Université de Montréal

Accelerated Algorithms for Temporal Difference Learning Methods

par

Anushree Rankawat

Département de mathématiques et de statistique
Faculté des arts et des sciences

Mémoire présenté en vue de l'obtention du grade de
Maître ès sciences (M.Sc.)
en Discipline

December 31, 2022

© Anushree Rankawat, 2022

Université de Montréal

Faculté des arts et des sciences

Ce mémoire intitulé

Accelerated Algorithms for Temporal Difference Learning Methods

présenté par

Anushree Rankawat

a été évalué par un jury composé des personnes suivantes :

Ioannis Mitliagkis
(président-rapporteur)

Pierre-Luc Bacon
(directeur de recherche)

Guillaume Lajoie
(membre du jury)

Résumé

L'idée centrale de cette thèse est de comprendre la notion d'accélération dans les algorithmes d'approximation stochastique. Plus précisément, nous tentons de répondre à la question suivante : *Comment l'accélération apparaît-elle naturellement dans les algorithmes d'approximation stochastique ?* Nous adoptons une approche de systèmes dynamiques et proposons de nouvelles méthodes accélérées pour l'apprentissage par différence temporelle (TD) avec approximation de fonction linéaire : Polyak TD(0) et Nesterov TD(0). Contrairement aux travaux antérieurs, nos méthodes ne reposent pas sur une conception des méthodes de TD comme des méthodes de descente de gradient. Nous étudions l'interaction entre l'accélération, la stabilité et la convergence des méthodes accélérées proposées en temps continu. Pour établir la convergence du système dynamique sous-jacent, nous analysons les modèles en temps continu des méthodes d'approximation stochastique accélérées proposées en dérivant la loi de conservation dans un système de coordonnées dilaté. Nous montrons que le système dynamique sous-jacent des algorithmes proposés converge à un taux accéléré de $\mathcal{O}(1/t^2)$. Ce cadre nous fournit également des recommandations pour le choix des paramètres d'amortissement afin d'obtenir ce comportement convergent. Enfin, nous discrétisons ces ODE convergentes en utilisant deux schémas de discrétisation différents, Euler explicite et Euler symplectique, et nous analysons leurs performances sur de petites tâches de prédiction linéaire.

Mots-clés Méthodes des différences temporelles, approximation stochastique, méthodes accélérées, méthodes de quantité de mouvement, apprentissage par renforcement, programmation dynamique approchée, lois de conservation, taux de convergence, apprentissage automatique

Abstract

The central idea of this thesis is to understand the notion of acceleration in stochastic approximation algorithms. Specifically, we attempt to answer the question: *How does acceleration naturally show up in SA algorithms?* We adopt a dynamical systems approach and propose new accelerated methods for temporal difference (TD) learning with linear function approximation: Polyak TD(0) and Nesterov TD(0). In contrast to earlier works, our methods do not rely on viewing TD methods as gradient descent methods. We study the interplay between acceleration, stability, and convergence of the proposed accelerated methods in continuous time. To establish the convergence of the underlying dynamical system, we analyze continuous-time models of the proposed accelerated stochastic approximation methods by deriving the conservation law in a dilated coordinate system. We show that the underlying dynamical system of our proposed algorithms converges at an accelerated rate of $\mathcal{O}(1/t^2)$. This framework also provides us recommendations for the choice of the damping parameters to obtain this convergent behavior. Finally, we discretize these convergent ODEs using two different discretization schemes, explicit Euler, and symplectic Euler, and analyze their performance on small, linear prediction tasks.

Keywords Temporal difference learning, stochastic approximation, accelerated methods, momentum methods, reinforcement learning, approximate dynamic programming, function approximation, conservation laws, convergence rates, machine learning

Contents

List of figures

List of acronyms and abbreviations

MDP	Markov Decision Process
TD	Temporal Difference
RL	Reinforcement Learning
SA	Stochastic Approximation
SGD	Stochastic Gradient Descent
DP	Dynamic Programming
ADP	Approximate Dynamic Programming
ODE	Ordinary Differential Equation
GD	Gradient Descent
AGM	Accelerated Gradient Methods
MC	Monte Carlo

MSE	Mean Square Error
MSBE	Mean Square Bellman Error
MSPBE	Mean Square Projected Bellman Error
RMSPBE	Root Mean Square Projected Bellman Error
GTD	Gradient Temporal Difference
HTD	Hybrid Temporal Difference
TDRC	Temporal Difference with Regularized Corrections

Acknowledgements

I would like to take a moment to thank all the people who were instrumental in making this thesis happen.

First of all, I would extend immense gratitude to my supervisor, Pierre-Luc Bacon. I am eternally indebted to you for believing in me and supporting me throughout this journey, especially at times when this work did not seem to offer much promise.

I would also like to thank my colleagues and friends who have supported me constantly throughout this process, especially David Yu-Tung Hui, Sakshi Shrivastava, and Paul Crouther. I would also like to thank Ryan D'Orazio and Pierluca D'Oro who have helped me better formulate my problem statement.

I would like to thank my parents, Ashutosh and Shalini Rankawat, whose constant encouragement and support have made it possible for me to reach this far. Lastly, I would like to thank my sister, Mansi Rankawat. This thesis would not have been possible without your undying support and constant help throughout the process, from helping me decide on the problem to painstakingly ironing through every detail of the proofs. I am and will always be eternally grateful to you for everything.

Chapter 1

Introduction

Temporal Difference (TD) methods (Sutton [1988]) lie at the heart of Reinforcement Learning (RL) (Sutton and Barto [2018]), which has seen a recent resurgence with the advent and success of Deep RL (Mnih et al. [2015]). However, TD methods are known to be slow to converge (Szepesvári [1997]). This has led to a rise in new TD methods proposed to counter this issue by accelerating the dynamics. Developing accelerated methods has a long history of work in the field of gradient-based optimization for stochastic gradient descent (SGD) (Polyak [1964], Nesterov, Amari [1998]). Inspired by the success of accelerated methods in SGD, a significant effort in accelerating TD methods comes from a subset of algorithms called Gradient TD methods (Sutton et al. [2009], Liu et al. [2016]. This can be attributed to the fact that it is easier to adapt accelerated methods, originally developed in optimization, in the GTD setting instead of the usual 'semi-gradient' TD setting. However, it is well-known that TD is not a gradient descent method (Barnard [1993]) and treating them as one could worsen their performance if acceleration is added naively (Bengio et al. [2021]).

This begs the question: *What does it mean to have acceleration in the RL setting?* Throughout this work, we attempt to answer this question by proposing new accelerated algorithms for temporal difference methods. In this work, instead of the usual approach taken in the RL community, we go back to the roots of how accelerated methods were first introduced using the dynamical systems approach in optimization (Polyak [1964]). We view TD as belonging to a general class of algorithms called Stochastic Approximation (SA) methods and take the Ordinary Differential Equation (ODE) approach for the design and analysis of accelerated methods. This approach also has a long history of usage in the SA community for establishing the stability and convergence of SA methods (Kushner and Yin [2003], Borkar [2009], Benveniste et al. [2012]). The fundamental idea behind the ODE approach used in SA for analysis is that the stability and convergence of the underlying ODEs guarantee the stability and convergence of the associated SA iterates (Borkar and Meyn [2000]). Specifically, under certain assumptions, we can approximate the behaviour of the discrete-time

stochastic systems by that of the underlying continuous-time deterministic systems (Borkar and Meyn [2000]). Also, taking the ODE perspective on accelerated methods in optimization has shown us that these methods are derived from second-order ODEs of a particular form. Most notably, these ODEs are known to belong to the class of mass-spring-damper systems or damped harmonic oscillators (Muehlebach and Jordan [2021], Wibisono et al. [2016]). We take these existing concepts and build upon them to introduce physics-inspired momentum-based TD methods. We also establish an intuitive way of analyzing these SA methods and deriving the convergence rates, drawing from the recent advancements of the ODE perspective in optimization (Wibisono et al. [2016], Diakonikolas and Jordan [2021], Suh et al. [2022]). We design the Polyak and Nesterov's momentum counterparts for acceleration in TD methods. Specifically, we start in the continuous-time domain, designing these ODEs with fast convergent behaviour before discretizing them to give us new algorithms.

A natural question that might arise next is: *How could the dynamical systems approach ultimately help design accelerated methods in the discrete-time domain?* While working in continuous-time introduces the problem of discretizing the continuous-time dynamical system, there are several advantages of adopting this approach to acceleration. Apart from being the original intuition and approach for introducing momentum in optimization (Polyak [1964]), acceleration holds a mathematical meaning as a differential concept in continuous-time. Acceleration also holds a distinctively physical meaning as the rate of change of speed across a curve and poses the problem of finding the *optimal way to optimize* as a physically meaningful question: *what is the fastest rate?* (Jordan [2018]).

Another advantage of choosing the dynamical systems approach is the ability to design convergent ODEs, followed by choosing appropriate discretization schemes to convert this ODE to an implementable discrete-time system. An important question in the field of numerical integration is to understand the quality of the said discrete approximation. In addition to this, discretization schemes for numerical integration have certain approximation guarantees. Various bounds exist for different integration schemes adapted to the structure of the underlying ODE (Wanner and Hairer [1996]). This allows for more control over the algorithm design and the ability to choose a discretization scheme that better suits the structure of our given ODE.

Preserving the convergent nature, stability and accelerated rates of these ODEs in the discrete-time system is a non-trivial task. Certain integrators are better suited to preserve the structure of a particular ODE than others (Shi et al. [2019]). Although the primary focus of our work is on building theoretical foundations for accelerated temporal difference methods, we also study the impact of different discretization schemes on translating the accelerated rates obtained from the ODEs to the discrete-time system. We perform explicit Euler and symplectic Euler discretization on our proposed ODEs and study their behaviour

on small, linear prediction tasks. Initial results for these accelerated methods have been reported and they show comparable performance to the existing TD methods.

A significant contribution of our work is that we extend the energy conservation framework of Suh et al. [2022] defined for accelerated gradient descent methods (AGM) to the semi-gradient TD setting. An advantage of using this approach is that instead of guessing the Lyapunov function needed to perform the analysis of such methods (Wilson et al. [2021]), which is cumbersome and often requires intensive trial-and-error, we obtain one directly through the energy conservation law associated with our accelerated TD method. We theoretically show accelerated rates for the ODE associated with TD(0) with linear function approximation. Rate derivations for algorithms are known to be very complex; this methodology, on the other hand, is much more intuitive.

The energy conservation methodology relies on a so-called dilated coordinate system to derive an energy term and arrive at the convergence rate of the continuous-time system. However, we can also choose a different dilated coordinate system and derive an energy term that can help us relate it to energy conservation of physical systems and understand what constitutes kinetic and potential energy for our accelerated temporal difference methods. We can therefore discover conserved properties of the underlying dynamical system along with meaningful interpretations and use them to understand our discrete algorithm using physics analogies.

1.1. Thesis Statement

This work aims to give a framework to intuitively arrive at the 'right' momentum counterparts for Polyak's Heavy Ball and Nesterov's acceleration in TD methods. We also introduce the energy conservation law-based methodology in SA methods for establishing convergence. To the best of our knowledge, this is the first work that extends and develops such energy conservation law-based framework for analysis in RL. We use this framework to derive an accelerated rate of $\mathcal{O}(1/t^2)$ for the underlying dynamical system of the proposed accelerated methods and analytically obtain the parameters needed to get this rate. To this end, we cast the TD learning problem as an SA method and derive the convergence rates of these accelerated methods in the continuous-time system. This is done by viewing the TD update as a discretization of the underlying ODE. Lastly, we study the impact of discretization schemes on how well these rates are preserved after discretization using empirical performance of our proposed methods on small, linear prediction tasks. We hope this work sheds some light on acceleration in TD methods and reduces the gap between analysis frameworks in RL and those in gradient-based optimization.

Chapter 2

Background and Related Works

In this chapter, we give the technical background and introduce key concepts, like dynamic programming, approximate dynamic programming and stochastic approximation. We explain the ODE approach used in SA for analysis. We also touch up on some concepts about discretization schemes and the performance metrics which will be used for evaluation of our proposed accelerated methods. Lastly, we give an in-depth discussion of accelerated methods in the optimization community as well as the RL community. A broad understanding of these concepts is essential to dive deeper into the world of accelerated temporal difference learning methods that are the focus of our study.

