalli, P., Jain, P., and Kakade, S. (2018). On the insufficiency of existing momentum schemes for stochastic optimization. In *Information Theory and Applications Workshop*, pages 1–9.

[55] Kolarijani, A. S., Ok, T., Mohajerin Esfahani, P., and Sharifi Kolarijani, M. A. (2025). Rank-one modified value iteration. In *International Conference on Machine Learning*, pages 31182–31201.

[56] Kolarijani, M. A. S., Max, G. F., and Mohajerin Esfahani, P. (2021). Fast approximate dynamic programming for infinite-horizon M arkov decision processes. In *Advances in Neural Information Processing Systems*, volume 34, pages 23652–23663.

[57] Kolarijani, M. A. S. and Mohajerin Esfahani, P. (2023). Fast approximate dynamic programming for input-affine dynamics. *IEEE Transactions on Automatic Control*, 68(10):6315–6322.

[58] Kolarijani, M. A. S. and Mohajerin Esfahani, P. (2025). From optimization to control: quasi policy iteration. *arXiv preprint arXiv:2311.11166*.

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

[60] Kushner, H. J. and Clark, D. S. (1978). *Stochastic approximation methods for constrained and unconstrained systems*, volume 26. Springer-Verlag.

[61] Kushner, H. J. and Yin, G. G. (2003). *Stochastic approximation and recursive algorithms and applications*, volume 35. Springer.

[62] Lee, D. and He, N. (2020). A unified switching system perspective and convergence analysis of q-learning algorithms. *Advances in neural information processing systems*, 33:15556–15567.

[63] Lee, J., Bravo, M., and Cominetti, R. (2025a). Near-optimal sample complexity for MDPs via anchoring. In *International Conference on Machine Learning*, pages 32907–32929.

[64] Lee, J., Rakhsha, A., Ryu, E. K., and Farahmand, A.-m. (2025b). Deflated dynamics value iteration. *Transactions on Machine Learning Research*.

[65] Lee, J. and Ryu, E. K. (2023). Accelerating value iteration with anchoring. In *Advances in Neural Information Processing Systems*, volume 36, pages 53924–53963.

[66] Lemaréchal, C. (2012). Cauchy and the gradient method. *Documenta Mathematica Extra*, pages 251–254.

[67] Liberzon, D. (2003). *Switching in systems and control*, volume 190. Springer.

[68] Lieder, F. (2021). On the convergence rate of the Halpern-iteration. *Optimization Letters*, 15(2):405–418.

[69] Liu, C. and Belkin, M. (2020). Accelerating SGD with momentum for over-parameterized learning. In *International Conference on Learning Representations*.

[70] Liu, J., Xie, C., Deng, Q., Ge, D., and Ye, Y. (2024). Sketched Newton value iteration for large-scale Markov decision processes. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(12):13936–13944.

[71] Liu, J. and Yuan, Y. (2022). On almost sure convergence rates of stochastic gradient methods. In *Conference on Learning Theory*, pages 2963–2983.

[72] Liu, Y. and Kolarijani, M. A. S. (2024). Fitted Q-iteration via Max-Plus-linear approximation. *IEEE Control Systems Letters*, 8:3201–3206.

[73] Ljung, L. (1977). Analysis of recursive stochastic algorithms. *IEEE Transactions on Automatic Control*, 22(4):551–575.

[74] Maei, H. R., Szepesvári, C., Bhatnagar, S., Precup, D., Silver, D., and Sutton, R. S. (2009). Convergent temporal-difference learning with arbitrary smooth function approximation. In *Advances in Neural Information Processing Systems*.

[75] Maei, H. R., Szepesvári, C., Bhatnagar, S., and Sutton, R. S. (2010). Toward off-policy learning control with function approximation. In *International Conference on Machine Learning*.

[76] Mai, V. and Johansson, M. (2020). Anderson acceleration of proximal gradient methods. In *International Conference on Machine Learning*, pages 6620–6629.

[77] McEneaney, W. M. (2006). *Max-Plus methods for nonlinear control and estimation*. Springer Science & Business Media.

[78] Mohajerin Esfahani, P., Sutter, T., Kuhn, D., and Lygeros, J. (2018). From infinite to finite programs: explicit error bounds with applications to approximate dynamic programming. *SIAM Journal on Optimization*, 28(3):1968–1998.

[79] Nesterov, Y. (1983). A method for solving the convex programming problem with convergence rate $o(1/k^2)$. In *Doklady Akademii Nauk SSSR*, volume 269, pages 543–547.

[80] Nesterov, Y. (2005). Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152.