2.1. Dynamic Programming and Reinforcement Learning

In mathematical optimization, Dynamic Programming (DP) techniques often come into play in the setting where decisions can be made in stages, or to put it simply, a decision can be broken down into a series of steps taken over. Typically, the objective is to minimize a given cost, or equivalently, maximize some reward. Unlike supervised learning where the training targets are provided, we are working in the setting where optimal (long-term) actions need to be inferred from a weaker form of supervisatory signal: the immediate reward function. While the outcome of the decision cannot be accurately stated, we can estimate the impact of the decision to a certain extent before the next decision is made. With discrete time steps, for every time step, we can take a discrete decision with a reward assigned for every step of the process. Our goal in this scenario is to maximize the total reward accumulated at the end: a quantity called the ‘return‘.

Reinforcement learning methods (especially model-free ones) come to the rescue for scenarios where a model of the system is infeasible to be obtained, possibly because the system is not fully known beforehand, not well understood or costly to construct.

As we can see, DP and RL primarily work under the discrete-time assumption. Note that our use of the ODE approach doesn't mean that we consider a continuous-time model of interaction; instead we consider a continuous-time evolution of the parameters of our model, not the actions.

In essence, dynamic programming (DP) is about leveraging the structure under the MDP assumption to derive efficient algorithms for finding optimal policies. It provides a solution for the curse of dimensionality that we encounter frequently in the RL community. Without it, the space of all possible policies becomes too large to search directly.

2.1.1. Elements of the learning problem

A controller (agent) interacts with the process (environment) using three signals: a state signal, which describes the state of the process, an action signal, which describes the action to be taken by the agent in a given state and a scalar reward signal, which gives the controller feedback to measure immediate performance. At each discrete time step, the controller gets a state signal which it uses to perform an action to transition into another state. A reward signal is given to the controller which allows it to evaluate the quality of performance.

The behaviour of this controller is dictated by its policy. A deterministic stationary policy is a mapping from states into actions outlining what action should be taken for every state the controller could exist in. The behaviour of the process is determined by the underlying dynamics, which describes how the state changes based on the controller's actions. Note that we obtain an MDP only when the underlying dynamics, specified by the transition matrix, are Markovian. The process dynamics and reward function, along with the state space and action space (set of all possible states and actions, respectively) constitute a Markov Decision Process (MDP). Note that we work under the assumption of a finite state space, allowing us to circumvent the need to work with Harris Chains.

Till now, we have used terms like process, state, action and reward without formally defining them. We will now formally define these terms and a Markov Decision Process.

2.1.2. Markov Decision Process

A finite discounted MDP is characterized by the following tuple: $(\mathcal{S}, \mathcal{A}, P, \mathcal{R}, \gamma)$ where $\mathcal{S} \in \mathbb{R}^{|S|}$ is the set of all possible states (state space), $\mathcal{A} \in \mathbb{R}^{|A|}$ is the set of all possible actions (action space), $\mathcal{T} \in \mathbb{R}^{|S|\times|A|\times|S|}$ is the transition probability, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function and $\gamma \in [0,1]$ is the discount factor. We assume the actions and states to be continuous, which allows us to write the transition dynamics as the following probability distribution: $P(S_{t+1}|S_t, A_t)$.

A Markovian stationary randomized policy $\pi(a_t|s_t)$ is a distribution over actions conditioned the states and determines the action to be taken in a given state.

In an MDP, at every time step $t \in 1,2,\cdots,T$, the controller takes an action and transitions from state s_t to state S_{t+1} determined by the transition probability $p(S_{t+1}|S_t, A_t)$. Upon this transition, the controller receives a reward R_t from the process. The expected reward can be defined as $R(S_t,A_t) = \mathbb{E}[R_t|S_t, A_t] = \sum_{r\in\mathbb{R}} r \sum_{s'} P(s', r|S_t, A_t)$ where $R : S \times A \to \mathbb{R}$.

2.1.3. Value Functions and Bellman Equations

The goal of the controller is to maximize the expected cumulative reward received in the long run. We define this cumulative reward over all time steps to be maximized as the expected return. If the sequence of rewards received after time step t is denoted as $R_t, R_{t+1}, \cdots$, we define the discounted return G_t as

$$G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \cdots = \sum_{k=0}^{T} \gamma^k R_{t+k} \ ,$$

where the discounted factor γ helps determine how to assign importance to the immediate reward and future rewards to avoid getting an infinite reward in continuous tasks.

Policies are often characterized using their value functions. A value function is tasked with estimating how good it is for an agent to be in any given state. We have two types of value functions: the state-value function, V-function, and the state-action value function, Q-function.

For a policy π, the value of a state s is given by $v_\pi(s) : S \to \mathbb{R}$ which is the expected return from a state s following policy π. For MDPs, we can formally define $v_\pi(s)$ as

$$v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k} \middle| S_t = s \right] \ , \tag{2.1.1}$$

where $\mathbb{E}_\pi[.]$ denotes the expected value of a random variable given policy π. This function v_π is called the state-value function for policy π.

Similarly, we denote the state-action value function as the value of taking action a when in the state s under policy π as $q_\pi(s,a) : S \times A \to \mathbb{R}$. It is defined as the expected return of the agent in state s, taking action a and following policy π thereafter.

$$q_\pi(s,a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k} \middle| S_t = s, A_t = a \right] \tag{2.1.2}$$

For a given policy π and state s, the following recursive relationship holds between the value of the current state s and the value of its successor states:

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k} \middle| S_t = s \right] \\ &= \mathbb{E}_\pi \left[R_t + \gamma \sum_{k=0}^{\infty} \gamma^{k-1} R_{t+k-1} \middle| S_t = s \right] \end{aligned}$$

$$v_\pi(s) = \mathbb{E}_\pi[R_t + \gamma v_\pi(S_{t+1})|S_t = s] \ . \tag{2.1.3}$$

Similarly, for a given policy π, state s, and action a, we can define the recursive relationship between the value of the current state s and the value of its possible successor states for the state-action value function as:

$$q_\pi(s,a) = \mathbb{E}_\pi[R_t + \gamma q_\pi(S_{t+1},A_{t+1})|S_t = s, A_t = a] \ . \tag{2.1.4}$$

The recursive expressions in (2.1.3) and (2.1.4) are called "Bellman equations" in the reinforcement learning community. They provide a relationship between the value of a state and the values of its successor states.

The Bellman equations provide a representation of v_π as a fixed point equation for solving v_π. For the case of a deterministic policy, $\pi \in \Pi$ and any given state $s \in \mathcal{S}$, we can also express the Bellman equation as:

$$\begin{aligned} v_\pi(s) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k}\middle|S_t = s\right] &= R_t + \mathbb{E}_\pi\left[+\gamma\sum_{k=0}^{\infty}\gamma^{k-1}R_{t+k-1}\middle|S_t = s\right] \\ &= R_t + \mathbb{E}_{T^\pi}\left[\mathbb{E}_\pi\left[\gamma\sum_{k=0}^{\infty}\gamma^{k-1}R_{t+k-1}\middle|S_t = s, S_{t+1} = s'\right]\middle|S_t = s\right] \\ &= R_t + \gamma\int_{\mathcal{S}} T^\pi(s'|s)\mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^{k-1}R_{t+k-1}\middle|S_{t+1} = s'\right]ds \\ &= R_t + \gamma\int_{\mathcal{S}} T^\pi(s'|s)v_\pi(s')ds' \ . \end{aligned} \tag{2.1.5}$$

Let us define the Bellman operator as $\mathcal{T}^\pi : \mathbb{R}^{\mathcal{S}} \to \mathbb{E}^{\mathcal{S}}$. For all $s \in \mathcal{S}$, we can write:

$$(\mathcal{T}v_\pi)(s) = R_t + \gamma\int_{\mathcal{S}} T^\pi(s'|s)v(s')ds' \ .$$

Therefore, we can now rewrite (2.1.5) in a compact from as the following fixed point equation:

$$v_\pi = \mathcal{T}^\pi v_\pi \ , \tag{2.1.6}$$

where $\mathcal{T}^\pi$ is a linear affine operator.

2.1.4. Bellman Optimality Equation

The goal of any reinforcement learning task is to find an optimal policy from data. Policies are assigned orderings using the aforementioned value functions. A policy π is said to be better than or equal to a policy π' if its expected return is greater than or equal to that of π' for all states. Mathematically, we can say $\pi \geq \pi'$ if and only if $v_\pi(s) \geq v_{\pi'}(s)$ for all $s \in \mathcal{S}$. An optimal policy $\pi_\star$ is defined as the policy which is better than or equal to all

other policies. Optimal policies share the same state-value function, $v^\star$ defined as:

$$v^\star(s) = \max_\pi v_\pi(s) \qquad \forall \quad s \in \mathcal{S} \ . \tag{2.1.7}$$

Optimal policies also share the same optimal state-action value function, $q^\star$ defined as:

$$q^\star(s,a) = \max_\pi q_\pi(s,a) \qquad \forall \quad s \in \mathcal{S} \quad ; \quad a \in \mathcal{A} \ . \tag{2.1.8}$$

We can therefore write $q^\star$ in terms of $v^\star$ as follows:

$$q^\star(s,a) = \mathbb{E}[R_t + \gamma v^\star(S_{t+1}) \mid S_t = s, A_t = a] \ . \tag{2.1.9}$$

Since $v^\star$ is a value function for a given policy, we can write the Bellman equation for state values given in (2.1.3) to get the Bellman optimality equation.

$$\begin{aligned} v^\star(s) &= \max_{a \in \mathcal{A}(s)} q_{\pi^\star}(s,a) \\ &= \max_a \mathbb{E}_{\pi^\star}\left[R_t + \gamma \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s, A_t = a\right] \\ &= \max_a \mathbb{E}\left[R_t + \gamma v^\star(S_{t+1}) \mid S_t = s, A_t = a\right] \end{aligned} \tag{2.1.10}$$

This equation states that the value of a state under an optimal policy equals the expected return for the best action from the state. The Bellman optimality equation for $q_\star$ is

$$q^\star(s,a) = \mathbb{E}[R_{t+1} + \gamma \max_{a'} q^\star(S_{t+1}, a') \mid S_t = s, A_t = a] \ . \tag{2.1.11}$$

Note that the optimal value function is the unique fixed point of the following fixed point equation:

$$v^\star(s) = \max_{a \in \mathcal{A}} \left[R_t + \gamma \int_{\mathcal{S}} T(s'|s,a) v^\star(s') ds'\right] \ . \tag{2.1.12}$$

Let $T^\star : \mathbb{R}^{\mathcal{S}} \to \mathbb{R}^{\mathcal{S}}$ be the Bellman optimality operator given by:

$$(\mathcal{T}^\star v)(s) = \max_{a \in \mathcal{A}} \left[R_t + \gamma \int_{\mathcal{S}} T(s'|s,a) v(s') ds'\right] \ .$$

The Bellman optimality equation can also be succinctly expressed as the following fixed point equation:

$$\mathcal{T}^\star v^\star = v^\star \ . \tag{2.1.13}$$

Note that given an optimal value function, $v^\star$ or $q^\star$, we can derive an optimal policy through the greedy (also called v-improving) policy (Puterman [2014]).

2.1.5. Approximate Dynamic Programming (ADP)

The classical dynamic programming (DP) algorithm requires exact representations of the value functions and the policies. An exact value function representation can only be

achieved by storing distinct estimates of the expected return for every state (for state-value function) or for every state-action pair (for state-action value functions). Similarly, to represent the policies exactly, distinct actions for every state have to be stored. However, such exact representations are infeasible for very large or continuous spaces. Therefore, the representations of these value functions and policies need to be approximated.

We will now formally define the problem of dynamic programming. Consider the Bellman equation given by

$$v_t(s) = \max_a \mathbb{E}[R_t + \gamma \bar{v}(s) | s,a] \ . \tag{2.1.14}$$

where the expectation is taken over the stationary distribution induced by running the given policy π, denoted by $X_d \in \mathbb{R}^{|S|}$. Note that since the computation of $v_t(s)$ requires $\bar{v}(s)$ for each $s \in \mathcal{S}$ which can be described as stepping backward through time, this is known as backward dynamic programming.

If S_t is a discrete, scalar variable, enumerating the states may be feasible for small state spaces; but if it is a vector, then the number of states grows exponentially with the number of dimensions. In the case of continuous s, even if it is scalar, enumerating the states is simply not possible. ADP addresses this issue by replacing the true value function $v(s)$ with an approximation which we will denote as $\bar{v}(s)$. This is known as value function approximation, allowing methods to search for policies in a restricted space of value functions. We can deal with multidimensional variables by simply treating the vector s to be a continuous function of some feature vector denoted by ϕ. Note that ϕ is a mapping of the states. A simple approximation is to assume the value function to have a linear relationship with respect to the feature vector ϕ:

$$\bar{v}(S; w) = \phi(s)^T w \ , \tag{2.1.15}$$

where $w \in$ is the weight vector to be learned. Choosing an appropriate value function approximation scheme is central to the idea of ADP. The task of designing a good approximate value function consists in choosing a sufficiently rich parametrization while keeping the computational resources under control.

In this setting, the policy is determined by the value function approximation $\bar{v}_{t+1}(S_{t+1})$ where the policy space Π is the set of all possible value function approximations.

2.2. Policy Evaluation

The goal of policy evaluation algorithms is to accurately estimate the value functions, $v_\pi(s)$ and $q_\pi(s,a)$ for a given policy π. We consider the case of on-policy algorithms where the policy is stationary.

2.2.1. Monte Carlo Methods

The simplest way to learn the value function is using Monte Carlo (MC) methods to estimate the value from experience, specifically averaging the returns observed by the agent after visits to the state. As we observe more returns, we expect the average to converge to the expected value.

We formally define a visit to state s as every occurrence of state s in a given episode. We have two ways of estimating the value function: the every-visit MC method and the first-visit MC method. While the every-visit MC method estimates $v_\pi(s)$ as the average of returns after all the visits to state s in a given set of experiments, the first-visit MC method averages the returns following first visits to state s. We can state the every-visit Monte Carlo method as:

$$v(S_t) = v(S_t) + \alpha[G_t - v(S_t)] \ , \tag{2.2.1}$$

where G_t is the return following time t and α is a step-size parameter. Note that this expression is simply an online average. It is useful as it provides an intuition for SA methods. Rather than averaging a bunch of samples at every step, we can 'average through time'.

Monte Carlo methods are only defined for episodic tasks since the value estimates are only updated once an entire episode is run. Though it might seem that Monte Carlo methods are a good way to estimate the value function, they have a lot of drawbacks. While the updates generated by the Monte Carlo algorithm in the tabular setting are not biased since we are approximating an average value of the value function by averaging all the observed values, however, since we wait until the entire episode is completed to update our estimate, a high amount of variance is introduced in the update owing to the existing randomness present in the environment explored during one run of the episode. In addition to this, we cannot change the estimate of our value function while we are interacting with our environment.

2.2.2. Temporal Difference Methods

Temporal Difference learning methods differ from Monte Carlo methods in that they learn from the difference between temporally successive estimates of the same quantity. Temporal Difference (TD) learning is a central idea of reinforcement learning which can be intuitively explained with a combination of ideas from Monte Carlo (MC) and Dynamic Programming (DP). TD methods can learn directly from experience without needing a model of the environment like Monte Carlo methods and it can update estimates using the previously learned estimates without waiting for the full episode to run (i.e., bootstrapping). This allows for learning in an online fashion.

Specifically, TD methods adjust the current value of the earlier state to be closer to the observed value of the later state. We do this by moving the estimate of the earlier state's value function a fraction of the way closer to the value of the later state in accordance with

the established recursive relationship. Let $S_t = s$ denote the state before the greedy move and $S_{t+1} = s^{'}$ denote the state after the move. We can then describe the update to $V(s)$ as:

$$V(s) \leftarrow V(s) + \alpha[V(s^{'}) - V(s)] \ ,$$

where α is the fraction that determines the influence of the update on the existing value of $V(s)$ This update rule is an example of a temporal difference learning method, defined as because the changes are based on a difference, $V(s^{'}) - V(s)$, between estimates taken at two different time steps.

- **The TD(0) algorithm**
  Consider the total-return problem given by Equation (2.1.3). We also call this the fixed-policy Value Iteration Algorithm which can be written as:

$$v_{t+1}(s) = \mathbb{E}_\pi[R_t + \gamma v_t(S_{t+1} = s^{'})|S_t = s]$$

$$= R_t(s_t) + \gamma \sum_{s'} P_\pi(S_{t+1} = s'|S_t = s) v_t(S_{t+1} = s^{'}), \quad s \in \mathcal{S} \ , \tag{2.2.2}$$

  or simply $v_{t+1} = R_\pi + \gamma P_\pi v_t$. Here, we assume that we do not have the exact values of R_π and P_π and are learning these quantities with each iteration.
  Let $\hat{v}$ be our estimate of v_π. We can say that $[R_t + \gamma \hat{v}(s_{t+1})]$ where s_t, s_{t+1} and R_t are observed at time t, is an unbiased estimate of the right-hand side of (2.2.2) such that:

$$\mathbb{E}_\pi[R_t + \gamma \hat{v}_t(s_{t+1})|s_t] = R_t(s_t) + \gamma \sum_{s'} P_\pi(s'|s) v_t(s^{'}) \ .$$

  However, since there is an introduction of noise in this estimate owing to the randomness in R_π and P_π, we decide to modify $\hat{v}$ by a small fraction in the direction of this quantity, given by:

$$\hat{v}^{(t)}(s_t) := (1 - \alpha_t)\hat{v}^{(t)}(s_t) + \alpha_t[R_t + \gamma \hat{v}^{(t)}(s_{t+1})]$$

$$= \hat{v}^{(t)}(s_t) + \alpha_t[R_t + \gamma \hat{v}^{(t)}(s_{t+1}) - \hat{v}^{(t)}(s_t)] \ , \tag{2.2.3}$$

  where α is the gain of the algorithm. This is called the TD(0) algorithm.
- **SARSA: On-Policy TD Control**
  Here, we are interested in using TD methods for the on-policy control problem. We start with learning a state-action value function rather than the TD(0) algorithm where we focused on learning the state value function. Specifically, we estimate $q_\pi(s,a)$ for the given policy π and for all given states $s \in \mathcal{A}$ and actions $a \in \mathcal{A}$. In each episode, we are in state s, having taken an action a given by the ϵ-greedy policy π to get to another state $s^{'}$, while observing reward r.

Here, we learn a value function estimating the value of state-action pairs by considering the transitions from state-action pair to state-action pair. The update rule for the state-action value function follows (2.2.2) and is done by choosing another action $a\prime$ following the same current policy.

$$q_\pi(s,a) = q_\pi(s,a) + \alpha[r + \gamma q_\pi(s^{'},a^{'}) - q_\pi(s,a)] \tag{2.2.4}$$

Since the update rule relies on the values of the quintuple $(S_t, A_t, R_t, S_{t+1}, A_{t+1})$, it is called the SARSA algorithm for TD control. We call this algorithm on-policy learning as the new action $a\prime$ is chosen using the same ϵ-greedy policy as the one used for taking action a.

- **Q-Learning: Off-Policy TD Control**

  While SARSA is an on-policy algorithm and hence follows the given policy π throughout, Q-learning attempts to directly approximate the optimal action function, $q_\star$ independent of the ϵ-greedy policy π being followed when learning the state-action value function (or Q-value function). However, the policy comes into play by governing which state-action pairs are visited and updated. In both cases, the policy followed by the agent is taken to be ϵ greedy.

  Q-learning uses different policies for choosing the next action a^g in a greedy fashion, i.e., choosing the action that maximizes the Q-value function at the new state value, $q(s^{'}, a)$. This can be written as:

$$q^{(t)}(s,a) = q^{(t)}(s,a) + \alpha[r + \gamma q^{(t)}(s^{'}, a^g) - q^{(t)}(s,a)]$$

$$= q^{(t)}(s,a) + \alpha[r + \gamma \max_u q^{(t)}(s^{'}, u) - q^{(t)}(s,a)] \ .$$

  As we can see, Q-learning is an off-policy algorithm because the new action a^g is chosen greedily and not using the current policy π.

  Following this update, we go back to starting from the new state $s^{'}$ till we finally reach the end state. Note that the next actions in SARSA and Q-learning, $a^{'}$ and a^g are not actually taken by the agent but are only chosen to update the Q-value function. The actual action taken by the agent a for both SARSA and Q-Learning is taken using the given ϵ-greedy policy.

  Q-Learning algorithms are known to converge to the optimal policy q_*.

- **n-step TD methods**

  The success of TD methods can be attributed to the idea of bootstrapping; where we update the estimate of the value function in one state using the estimate of the value function in the next state. In other terms, $v(S_t)$ is updated using $v_t(S_{t+1})$. These methods often perform more efficiently than averaged sample returns.

However, there could be instances where $v_t(S_{t+1})$ may not be accurate. This could be the case when the rewards are sparse or when our initial estimates of $v(S_{t+1})$ are poor. In such situations, relying solely on this value for updating $v_{t+1}(S_t)$ would not be the best possible learning strategy. Such cases might benefit from using Monte Carlo methods to add robustness to the value function updates.

Since we would still like to leverage the advantages that come with TD methods in the ability to make updates before the end of episodes, this requires the need for methods that lie at the intersection of Monte Carlo methods and one-step TD methods.

While one-step TD methods allow for quicker updates to the value function estimate, bootstrapping works best when done over a length of time where a significant change in the state has occurred. We, therefore, extend the existing definition of temporal methods to allow for multi-step methods enabling us to bootstrap over multiple time steps.

TD(n) algorithms combine both methods by performing an update using an intermediate number of rewards that is more than 1 (one-step TD methods) but less than all the rewards until the termination of the episode (MC methods). Specifically, we modify the TD target to slightly resemble the Monte Carlo target. Since these algorithms still rely on later estimates to update the current estimate, they fall under the category of temporal difference methods.

Consider our problem of estimating the value function v_π from episodes generated by policy π. For the TD(0) update given by (2.2.2), we used a 1-step return, denoted by $G_t^{(1)} = R_t + \gamma v_t(S_{t+1})$ where $v_t(S_{t+1})$ acts as a stand-in for the rest of terms of the total return: $\gamma R_{t+1} + \gamma^2 R_{t+2} + \cdots$.

Instead of using this 1-step return, we now define an n-step return as:

$$G_t^{(n)} = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \cdots + \gamma^{n-1} R_{t+n-1} + \gamma^n v_{t+n-1}(S_{t+n}) \ .$$

Similar to the TD(0) update equation, we can now define the TD(n) update to be:

$$v_{t+n}(s) := v_{t+n-1}(s) + \alpha_t [G_t^{(n)} - v_{t+n-1}(s)] \tag{2.2.5}$$

$$= v_{t+n-1}(s) + \alpha_t [R_t + \cdots + \gamma^{n-1} R_{t+n-1} + \gamma^n v_{t+n-1}(S_{t+n}) - v_{t+n-1}(s)] \ .$$

We know that temporal difference methods use rewards obtained in the future. Recall that in the TD(0) algorithm, we are required to wait for one step, and observe the reward and next state before we can make an update to the previous state's value function. As n-step methods use rewards obtained from the upcoming n steps, we have to wait for n steps before we can make any updates to the value function $v_\pi(s)$.

2.3. Stochastic Approximation

Stochastic Approximation was first introduced by Robbins and Monro [1951] for solving stochastic root-finding problems from noisy observations. Stochastic approximation algorithms are iterative procedures that make small changes in order to improve a certain performance criterion based on the feedback received from the environment. They are used to solve optimization problems and fixed point equations where the direct observations are corrupted by noise.