[81] Nesterov, Y. (2018). *Lectures on convex optimization*. Springer.

[82] Park, J. and Ryu, E. K. (2022). Exact optimal accelerated complexity for fixed-point iterations. In *International Conference on Machine Learning*, pages 17420–17457.

[83] Perkins, S. and Leslie, D. S. (2012). Asynchronous stochastic approximation with differential inclusions. *Stochastic Systems*, 2(2):409–446.

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

[85] Porteus, E. L. and Totten, J. C. (1978). Accelerated computation of the expected discounted return in a Markov chain. *Operations Research*, 26(2):350–358.

[86] Powell, M. J. (1970). A new algorithm for unconstrained optimization. In *Nonlinear programming*, pages 31–65. Elsevier.

[87] Powell, W. B. (2007). *Approximate dynamic programming: solving the curses of dimensionality*, volume 703. John Wiley & Sons.

[88] Puterman, M. L. and Brumelle, S. L. (1979). On the convergence of policy iteration in stationary dynamic programming. *Mathematics of Operations Research*, 4(1):60–69.

[89] Robbins, H. and Monro, S. (1951). A stochastic approximation method. *Annals of Mathematical Statistics*, 22(3):400–407.

[90] Ruppert, D. (1985). A Newton-Raphson version of the multivariate Robbins-M onro procedure. *Annals of Statistics*, 13(1):236–245.

[91] Rust, J. (1994). Structural estimation of Markov decision processes. *Handbook of Econometrics*, 4:3081–3143.

[92] Samuelson, P. A. (1948). Foundations of economic analysis. *Science and Society*, 13(1).

[93] Schmidhuber, J. (2015). Deep learning in neural networks: an overview. *Neural Networks*, 61:85–117.

[94] Shi, W., Song, S., Wu, H., Hsu, Y.-C., Wu, C., and Huang, G. (2019). Regularized Anderson acceleration for off-policy deep reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 32.

[95] Sun, K., Wang, Y., Liu, Y., Pan, B., Jui, S., Jiang, B., Kong, L., et al. (2021). Damped Anderson mixing for deep reinforcement learning: acceleration, convergence, and stabilization. In *Advances in Neural Information Processing Systems*, volume 34, pages 3732–3743.

[96] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: an introduction*. MIT press.

[97] Sutton, R. S., Maei, H. R., Precup, D., Bhatnagar, S., Silver, D., Szepesvári, C., and Wiewiora, E. (2009). Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *International Conference on Machine Learning*.

[98] Szepesvári, C. (2010). *Algorithms for reinforcement learning*. Morgan & Claypool.

[99] Tran-Dinh, Q. (2024). From Halpern's fixed-point iterations to Nesterov's accelerated interpretations for root-finding problems. *Computational Optimization and Applications*, 87(1):181–218.

[100] Tsitsiklis, J. N. and Van Roy, B. (1997). An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42(5):674–690.

[101] Vieillard, N., Pietquin, O., and Geist, M. (2019). On connections between constrained optimization and reinforcement learning. *arXiv preprint arXiv:1910.08476*.

[102] Villani, C. et al. (2008). *Optimal transport: old and new*, volume 338. Springer.

[103] Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., and Freitas, N. (2016). Dueling network architectures for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1995–2003.

[104] Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3):279–292.

[105] Wei, F., Bao, C., and Liu, Y. (2021). Stochastic Anderson mixing for nonconvex stochastic optimization. In *Advances in Neural Information Processing Systems*, volume 34, pages 22995–23008.

[106] Weng, B., Xiong, H., Zhao, L., Liang, Y., and Zhang, W. (2021). Finite-time theory for momentum Q-learning. In *Conference on Uncertainty in Artificial Intelligence*, pages 665–674.

[107] Yang, T., Lin, Q., and Li, Z. (2016). Unified convergence analysis of stochastic momentum methods for convex and non-convex optimization. *arXiv preprint arXiv:1604.03257*.

[108] Yoon, T. and Ryu, E. K. (2021). Accelerated algorithms for smooth convex-concave minimax problems with $\mathcal{O}(1/k^2)$ rate on squared gradient norm. In *International Conference on Machine Learning*, pages 12098–12109.

[109] Zhang, J., O'Donoghue, B., and Boyd, S. (2020). Globally convergent type-I Anderson acceleration for nonsmooth fixed-point iterations. *SIAM Journal on Optimization*, 30(4):3170–3197.

[110] Zuo, G., Huang, S., Li, J., and Gong, D. (2022). Offline reinforcement learning with Anderson acceleration for robotic tasks. *Applied Intelligence*, 52(9):9885–9898.