Consider the nonlinear equation $h(x) = 0$ given noisy measurements of the function h. To put it simply, we do not have access to the mathematical model h, instead, we are given a black box such that given an input x, it returns a value $y = h(x) + \xi$ where ξ is a zero mean noise random variable. Let x^* be the solution to this equation such that $h(x^*) = 0$. The stochastic approximation scheme given by Robbins and Monro Robbins and Monro [1951] is given by:

$$x_{n+1} = x_n + a(n)[h(x_n) + M_{n+1}] \ . \tag{2.3.1}$$

Here, $\{M_n\}$ is the noise sequence and $\{a(n)\}$ are positive scalars such that:

$$\sum_n a(n) = \infty \quad \text{and} \quad \sum_n a(n)^2 < \infty \ . \tag{2.3.2}$$

The conditions on the discrete-time steps $\{a(n)\}$ given by (2.3.2) ensure that the noisy time-discretization covers the entire time axis (retaining $a(n) \to 0$) while also having the error due to the noise to be asymptotically negligible, suppressing the noise variance. Note that $h(n)$ and M_{n+1} are only available as a sum. It is also assumed that $\{M_n\}$ is a martingale difference sequence (Borkar and Meyn [2000]), i.e., a sequence of integrable random variables satisfying

$$\mathbb{E}[M_{n+1}|x_m, M_m, m \leq n] = 0 \ . \tag{2.3.3}$$

Note that if $h(x)$ is continuously differentiable in x and $\nabla_x h(x)$ denotes its gradient w.r.t. x, we end up with the stochastic gradient method:

$$x_{n+1} = x_n + a(n)[-\nabla_x h(x_n) + M_{n+1}] \ . \tag{2.3.4}$$

2.3.1. ODE approach to the analysis of SA algorithms

A popular approach for the theoretical analysis of these SA algorithms is using the so-called ODE method where we view the SA scheme given by (2.3.1) as a noisy time-discretization of a limiting ODE. We will now formally analyze the stochastic approximation method in $\mathbb{R}^d$ given by

$$x_{n+1} = x_n + a(n)[h(x(n)) + M_{n+1}], \quad n \geq 0 \tag{2.3.5}$$

with a given initial x_0. The following assumptions are made to perform basic convergence analysis(Borkar and Meyn [2000]):

(1) The map $h : \mathbb{R}^d \to \mathbb{R}^d$ is Lipschitz, i.e., there exists $L > 0$ such that:

$$||h(x) - h(y)|| \leq L||x - y||, \quad \forall x,y \in \mathbb{R}^d \ .$$

(2) Stepsizes $a(n)$ are positive scalars satisfying

$$\sum_n a(n) = \infty, \quad \sum_n a(n)^2 < \infty \ .$$

(3) $\{M_n\}$ is a martingale difference sequence with respect to increasing family of σ-fields

$$\mathcal{F} = \sigma(x_m, M_m, m \leq n) = \sigma(x_0, M_1, \cdots, M_n), \quad n \geq 0$$

This gives us $\mathbb{E}[M_{n+1}|\mathcal{F}_n] = 0 \quad a.s., \quad n \geq 0.$

We also have $\{M_n\}$ to be square-integrable with

$$\mathbb{E}[||M_{n+1}||^2|\mathcal{F}_n] \leq K(1 + ||x_n||^2) \quad a.s., \quad n \geq 0$$

for some constant $K > 0$.

(4) The iterates of (2.3.5) remain bounded a.s., i.e.,

$$\sup_n ||x_n|| < \infty, \quad a.s. \ . \tag{2.3.6}$$

Assumption 4 is not a given for the iterates and is usually known to be hard to establish. For general unbounded spaces, this is a stability assumption that must be proved separately using other methods, like stochastic Lyapunov functions (Kushner and Yin [2003]). Under the given stability assumption, the SA iterates given by (2.3.5) are said to asymptotically track the ODE:

$$\dot{x}(t) = h(x(t)), \quad t \geq 0 \ . \tag{2.3.7}$$

We will now discuss how the given assumptions help us arrive at the given limiting ODE. Assumption 1 ensures that the ODE is well-posed, i.e., it has a unique solution for any $x(0)$ that depends continuously on $x(0)$. To formally describe the idea that stochastic approximation asymptotically tracks the trajectories of the ODE, we first define time instants $t(0) = 0, t(n) = \sum_{m=0}^{n-1} a(m), n \geq 1$. The approach taken to show this requires us to construct a continuous interpolated trajectory $\bar{x}(t), t \geq 0$, i.e., a continuous trajectory $\bar{x}(t)$ interpolating the iterates $\{x_n\}$ at times $\{t_n\}$, and show that this trajectory almost surely approaches the solution set of the ODE (2.3.7) asymptotically. In other words, $\bar{x}(t_n) = x_n, n \geq 0$ and $\bar{x}$ is piecewise linear, defined on the intervals $[t_n, t_{n+1}]$. Note that the assumption $\sum_{m=0}^{\infty} \gamma_m$ is required to cover the whole time-axis and in turn, track the ODE asymptotically. We define $x^s(t), t \geq s$ to denote the unique solution to ODE (2.3.7) with $x^s(s) = \bar{x}(s)$. We will now state the general results that follow from the above defined setting, proofs of which can be found in Borkar [2009].

Lemma 2.3.1. *For any* $T > 0$,

$$\lim_{s\to\infty} \sup_{t\in[s,s+T]} ||\bar{x}(t) - x^s(t)|| = 0, \quad a.s. \ .$$

This lemma states that as $s \to \infty$, the interpolated trajectory $\bar{x}$ starting from s remains close to a trajectory of the ODE. In other words, we can say that the noise becomes asymptotically too weak to push the iterates away from the trajectories of the ODE which describe the typical behavior of the system.

For a formal discussion of the convergence of stochastic approximation, we first need to define the concept of chain transitivity. A closed set $A \subset \mathbb{R}^d$ is said to be an invariant set for the given ODE (2.3.7) if any trajectory $x(t), -\infty < t < +\infty$ with $x(0) \in A$ satisfies $x(t) \in A$ for all $t \in \mathbb{R}$. A close set $A \subset \mathbb{R}^d$ is an internal chain transitive set for the ODE if for any x,$y \in A$ and any $\epsilon > 0, T > 0$, there exists points $x_0 = x, x_1, \cdots, x_{n-1}, x_n = y$ in A, for some $n \geq 1$, such that the trajectory of ODE (2.3.7), starting at x_i, for any $0 \leq i < n$ meets with the ϵ-neighbourhood of x_{i+1} after a time greater than or equal to T. These small jumps at the points of the chain are a natural assumption for stochastic approximations, where the noise pushes the iterates away from the trajectories of the ODE.

We can now state the convergence theorem for stochastic approximation as

Theorem 2.3.2 (Benaïm [1999])**.** *Let us assume that assumptions (1)-(4) are satisfied. Then, almost surely, the sequence* $\{x_n\}$ *generated by* (2.3.5) *converges to a (possibly time-dependent) compact connected internally chain transitive invariant set of* (2.3.7).

This ODE approach is widely employed for establishing the convergence of stochastic approximation methods. In addition to this, the ODE approach allows us to understand the average or typical behavior of the algorithm. However, the biggest advantage of taking this approach is that it is often used to create new discrete stochastic approximation algorithms from convergent ODEs (Devraj and Meyn [2017], Meyn [2022]).

2.3.2. ADP and connection with Stochastic Approximation

Approximate dynamic programming is a powerful strategy to address the curse of dimensionality in dynamic programming, with TD being one of the most popular ADP methods. TD was first studied under the lens of SA by Tsitsiklis and Van Roy [1996] who established convergence using an operator-theoretic variant of the ODE method. We know that stochastic approximation is an algorithm to find the roots of a function with noisy observations. If TD can be cast as an SA algorithm, the question that arises is: *What is the function we are trying to find the roots of? Why is this function noisy?*

The root finding problem for the TD method happens to be the fixed point problem associated with the projected Bellman equation. The projected Bellman equation is a variant

of the Bellman equation given by:

$$\hat{v} = \Pi \mathcal{T} \hat{v} \ , \tag{2.3.8}$$

where $\hat{v} \in \text{span}(\Phi)$ and Π is a projection operator with $\text{im}(\Pi) = \text{span}(\Phi)$. Here, $\text{im}(\Pi)$ is the image of Π or more specifically the set $\text{im}(\Pi) := \{\Pi x : x \in \mathcal{H}\}$. We would now show how this projected Bellman operator can be expressed as a stochastic approximation method, which plays a fundamental role in all of our derivations ahead. Note that much of the proof sketch below follows Yang-Zhao et al. [2018]'s discussion of the projected Bellman equations. As a recap, for any $v \in \mathbb{R}^{\mathcal{S}}$, the Bellman operator is given by:

$$(\mathcal{T} v)(s) = R_t + \gamma \int_{\mathcal{S}} T(s'|s) v(s') ds' \ .$$

Let P_T be an operator defined as

$$P_T f(s) = \int_{\mathcal{S}} T(s'|s) f(s') ds' \ .$$

We can therefore rewrite the Bellman operator in terms of P_T as

$$\mathcal{T} v(s) := R_t + \gamma P_T v(s) \ .$$

Note that we would consider the scenario where we have applied linear funtion approximation to our TD method. We can therefore write $\hat{v} \in \text{im}(\Phi)$ as $\hat{v}(.) = \phi^T(.) w$. We can therefore define P_T as:

$$P_T \phi^T(s) := [P_T \phi_1(s) \cdots P_T \phi_k(s)] \in \mathbb{R}^k \ .$$

Let Π_ψ be a natural projection operator characterized by (ϕ, ψ). Let $\hat{v} \in \text{im}(\Pi_\psi)$ be given as $\hat{v} = \sum_{i=1}^k \phi_i w_i$. Note that since Π_ψ projects orthogonally to $\text{im}(\Pi_\psi^\star)$ and onto $\text{im}(\Pi_\psi)$, we have $\mathcal{T}\hat{v} - \Pi_\psi \mathcal{T} \hat{v} \perp \ker(\Pi_\psi)$. We can therefore say that:

$$\langle \psi_i, \mathcal{T}\hat{v} - \Pi_\psi \mathcal{T} \hat{v} \rangle = 0 \ .$$

Substituting the projected Bellman equation, we get:

$$\begin{aligned}
0 &= \langle \psi_i, \mathcal{T}\hat{v} - \hat{v} \rangle \\
&= \langle \psi_i, \Big[R + \gamma P_T \phi^T(.) w - \phi^T(.) w \Big] \rangle \\
&= \langle \psi_i, R \rangle + \Big\langle \psi_i, \gamma \sum_{j=0}^k P_T \phi_j w_j \Big\rangle - \Big\langle \psi_i, \sum_{j=0}^k \phi_j w_j \Big\rangle \\
&= \langle \psi_i, R \rangle - \sum_{j=0}^k \Big\langle \psi_i, (\phi_i - \gamma P_T \phi_j) w_j \Big\rangle \\
&= \langle \psi_i, R \rangle - \sum_{j=0}^k \Big\langle \psi_i, (I - \gamma P_T) \phi_j w_j \Big\rangle \ .
\end{aligned}$$

We can rearrange the terms to write them as:

$$\langle \psi_i, R \rangle - \sum_{j=0}^{k} \Big\langle \psi_i, (I - \gamma P_T)\phi_j w_j \Big\rangle = 0 \ . \tag{2.3.9}$$

Let us define $A \in \mathbb{R}^{k \times k}$ and $\mathbf{b} \in \mathbb{R}^k$ as:

$$A_{ij} = \Big\langle \psi_i, (I - \gamma P_T)\phi_j \Big\rangle \quad \text{and} \quad \mathbf{b}_i = \langle \psi_i, R \rangle \quad \text{where} \quad i,j = 1,2,\cdots k \ .$$

We can rewrite this equation as:

$$h(w) = \mathbf{b} - A\mathbf{w} = 0 \ . \tag{2.3.10}$$

This is the exact form of a stochastic approximation algorithm. Hence, the fixed point equation of the projected Bellman equation can therefore be expressed as a stochastic approximation algorithm, prompting us to say that TD is indeed an SA algorithm.

2.4. Momentum and Acceleration in Optimization

Momentum is a powerful technique for accelerating the convergence of optimization methods. Momentum-based methods have been known to achieve optimal iteration complexity for the task of minimization of smooth convex functions over convex sets. One intuitive way of adding momentum is to add the weighted average of previous steps to the current step, thereby moving in directions favored by previous steps. In the deterministic setting, leveraging information from two successive gradients for making a step has been shown to yield better convergence rates (Nesterov). Many empirical studies also showed that momentum methods are capable of exploring multiple local minima (Attouch et al. [2000]) which can help us explore and achieve a better local minima. Momentum has often also proven to be useful in non-convex settings where they have theoretically been shown to escape saddle points faster than non-accelerated versions (Jin et al. [2018]).

The introduction of momentum in optimization is largely attributed to Polyak [1964]. Motivated by the analogy of a heavy ball in a potential well defined by the objective function, Polyak [1964] introduced accelerated methods for smooth and strongly non-convex minimization. The Heavy Ball Method, which can be seen as a two-step discretization of the following second-order ODE with a constant damping term:

$$\ddot{x}(t) = \alpha_1 \dot{x}(t) + \alpha_2 \nabla f(x(t)) \ . \tag{2.4.1}$$

In relation to mechanical systems, this second-order ODE defines the dynamics of a continuous-time harmonic oscillator. Therefore, momentum-based methods in optimization can be seen as particular time-discretization of the above continuous-time harmonic oscillators. Despite the physical intuition, providing mathematical guarantees for momentum algorithms in optimization can be difficult. In the case of a quadratic objective function,

Polyak gave a spectral argument stating that his method requires no more iterations than the conjugate gradients method (Polyak [1964]). However, this eigenvalue analysis lacked rigorousness as it did not apply globally for general convex cost functions.

Another seminal work in the realm of accelerated algorithms for optimization was of Nesterov where they relied on algebraic arguments and later devised a general scheme to accelerate convex optimization methods. Instead of relying on the physical intuition of momentum and the subsequent eigenvalue arguments, Nesterov [2005] devised the method of estimate sequences to establish convergence rates of these accelerated methods. However, the motivations and the foundations of the estimate sequence methodology can be challenging to fully grasp.

There have been many attempts to propose alternate schemes for achieving acceleration (Drusvyatskiy et al. [2018], Bubeck et al. [2015], Lessard et al. [2016], Drori and Teboulle [2014], Tseng [2008]). A notable line of work in the analysis of acceleration revolves around taking the dynamical systems approach. Indeed, it can be argued that the concept of acceleration better translates into the continuous-time framework where acceleration can be viewed as a differential concept, specifically the change of speed along a curve. In the same spirit as Polyak [1964], there has been a recent surge in the dynamical system perspective for optimization. The works of Su et al. [2014] and Krichene et al. [2015] analyzed the accelerated gradient method and accelerated mirror descent with vanishing damping terms. These accelerated methods were seen as continuous time limits of second-order ODEs and showed that the trajectories of these methods approach the solutions of a given second-order ODE. The corresponding differential equations were further analyzed by Attouch et al. [2018b].

The connections between dynamical systems and optimization for the analysis of optimization algorithms have also given rise to the development of new frameworks to understand the behaviour of these accelerated methods better. A recent flurry of works has adopted a variational perspective and looked into the Lagrangian frameworks to study acceleration in continuous time (Wibisono et al. [2016], Jordan [2018], Wilson et al. [2021]). Specifically, Wibisono et al. [2016] shows that the continuous-time limits of all accelerated methods correspond to travelling the same curve in space-time generated by a specific Lagrangian functional, called the *Bregman Lagrangian*, at different speeds. There are other such frameworks developed, such as Hamiltonian mechanics (Diakonikolas and Jordan [2021]), control theory (Hu and Lessard [2017]), and high-resolution ODEs (Shi et al. [2022]).

Though this approach has helped in improving the underlying mechanisms behind the acceleration in optimization methods and developed crucial links between dynamical systems and optimization, the analysis relies heavily on the hand-engineering Lyapunov functions (Wilson et al. [2021]) or estimate sequences . Oftentimes, these functions are developed using the trial-and-error method while making educated guesses. These Lyapunov functions, which can be seen as non-increasing energy functions, are essential for the exact characterization

of the convergence rates of our algorithm. While the Lyapunov function approach allows us to move beyond estimate sequences, the lack of a systematic methodology for arriving at these functions is undesirable. Muehlebach and Jordan [2021] forgoes the Lyapunov function approach in favour of exploiting the topological properties of the dynamical system to give an explicit characterization of the convergence rates up to a constant, which holds true for both the continuous and discrete settings.

In addition to the analysis of existing algorithms, the continuous-time framework has also facilitated the development of new accelerated algorithms. A challenge in the application of the dynamical systems perspective (Betancourt et al. [2018], França et al. [2020], Maddison et al. [2018]) is that rate-matching discrete-time counterparts depend on the chosen discretization scheme: finding the right scheme is a challenge on its own. Previous works have shown that naive discretizations of the continuous-time dynamical system can fail to replicate the fast oracle rates seen in the underlying system or lead to unstable discrete-time systems (Wibisono et al. [2016]). Subsequently, there has been a lot of work on discretizing these ODEs with vanishing damping terms (Wibisono et al. [2016], Attouch et al. [2018a], Attouch and Cabot [2018], Adly and Attouch [2022], Diakonikolas and Jordan [2021]) and discretizing alternate ODEs (Wilson et al. [2019], Muehlebach and Jordan [2019], Zhang et al. [2019]).

Motivated by the Lagrangian and Hamiltonian mechanics, a body of work has explored physical interpretations of Polyak's (Attouch and Alvarez [2000], Attouch et al. [2000], França et al. [2020]) and Nesterov's (Wibisono and Wilson [2015], Wibisono et al. [2016], Betancourt et al. [2018]) methods in the Lagrangian and Hamiltonian formulation. An added benefit of working with the Lagrangian or Hamiltonian formulations of the continuous-time dynamical systems is the existence of certain conserved quantities of the Hamiltonian which can be analyzed to characterize the convergence rate (Diakonikolas and Jordan [2021]). Since these conserved quantities can be viewed as Lyapunov functions, this provides a more intuitive way of arriving at these functions whose origins have often been unclear (Suh et al. [2022], Diakonikolas and Jordan [2021]). Specifically, instead of guessing these Lyapunov functions, they are derived from the same Hamiltonian whose equations of motion are the momentum dynamics.

Another notable work that explores a similar line of thought is that of Suh et al. [2022] where they develop a methodology for analyzing these accelerated methods by deriving conservation laws, which can be seen to be analogous to the conservation of energy in physics, in a dilated coordinate system. Along with providing a unified framework for the analysis of accelerated methods in the continuous time realm, they also provide a streamlined methodology for deriving Lyapunov functions from first principles. This work offers a more intuitive way of starting from the given ODE and arriving at the conservation law, whereas

Diakonikolas and Jordan [2021] start from a Hamiltonian with given kinetic energy and potential energy terms which are then used to derive the ODE.

Working with Lagrangian or Hamiltonian formulations also allows for using symplectic integration (Wanner and Hairer [1996])for the discretization of ODEs while preserving the continuous symmetries of the underlying system (Haier et al. [2006]). These conserved symmetries could also include meaningful terms such as energy or momentum which can be conserved exactly by these symplectic integrators even if the dynamical flow is only approximated. This idea was initiated in the field of accelerated optimization methods by Betancourt et al. [2018] for their ability to be more stable than other integration schemes, allowing us to use larger step sizes in the discrete-time systems without off-shooting. This approach has been further explored in works like (Maddison et al. [2018], França et al. [2020], Muehlebach and Jordan [2021], França et al. [2021], Shi et al. [2019]), with Shi et al. [2019] achieving an $O(1/k^2)$ by combining symplectic integration with a high-resolution ODE framework.

2.5. Acceleration of Temporal Difference Methods

While RL has made big strides with the introduction of deep-learning techniques with the advent of Deep RL, the fundamental ideas underlying these new algorithms remain the same. Q-Learning algorithm proposed by Watkins and Dayan (Watkins [1989], Watkins and Dayan [1992]) forms the backbone of deep Q-Learning. In addition to it, actor-critic methods or policy gradient algorithms have adapted notions from the monumental works of Sutton et al. [1999], and Konda and Tsitsiklis [1999]. However, at the heart of these advancements lies the most fundamental RL algorithm: Temporal Difference Learning (Sutton [1988], Tsitsiklis and Van Roy [1996]).

It is widely known that RL algorithms like TD-learning and Q-learning are slow to converge and sample inefficient (Szepesvári [1997], Azar et al. [2011]). This necessitates the development of accelerated algorithms for TD learning. With the development of new algorithms, mathematical analysis of these algorithms is important for building interpretable, stable, and convergent algorithms. For the purpose of this thesis, we focus on developing convergent algorithms along with an asymptotic analysis for the fundamental building block of RL: TD(0) with linear function approximation. Since this work aims to develop a new approach to algorithm analysis, it is prudent to start the development of these methods on algorithms that have known convergent guarantees with existing analysis methodologies.

Bounds for convergence rate can often be obtained either in high-probability or in expectation. There are two main styles of analysis in the literature for TD: asymptotic analysis, and finite-time analysis. Recent works have focused on obtaining finite time bounds for TD methods (Dalal et al. [2018b], Dalal et al. [2018a], Lakshminarayanan and Szepesvari [2018]).

Bounds for two-time scale SA algorithms were developed in Dalal et al. [2018b]. Finite-time bounds for TD(0) with i.i.d observation noise assumption were derived in Dalal et al. [2018a] and Lakshminarayanan and Szepesvari [2018]. However, research on finite time bounds in realistic settings with Markovian noise have been fairly recent. Bhandari et al. [2018] provides an explicit finite-time analysis similar to the ones seen in stochastic gradient descent by providing bounds for vanishing and finite step-sizes for projected TD-learning. Another work obtaining finite-time bounds on mean square error for constant step-size is Srikant and Ying [2019] where they do this by considering the drift of an appropriately chosen Lyapunov function. While establishing finite-time bounds are desirable and often more realistic, we note that these methods do not provide for a unified theory to design new algorithms but serve as a way to analyze the existing ones.

Since the introduction of stochastic approximation methods in the early 1950s with the seminal work of Robbins and Monro [1951], the ODE method has been a staple of algorithm design and analysis (Borkar and Meyn [2000]). The ODE approach relies on the idea that, under certain conditions, the noise seen in these SA iterates averages out and they closely track the trajectory of a limiting ODE. The asymptotic analysis theory for stochastic approximation methods has been well-developed and is often a convenient way to show asymptotic convergence of these SA algorithms (Kushner and Yin [2003], Borkar [2009], Benveniste et al. [2012]). While this approach is often convenient as it allows to circumvent issues pertaining to noise, it fails to provide insights into the effect of noise, choice of step-size, or ill-conditioning on the algorithm. To counter this, an approach based on the central limit theorem for the SA algorithm was developed by Konda and Tsitsiklis [1999], Devraj and Meyn [2017] and, Devraj [2019].

A prominent class of new TD algorithms is that of gradient TD-learning methods (GTD, GTD2, and TDC) which were first introduced by Sutton et al. [2008, 2009] to address the divergence of TD in the off-policy regime. For RL, off-policy translates to learning the value function for the *target* policy, using data obtained while following a different *behavior* policy. These methods were designed to replicate gradient descent over the mean squared projected Bellman error loss function (MSPBE).

Given the close relation between gradient-based TD methods and gradient descent, adding a momentum term to accelerate the convergence of the iterates, similar to what is often seen in stochastic gradient methods, is a fairly straightforward extension. Hence, a lot of work on developing accelerated algorithms has been done with the gradient TD or fitted value methods (Meyer et al. [2014], Pan et al. [2017], Deb and Bhatnagar [2022], Ghiassian et al. [2020]).

One of the first significant attempts at incorporating momentum seen in gradient descent to TD methods is that of Meyer et al. [2014] who extended Nesterov's momentum seen in gradient descent algorithms to Gradient TD methods to the setting of policy evaluation

with linear function approximation for on-policy learning. Pan et al. [2017] introduced an accelerated gradient TD (ATD) method that performed quasi-second-order gradient descent on the MSPBE that provided similar data efficiency benefits as least-squares methods at a fraction of the computation and storage cost. Ghiassian et al. [2020] introduced another Gradient TD method called TD with regularized corrections (TDRC) which makes slight modifications to the TDC methods while achieving performance seen in TD methods.

Farahmand and Ghavamzadeh [2021] view the Value Iteration (VI) algorithm as a dynamical system and propose modifications to using tools from control theory to accelerate their dynamics. The modifications are based on simply controllers that show accelerated behaviour in these modified VI algorithms empirically.

Momentum counterparts were also introduced to the general SA algorithm with applications to TD and Q-learning (Devraj et al. [2019], Mou et al. [2020], Avrachenkov et al. [2022], Deb and Bhatnagar [2022]). Devraj et al. [2019] introduce matrix momentum to SA with applications to Q-learning while providing a unified perspective on Polyak's heavy ball method as the linearization of a particular Nesterov's method. However, it can be argued that matrix momentum is not equivalent to the heavy ball momentum. Mou et al. [2020] briefly discuss SA with momentum and show an improvement in the mixing rate for linear SA where the matrix A is Hurwitz with a broader discussion on asymptotic and non-asymptotic properties of linear SA algorithms with Polyak-Ruppert averaging. Avrachenkov et al. [2022] studied one time-scale (TS) SA with a heavy ball momentum term in the univariate case in the context of web-page crawling. Deb and Bhatnagar [2022] consider One-TS and Three-TS decomposition of Gradient TD with heavy ball momentum iterates and provide asymptotic convergence guarantees.

Recent works have also attempted to accelerate Watkins' Q-learning algorithm which is known to be slow to converge and have poor performance (Azar et al. [2011], Devraj and Meyn [2017], Chen et al. [2020], Chen et al. [2021]). Szepesvári [1997] first quantified the slow convergence rates of the original Watkins'algorithm and obtained explicit finite-time PAC (probably almost correct) bounds and Azar et al. [2011] provide such bounds for their proposed Q-learning algorithm with faster convergence guarantees. Devraj and Meyn [2017] introduced the Zap Q-learning with a two time-scale update for a matrix gain sequence such that the asymptotic variance is optimal and faster convergence is obtained. This framework was then extended by Chen et al. [2020] to ensure stability of the algorithm. Most of the above works show convergence in the linear setting, however Chen et al. [2020] designed a new class of Q-learning algorithms that are convergent even for non-linear function approximation architectures, like neural networks.

2.6. Performance Metrics

While off-policy TD is known to diverge with function approximation, several TD-based algorithms, especially in the on-policy setting, are proven to be convergent. Also, when TD learning converges, it converges to the TD fixed point. Note that the TD fixed point corresponds to the weight vector $\mathbf{w}_\star$ when $h(\mathbf{w}_\star) = 0$.

A natural question that might arise is: Which choice of performance metric would best help encapsulate the advantages of a given TD method over others while showing the method to be convergent in nature? Previous works on accelerated gradient temporal difference methods (Sutton et al. [2009]) that looked into choosing a different objective (such as GTD methods) chose the Mean Square Projected Bellman Error (MSPBE). Following this, most works measured the performance of their methods by reporting the Root Mean Square Projected Bellman Error (RMSPBE).

We start with describing the possible choices of functions that could be used to evaluate the performance of our methods and show why RMSPBE is the preferred choice. This explanation heavily relies on the detailed discussion given in Sutton et al. [2009].

Consider the approximated value function to be $v_{\mathbf{w}}$ and the true value function to be $v^\star$. An obvious choice to evaluate the methods is the mean squared error (MSE) between $v_{\mathbf{w}}$ and $v^\star$, averaged over the state space depending on how often each state occurs. We can write this as:

$$\text{MSE}(\mathbf{w}) = \sum_s x_d(v_{\mathbf{w}} - v^\star)^2 = ||\mathbf{v}_{\mathbf{w}} - \mathbf{v}^\star||_X^2 \ . \tag{2.6.1}$$

Note that, in the second equation $\mathbf{v}_{\mathbf{w}}$ and $\mathbf{v}^\star$ are vectors and the norm $||v||_X^2 = v^T X v$ where X is a diagonal matrix with the limiting distribution x_d on its diagonal.

While this function might seem like a good choice on first glance, the idea behind temporal difference methods is for the approximated value function to closely satisfy the Bellman equation. The true value function $v^\star$ satisfies the Bellman function exactly to give:

$$\mathbf{v}^\star = r + \gamma P \mathbf{v}^\star = T\mathbf{v}^\star \ ,$$

where T is the Bellman operator. Another measure of how closely the approximate value function satisfies the Bellman equation could therefore be the mean squared Bellman error given by:

$$\text{MSBE} = ||\mathbf{v}_{\mathbf{w}} - T\mathbf{v}_{\mathbf{w}}||_X^2 \ . \tag{2.6.2}$$

However, it was observed that most temporal difference methods, like TD, LSTD, and GTD, do not converge to the minimum of MSBE. This happens due to the presence of function approximators in the value function. The Bellman operator follows the dynamics of the underlying Markov Chain, irrespective of the function approximator used. This results in $T\mathbf{v}_{\mathbf{w}}$ not being representable by $\mathbf{v}_{\mathbf{w}}$ for any $\mathbf{w}$. We therefore require a projection operator,

which takes a value function $\mathbf{v}$ and projects it to the nearest possible value function that is representable by a function approximator. We call this projection operator Π which can be written as:

$$\Pi\mathbf{v} = \mathbf{v}_{\mathbf{w}} \quad \text{where} \quad \mathbf{w} = \arg\min_{\mathbf{w}} ||\mathbf{v}_{\mathbf{w}} - \mathbf{v}||_X^2 \ .$$

For our case with linear function approximator ($\mathbf{v}_{\mathbf{w}} = \boldsymbol{\phi}^T(s_k)\mathbf{w}$), the projection operator turns out to be linear and independent of $\mathbf{w}$ and can be written as:

$$\Pi = \boldsymbol{\phi}(\boldsymbol{\phi}^T X \boldsymbol{\phi})^{-1}\boldsymbol{\phi}^T X \ .$$

All the temporal difference methods mentioned above are seen to converge to a fixed point of the composed projected Bellman operator, converging to a value $\mathbf{w}$ such that:

$$\mathbf{v}_{\mathbf{w}} = \Pi T \mathbf{v}_{\mathbf{w}} \ .$$

Hence, we can now empirically track the convergence properties of our method by using the quantity mean square projected Bellman error (MSPBE), which can be defined as:

$$\text{MSPBE}(\mathbf{w}) = ||\mathbf{v} - \Pi T \mathbf{v}_{\mathbf{w}}||_X^2 \ . \tag{2.6.3}$$

For our specific case, we can further expand on this definition of MSPBE to better understand this quantity. To do so, we would first rewrite our TD update in terms of the TD Error as:

$$\begin{aligned} \delta_k &= \mathbf{r}_k + \gamma \boldsymbol{\phi}^T(s_{k+1})\mathbf{w}(s_k) - \boldsymbol{\phi}^T(s_k)\mathbf{w}_k \\ \mathbf{w}(s_{k+1}) &= \mathbf{w}(s_k) + \alpha\delta_k \mathbf{w}(s_k) \end{aligned} \tag{2.6.4}$$

for the stepsizes $\alpha_t > 0$. MSPBE (Sutton et al. [2009]) is then defined as:

$$\begin{aligned} \text{MSPBE}(\mathbf{w}(s_k)) &= \mathbb{E}[\delta_k \boldsymbol{\phi}(s_k)]^T \mathbb{E}[\boldsymbol{\phi}(s_k)\boldsymbol{\phi}(s_k)]^{-1}\mathbb{E}[\delta_k \boldsymbol{\phi}(s_k)] \\ &= (\mathbf{b} - A\mathbf{w})^T C^{-1}(\mathbf{b} - A\mathbf{w}) \ , \end{aligned}$$

$$\begin{aligned} \text{where} \quad \mathbb{E}[\delta_k \boldsymbol{\phi}(s_k)] = \mathbf{b} - A\mathbf{w}_k, \quad C = \mathbb{E}[\boldsymbol{\phi}(s_k)\boldsymbol{\phi}^T(s_k)], \\ A = \mathbb{E}[\boldsymbol{\phi}(s_k)(\boldsymbol{\phi}(s_k) - \gamma \boldsymbol{\phi}^{'T}(s_k))], \quad \text{and} \quad b = \mathbb{E}[r_k \boldsymbol{\phi}(s_k)] \ . \end{aligned}$$

Note that the TD fixed point corresponds to $\mathbb{E}[\delta_k \mathbf{w}_k] = 0$ which happens to be the solution of the system $A\mathbf{w}_k = \mathbf{b}$. This is the underlying limiting ODE for our system defined as $h(\mathbf{w}) = \mathbf{b} - A\mathbf{w}(t)$. Hence, this shows that our method also optimizes for the MSPBE without explicitly taking the gradient of it. It also shows that RMSPBE therefore serves as an apt metric to check for convergence of our accelerated TD methods.

2.7. Discretization schemes

This section explores the use of numerical methods for transitioning from differential equations to implementable difference equations. Note that this transition from continuous-time ODEs to the discrete-time algorithm is a non-trivial step. There are numerous discretization schemes that attempt to come up with a discrete-time algorithm that preserves the properties of the continuous-time system (Wanner and Hairer [1996]). Note that we are only concerned with time-discretization here. The idea is to choose a *consistent* discretization scheme for a given differential equation. A discretization scheme is said to be *consistent* if the discretized equations converge to the given ODE as the time step tends to zero (Blazek [2015]).

Consider a function $w(t)$ that is infinitely differentiable with respect to t. We can therefore use the Taylor series method to approximate the value of the function w at $t + \Delta t$ using the function value at t with derivatives as:

$$w(t + \Delta t) = w(t) + \Delta t \cdot \dot{w}(t) + \frac{\Delta^2 t}{2}\ddot{w}(t) + \cdots + \frac{\Delta^n t}{n!}w^{(n)}(t) \ ,$$

where $w^{(n)}(t)$ denotes the n^{th} derivative of the function w with respect to t. First order discretization schemes, like Euler's method, correspond to using the first derivative $\dot{w}(t)$ to approximate the function value. Second order discretization schemes, like Verlet integration and leapfrog integrator, also include the second derivative, $\ddot{w}(t)$ to compute the function value whereas higher order symplectic schemes, like Runge-Kutta method, include more derivative terms in the Taylor series expansion to approximate the function value. Second or higher order discretization schemes are more computationally intensive than first order schemes but provide more stability. Note that in most settings, it is not practical to use higher order discretization schemes as we do not have more than two derivatives. For this work, we focus on using first-order schemes for the sake of simplicity.

Consider a first-order ODE given by:

$$\dot{w}(t) = h(w(t)) \ , \quad \text{where} \quad w(t_0) = w_0 \ . \tag{2.7.1}$$

Our goal is to derive a difference equation which is an approximation of ODE (2.7.1) that involves difference in only function values without the presence of any derivatives.

2.7.1. Euler Methods

The simplest method to discretize this ODE is the Euler method. The basic idea here is to obtain a difference equation which involves differences in function value w at different time steps t_i.

From calculus, we can write the derivative of a function $w(t)$ at a given point $t = a$ as:

$$\dot{w}(a) = \lim_{i \to 0} \frac{w(a+i) - w(a)}{i} \ . \tag{2.7.2}$$

Physically, this just means taking the slope of the secant line joining $(a, w(a))$ and $(a + i, w(a+i))$. Note that as $a+i$ gets closer to a, the slope of this secant line gets closer to the slope of the tangent line at a, given by $\dot{w}(a)$. For a small fixed i, we have the approximation of $\dot{w}(a)$ to be equal to . Now, for a limit to exist, both the limit as $i \to 0$ approached from i^+ and i^- should be equal. Different forms of the Euler method correspond to approaching this limit from either the right (i^+) or the left (i^-).

Explicit Euler Methods: Explicit Euler Methods correspond to taking this limit in the definition of the derivative from the right (i^+). From the fundamental theorem of calculus, we can write:

$$\dot{w}(t_0)) \approx \frac{w(t_1) - w(t_0)}{\Delta t}$$

as an approximation to $\dot{w}(t_0))$. Let $w_1 = w(t_1)$ and $w_0 = w(t_0)$. We can use (2.7.1) to give us our Euler method. Note that to obtain the Euler method at t_1, we use our ODE (2.7.1) evaluated at t_0 along with our approximation to $\dot{w}(t_0)$ to get:

$$\frac{w_1 - w_0}{\Delta t} = h(w_0,t_0)$$
$$w_1 = w_0 + \Delta t \cdot h(w_0, t_0) \ .$$

We can now write the Explicit Euler update in general as:

$$w_{k+1} = w_k + \Delta t \cdot h(w_k, k) \ . \tag{2.7.3}$$

We call this method an explicit scheme because we can write the unknown quantity in terms of known quantities making the computations of the unknown quantities easy. This method is also called the forward Euler method because we write the equation at the point k and the difference forward in time to $k+1$. It essentially translates to making an update at time step $k+1$ using function values obtained at time step k.

Implicit Euler Methods: Note that the explicit Euler scheme was derived by evaluating the approximation of $\dot{w}(t_0)$ using the definition of the derivative at a given point a using $i \to 0^+$. Implicit Euler obtains a difference equation by taking the limit from the left $(i \to 0^-)$. Hence, the secant line is now between $(a, w(a))$ and a point to its left for approximating $\dot{w}(a)$. In (2.7.2), we can therefore set $a + h = t_0$ and $a = t_1$. This gives us:

$$\dot{w}(t_1) \approx \frac{w(t_0) - w(t_1)}{t_0 - t_1} \ .$$

Note that since $t_0 - t_1 < 0$, this gives us the approximation at time t_1 to be:

$$w_1 = w_0 + \Delta t \cdot h(w_1, t_1) \ .$$

We can now write the implicit Euler update in general as:

$$w_{k+1} = w_k + \Delta t \cdot h(w_{k+1}, k+1) \ . \tag{2.7.4}$$

This method is also called the backward Euler method as we are writing the equation at $k+1$ and differencing backward in time. Note that (2.7.4) differs from (2.7.3) since we must evaluate the function value $h(w,t)$ at the unknown point $(w_{k+1}, k+1)$ which leads to a non-linear equation to solve for each w_{k+1}. Therefore, while the implicit method can be more stable in nature than the explicit Euler method, it can also be computationally more intensive. Also, note that these implicit methods are not more accurate than explicit methods, but more stable in nature.

Symplectic Euler Methods: A natural question that arises is: *How do we leverage the stability of implicit Euler methods while reducing the computational cost as seen in the explicit Euler methods?* Symplectic Euler methods are an answer to this. These methods are often used to discretize Hamiltonian equations and almost conserve the energy (when the Hamiltonian is time-independent).

Consider the system of differential equations of the form:

$$\begin{aligned} \dot{w}(t) &= h(y,t) \\ \dot{y}(t) &= g(w,t) \end{aligned} \tag{2.7.5}$$

We use the explicit Euler method on (2.7.5) (a) and the implicit Euler method to perform the discretization on (2.7.5) (b). Therefore, we can write the symplectic Euler update for the system of equations given in (2.7.5) as:

$$\begin{aligned} w_{k+1} &= w_k + \Delta t \cdot h(y_k, k) \\ y_{k+1} &= y_k + \Delta t \cdot g(w_{k+1}, k+1) \end{aligned} \tag{2.7.6}$$

2.8. Concluding Remarks

In this chapter, we start off with a discussion of basic concepts from Dynamic Programming and RL, like value functions, Bellman equations, and approximate dynamic programming techniques in Section 2.1. We then move on to describing policy evaluation methods and our specific setting of TD methods in Section 2.2. We follow this by describing the generic stochastic approximation problem, describing the ODE approach to analyzing such algorithms, and establishing a link between ADP and SA algorithms in Section 2.3. Specifically, this section helps us see that TD is an SA method which is an instrumental fact in the derivation of our proposed methods. We then switch to talk about accelerated methods in optimization, focusing especially on the dynamical systems viewpoint taken to understand the behaviour of these methods and analyze them in Section 2.4. Following this, we discuss the directions of work addressing acceleration in TD methods or RL, in general, in Section

2.5. Finally, we describe the performance metrics used for evaluating our methods in Section 2.6. Lastly, we describe the first-order Euler discretization schemes in Section 2.7. We would now use the notions described in this Chapter to design accelerated algorithms for TD methods.

Chapter 3

Accelerated Algorithms for TD Learning

In this Chapter, we build upon ideas previously discussed in Section 2.4 for designing accelerated TD methods. Note that throughout this thesis, we work on the on-policy setting for TD(0) with linear function approximation. While we focus on devising accelerated algorithms in this setting, these proofs serve as a recipe that can be used to easily derive such accelerated algorithms in other settings.

We start this chapter by showing how we can express the TD(0) algorithm as a limiting ODE. Note that this particular limiting ODE is not new to SA and is used to establish stability and convergence of the SA iterates (see Section 2.3.1). Taking inspiration from the optimization literature establishing acceleration to be a second-order ODE with interpretations as a mass-spring damper system (Su et al. [2014], Muehlebach and Jordan [2019]), we modify the first-order limiting ODE of TD to two second-order ODEs: one corresponding to Polyak's Heavy Ball method and the other corresponding to Nesterov's acceleration. Lastly, we perform two different discretization schemes: explicit Euler and symplectic Euler on both these ODEs to obtain four implementable discrete-time accelerated TD algorithms.

3.1. ODE for TD(0)

We know that the TD(0) update rule can be given by:

$$v^{(t+1)}(s_t) = v^{(t)}(s_t) + \eta_t[r_t + \gamma v^{(t)}(s_{t+1}) - v^{(t)}(s_t)]] \ .$$

Consider the on-policy setting for our TD(0) algorithm where the state transitions come directly from the evolution of a Markov Chain. For the sake of simplicity, we consider the case of the TD(0) algorithm with linear function approximation, where we approximate the value function across state space as $v = \boldsymbol{\phi}^T(s)\mathbf{w}$. Here, $\Phi \in \mathbb{R}^{|S| \times k}$ is the concatenation of all feature vectors as rows. Note that $|S|$ is the number of states and k is the dimension of the features. Hence, $\boldsymbol{\phi}(s) \in \mathbb{R}^{k \times 1}$ corresponds to the s^{th} feature vector. $\mathbf{w} \in \mathbb{R}^{k \times 1}$ is the weight vector to be learned. Note that Φ is assumed to be full rank for $\mathbf{w}$ to be unique. η

is the learning rate of this algorithm. We can therefore rewrite the TD(0) update rule as:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \eta_t[r_t + \gamma \boldsymbol{\phi}^T(s_{t+1})\mathbf{w}^{(t)} - \boldsymbol{\phi}^T(s_t)\mathbf{w}^{(t)}]\boldsymbol{\phi}(s_t) \ . \tag{3.1.1}$$

To study the average behavior or for analyzing the corresponding deterministic system of this stochastic process, we need to average out the noise and look at the ODE for the given deterministic system. To average out the noise, we take the expectation of the SA iterates under the stationary distribution induced by running a given policy inside our MDP. Note that a Markov Chain that is irreducible and aperiodic has a unique stationary distribution $\mathbf{x}_d \in \mathbb{R}^{|S|\times 1}$. Since $\mathbf{x}_d$ is a stationary distribution, we also have $\mathbf{x}_d^T = \mathbf{x}_d^T P_d$ where $P_d \in \mathbb{R}^{|S|\times|S|}$.

Taking expectation of (3.1.1) where s_t and s_{t+1} come from stationary distribution $\mathbf{x}_d$ gives us:

$$\begin{aligned}
&\mathbb{E}_{\mathbf{x}_d}[\Big(r_t + \gamma \boldsymbol{\phi}^T(s_{t+1})\mathbf{w} - \boldsymbol{\phi}^T(s_t)\mathbf{w}\Big)\boldsymbol{\phi}(s_t)] \\
&\qquad = \mathbb{E}[r(s_t,a_t)\boldsymbol{\phi}(s_t) + \gamma \boldsymbol{\phi}^T(s_{t+1})\mathbf{w}\boldsymbol{\phi}(s_t) - \boldsymbol{\phi}^T(s_t)\mathbf{w}\boldsymbol{\phi}(s_t)] \\
&\qquad = \sum_{i,j\in S, a\in A} [x_d(i)\boldsymbol{\phi}(i)r(i,a) + \gamma x_d(i)[P_d]_{ij}\boldsymbol{\phi}(i)^T\boldsymbol{\phi}(j)w(i) - x_d(i)\boldsymbol{\phi}(i)\boldsymbol{\phi}(\boldsymbol{i})^T w(i)] \\
&\qquad = \Phi^T X_d \mathbf{r}_d + \gamma \Phi^T X_d P_d \Phi\mathbf{w} - \Phi^T X_d \Phi\mathbf{w} \\
&\qquad = \Phi^T X_d(\mathbf{r}_d - (I - \gamma P_d)\Phi\mathbf{w}) \ ,
\end{aligned}$$

where $\mathbf{r}_d \in \mathbb{R}^{|S|\times 1}$ and $X_d \in \mathbb{R}^{|S|\times|S|}$.

Hence, averaging the iterates out under the stationary distribution gives us the underlying limiting ODE for TD(0):

$$\dot{\mathbf{w}}(t) = \Phi^T X_d \mathbf{r}_d - \Phi^T X_d(I - \gamma P_d)\Phi\mathbf{w} \ . \tag{3.1.2}$$

Note that this equation now defines a system that is entirely deterministic. We can also rephrase this equation to mimic the form for SA algorithms for the ease of analysis. Recall that stochastic approximation aims to solve non-linear problems of the type $h(\mathbf{w}) = 0$ where $\mathbf{w}^\star$ is the solution to this equation such that $h(\mathbf{w}^\star) = 0$. For the specific case of TD(0) with linear function approximation, we can write (3.1.1) in terms of SA iterates as:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \alpha_t[h(\mathbf{w}_t) + M_{t+1}] \ , \tag{3.1.3}$$

where M_{t+1} is the Martingale difference taken to be zero (i.e., $M_{t+1} = 0$) and $h(\mathbf{w}) = \mathbf{b} - A\mathbf{w}$ with

$$A = \mathbb{E}_{\mathbf{x}_d}[\boldsymbol{\phi}(s_t)(\boldsymbol{\phi}(s_t) - \gamma\boldsymbol{\phi}(s_{t+1}))^T] \quad \text{and} \quad \mathbf{b} = \mathbb{E}_{\mathbf{x}_d}[\mathbf{r}_t\boldsymbol{\phi}(s_t)] \ . \tag{3.1.4}$$

Note that matrix A is assumed to be positive definite and (3.1.3) converges to $\mathbf{w}^\star := A^{-1}b$. We can therefore write the limiting ODE as

$$\dot{\mathbf{w}}(t) = h(\mathbf{w}(t)) = \mathbf{b} - A\mathbf{w}(t) \ . \tag{3.1.5}$$

3.2. Polyak's momentum for TD(0)

Interpretations of accelerated ODEs in optimization as mass-spring damper systems have been long established (Su et al. [2014]). Instead of starting with the difference equation and deriving the corresponding ODE, we approach this in the converse way. We start from a particular second-order ODE with interpretations as a mass-spring damper system and show that Polyak's momentum version of TD(0) can be derived from the discretizations of these continuous-time systems.

We can also draw parallels between accelerated methods and damped harmonic oscillators in physics with second-order ODEs (Su et al. [2014], Wibisono et al. [2016]). Consider the ODE form for forced mass-spring damper systems which is given by

$$M\frac{d^2y}{dt^2} + R\frac{dy}{dt} + ky = f(t) \ .$$

Extending this idea to TD(0) and modifying the limiting ODE obtained in (3.1.5), we get a second-order ODE for the form:

$$\ddot{\mathbf{w}}(t) + \beta(t)\dot{\mathbf{w}}(t) = h(\mathbf{w}(t)) = \mathbf{b} - A\mathbf{w}(t) \ , \tag{3.2.1}$$

$$\text{where} \quad A = \mathbb{E}_{\mathbf{x}_d}[\boldsymbol{\phi}(s_t)(\boldsymbol{\phi}(s_t) - \gamma\boldsymbol{\phi}(s_{t+1}))^T] \quad \text{and} \quad b = \mathbb{E}_{\mathbf{x}_d}[\mathbf{r}_t\boldsymbol{\phi}(s_t)] \ . \tag{3.2.2}$$

This is our proposed limiting ODE for Polyak's momentum in TD(0).

We will now perform two different kinds of discretizations **(1)** Explicit Euler discretization and **(2)** Symplectic Euler discretization to get the modified update equations. Note that choosing different discretization schemes could yield us different discrete time algorithms from the same ODE. Different discretization schemes correspond to different approximations. Taking inspiration from ideas developed in Section 2.7, we will discretize our second-order ODE given by (3.2.1). Since we are working with a second-order ODE, we perform these discretizations by converting the second-order ODE into two first-order ODEs. Let $\dot{\mathbf{w}}(t) = \mathbf{z}(t)$. We can therefore express (3.2.1) as a system of two first-order ODEs as:

$$\begin{aligned} \dot{\mathbf{w}}(t) &= \mathbf{z}(t) \\ \dot{\mathbf{z}}(t) = \ddot{\mathbf{w}}(t) &= h(\mathbf{w}(t)) - \beta(t)\mathbf{z}(t) \ . \end{aligned} \tag{3.2.3}$$

3.2.1. Explicit Euler discretization

We will now discretize (3.2.3) using the explicit Euler scheme discussed in 2.7.1. By doing so, we get the following equations:

$$\begin{aligned} \mathbf{w}_{k+1} &= \mathbf{w}_k + \eta\mathbf{z}_k \\ \mathbf{z}_{k+1} &= \mathbf{z}_k - \eta\beta_k\mathbf{z}_k + \eta h(\mathbf{w}_k) \ . \end{aligned} \tag{3.2.4}$$

Here, $\Delta t = \eta$ is the step size or the learning rate.

We will now substitute the $h(\mathbf{w}_k)$ in the equation with values of A and b from (3.2.2). We have $v^{(t)}(s_t) = \boldsymbol{\phi}^T(s_t)\mathbf{w}_t$ and $u^{(t)}(s_t) = \boldsymbol{\phi}^T(s_t)\mathbf{z}_t$. This gives us the following update equations for TD(0) with momentum:

$$\begin{aligned} v^{(t+1)}(s_t) &= v^{(t)}(s_t) + \eta u^{(t)}(s_t) \\ u^{(t+1)}(s_t) &= u^{(t)}(s_t) - \eta\beta_t u^{(t)}(s_t) + \eta(r_t + \gamma v^{(t)}(s_{t+1}) - v^{(t)}(s_t)) \ . \end{aligned} \tag{3.2.5}$$

Note that β_t is the damping parameter and is a function of time. The analytical form of β_t will be given using the conservation law that will be derived in the next chapter.

Similar to the algorithm seen in Sutton and Barto [2018], we can now define our algorithm for TD(0) with Polyak's Heavy Ball momentum using explicit Euler discretization as shown in Algorithm 1. Note that the update terms defined in Algorithm 1 differ from the analytical

Algorithm 1: Polyak TD(0) Algorithm with Explicit Euler discretization

Input: Policy π to be evaluated, Set η, β_k, γ
Initialize $V(s)$ arbitrarily (e.g., $V(s) = 0 \quad \forall \quad s \in S^+$)
Repeat (for each episode):
    Initialize S; Set $V(s) = 0$, $k = 1$
    Repeat (for each step of the episode):
        $A \leftarrow$ action given by π for S
        Take action A, observe R, $S^{'}$ (new state)
        $V_{prev}(S) \leftarrow V(S)$
        $V_{prev}(S') \leftarrow V(S')$
        $V(S) \leftarrow V(S) + \eta U(S)$
        $U(S) \leftarrow \eta U(S) - \eta\beta_k U(S) + \eta[R + \gamma V_{prev}(S^{'}) - V_{prev}(S)]$
        $k \leftarrow k + 1$
        $S \leftarrow S^{'}$
    until S **is terminal**

updates given in (3.3.5) by the additional scaling factor of η in $u^{(t)}(s_t)$ term. Note that this phenomenon has also been observed in Shi et al. [2019] where there was an additional scaling factor introduced in the update equations of the infamous Nesterov's Accelerated Gradient (NAG) method when compared with the discrete-time counterparts obtained by simply discretizing the ODE. We will discuss this in more detail in Chapter 5.

3.2.2. Symplectic Euler discretization

We will now discretize (3.2.1) using the symplectic Euler scheme discussed in 2.7.1. Recall that in this scheme, we use explicit Euler to advance one component and implicit Euler to advance another. Note that since we simply combine explicit and implicit steps, this does not strictly correspond to a *symplectic* discretization as we are not preserving the symplectic structure. Using this discretization, we get the following discrete-time counterparts for

(3.2.1):

$$\begin{aligned} \mathbf{w}_{k+1} &= \mathbf{w}_k + \eta \mathbf{z}_k \\ \mathbf{z}_{k+1} &= \mathbf{z}_k - \eta\beta_{k+1}\mathbf{z}_{k+1} + \eta h(\mathbf{w}_{k+1}) \ , \end{aligned} \tag{3.2.6}$$

where η is the associated learning rate for the update equation.

Note that we can further simplify this equation by clubbing terms with $\mathbf{z}_{k+1}$ as:

$$\begin{aligned} \mathbf{w}_{k+1} &= \mathbf{w}_k + \eta \mathbf{z}_k \\ \mathbf{z}_{k+1} &= \frac{1}{(1+\eta\beta_{k+1})} \left[\mathbf{z}_k + \eta h(\mathbf{w}_{k+1})\right] \ , \end{aligned}$$

We will now substitute the $h(\mathbf{w}_{k+1})$ in (3.3.4) with values of A and $\mathbf{b}$ from (3.1.4). Also, note that $v^{(t)}(s_t) = \boldsymbol{\phi}^T(s_t)\mathbf{w}_t$ and $u^{(t)}(s_t) = \boldsymbol{\phi}^T(s_t)\mathbf{z}_t$. This gives us the following update equations for Polyak TD(0) with symplectic euler discretization:

$$\begin{aligned} v^{(t+1)}(s_t) &= v^{(t)}(s_t) + \eta u^{(t)}(s_t) \\ u^{(t+1)}(s_t) &= \left(\frac{1}{1+\eta\beta_{t+1}}\right)\left[u^{(t)}(s_t) + \eta\left(r_t + \gamma v^{(t+1)}(s_{t+1}) - v^{(t+1)}(s_t)\right)\right] \ . \end{aligned} \tag{3.2.7}$$

We can now define our algorithm for TD(0) with Polyak's Heavy ball momentum discretized using symplectic Euler method as follows:

Algorithm 2: Polyak TD(0) Algorithm with Symplectic Euler discretization

Input: Policy π to be evaluated, Set η, β_k, γ
Initialize $V(s)$ arbitrarily (e.g., $V(s) = 0 \quad \forall \quad s \in S^+$)
Repeat (for each episode):
    Initialize S; Set $V(s) = 0$, $k = 1$
    Repeat (for each step of the episode):
        $A \leftarrow$ action given by π for S
        Take action A, observe R, S' (new state)
        $V(S) \leftarrow V(S) + \eta U(S)$
        $U(S) \leftarrow \left(\frac{1}{1+\eta\beta_{k+1}}\right)\left[\eta U(S) + \eta\left(R + \gamma V(S') - V(S)\right)\right]$
        $k \leftarrow k + 1$
        $S \leftarrow S'$
    until S **is terminal**

Similar to the additional scaling factor introduced in Algorithm 1, we see that the analytical discrete-time updates given in (3.2.7) differ from the updates in Algorithm 2 with the introduction of an additional scaling factor η in $u^{(t)}(s_t)$ term.

3.3. Nesterov's momentum for TD(0)

We use the second-order ODE stated in Muehlebach and Jordan [2019] as our starting point for obtaining the Nesterov TD(0) counterpart. Muehlebach and Jordan [2019] provide

Nesterov's accelerated method for SGD without relying on a vanishing step size argument while showing that the curvature-dependent damping term is responsible for acceleration. From Muehlebach and Jordan [2019], we know that the Nesterov accelerated gradient descent method results from a semi-implicit (symplectic) Euler discretization of the following ODE:

$$\ddot{x}(t) + 2d\dot{x}(t) + \frac{1}{L\gamma^2}\nabla f(x(t) + \beta\dot{x}(t)) = 0 \ , \tag{3.3.1}$$

where γ is constant and $d := \frac{1}{\sqrt{\kappa}+1}\frac{1}{\gamma}$ and $\beta = \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\frac{1}{\gamma}$. Specifically, $2d + \beta = 1$. Note that changing the value of γ amounts to rescaling the solutions to the ODE (3.3.1) in time.

Following a similar trajectory, we augment the TD(0) equation to give us an ODE which is the SA counterpart to (3.3.1). We define the ODE as:

$$\ddot{\mathbf{w}}(t) + 2d(t)\dot{\mathbf{w}}(t) = h(\mathbf{w}(t) + \beta(t)\dot{\mathbf{w}}(t)) = \mathbf{b} - A[\mathbf{w}(t) + \beta(t)\dot{\mathbf{w}}(t)] \ , \tag{3.3.2}$$

where $d(t)$ and $\beta(t)$ are akin to damping parameters, and $2d + \beta = 1$. Note that in our case, $d(t)$ and $\beta(t)$ are dependent on time. This is our proposed limiting ODE for Nesterov's acceleration in TD(0).

We will now perform: **(1)** Explicit Euler, and **(2)** Symplectic Euler discretization for ODE (3.3.2) to get the modified update equations and the corresponding algorithms for TD(0) with Nesterov's acceleration. Similar to how we obtained the algorithms for Polyak TD(0) ODE, we will discretize our second-order ODE (3.3.2) by translating our second-order ODE into a system of two first-order ODEs. Let $\dot{\mathbf{w}}(t) = \mathbf{z}(t)$. We can therefore express (3.3.2) as a system of two first-order ODEs as:

$$\begin{aligned}\dot{\mathbf{w}}(t) &= \mathbf{z}(t) \\ \dot{\mathbf{z}}(t) = \ddot{\mathbf{w}}(t) &= h[\mathbf{w}(t) + \beta(t)\mathbf{z}(t)] - 2d(t)\mathbf{z}(t) \ .\end{aligned}$$

We will now write the function h in terms of A and $\mathbf{b}$ from (3.1.4). We can therefore rewrite the above equations as:

$$\begin{aligned}\dot{\mathbf{w}}(t) &= \mathbf{z}(t) \\ \dot{\mathbf{z}}(t) = \ddot{\mathbf{w}}(t) &= \mathbf{b} - A\mathbf{w}(t) - [2d(t)I + \beta(t)A]\mathbf{z}(t) \ .\end{aligned} \tag{3.3.3}$$

3.3.1. Explicit Euler discretization

We will now discretize (3.3.3) using the explicit Euler scheme discussed in 2.7.1. Doing so, we get the following equations:

$$\begin{aligned}\mathbf{w}_{k+1} &= \mathbf{w}_k + \eta\mathbf{z}_k \\ \mathbf{z}_{k+1} &= \mathbf{z}_k - \eta[2d_kI + \beta_kA]\mathbf{z}_k + \eta h(\mathbf{w}_k) \ .\end{aligned} \tag{3.3.4}$$

Here, η is the learning rate. Note that since we are working on TD(0) with linear function approximation, we have $v^{(t)}(s_t) = \boldsymbol{\phi}^T(s_t)\mathbf{w}_t$ and $u^{(t)}(s_t) = \boldsymbol{\phi}^T(s_t)\mathbf{z}_t$. This gives us the

following update equations for Nesterov TD(0) with explicit Euler discretization:

$$v^{(t+1)}(s_t) = v^{(t)}(s_t) + \eta u^{(t)}(s_t)$$
$$u^{(t+1)}(s_t) = u^{(t)}(s_t) - \eta 2 d_t u^{(t)}(s_t) + \eta \left[\left(r_t + \gamma v^{(t)}(s_{t+1}) - v^{(t)}(s_t) + \beta_t \left(\gamma u^{(t)}(s_{t+1}) - u^{(t)}(s_t) \right) \right) \right] .$$

We can also rearrange the terms to write the update equation as:

$$\begin{aligned} v^{(t+1)}(s_t) &= v^{(t)}(s_t) + \eta u^{(t)}(s_t) \\ u^{(t+1)}(s_t) &= u^{(t)}(s_t) - \eta 2 d_t u^{(t)}(s_t) + \eta \beta_t \left(\gamma u^{(t)}(s_{t+1}) - u^{(t)}(s_t) \right) \\ &\quad + \eta \left(r^t + \gamma v^{(t)}(s_{t+1}) - v^{(t)}(s_t) \right) . \end{aligned} \tag{3.3.5}$$

Using the update equations given by (3.3.5), we get the Algorithm (3) for Nesterov TD(0) with explicit Euler discretization.

Algorithm 3: Nesterov TD(0) with Explicit Euler discretization

Input: Policy π to be evaluated, Set $\eta, \beta_k, d_k, \gamma$
Initialize $V(s)$ arbitrarily (e.g., $V(s) = 0 \quad \forall \quad s \in S^+$)
Repeat (for each episode):
    Initialize S; Set $V(s) = 0$, $k = 1$
    Repeat (for each step of the episode):
        $A \leftarrow$ action given by π for S
        Take action A, observe R, $S^{'}$ (new state)
        $V_{prev}(S) \leftarrow V(S)$
        $V_{prev}(S') \leftarrow V(S')$
        $V(S) \leftarrow V(S) + \eta U(S)$
        $U(S) \leftarrow U(S) - \eta 2 d_k U(S) + \eta \beta_k \left(\gamma U(S^{'}) - U(S) \right)$
        $\quad + \eta \left[R + \gamma V_{prev}(S^{'}) - V_{prev}(S) \right]$
        $k \leftarrow k + 1$
        $S \leftarrow S^{'}$
    until S **is terminal**

Note that we did not require an additional scaling term here as seen in Polyak TD(0). Hence, the update equations in Algorithm 3 directly follow the discrete-time counterpart seen in (3.3.5).

3.3.2. Symplectic Euler Discretization

Lastly, we will discretize (3.3.3) using the symplectic Euler scheme as discussed in 2.7.1. Again, note that we use the phrase *symplectic* to mean taking one explicit step and one implicit step for our system of continuous-time equations. Using this discretization, we get

the following discrete-time counterparts for (3.3.3)

$$\begin{aligned} \mathbf{w}_{k+1} &= \mathbf{w}_k + \eta \mathbf{z}_k \\ \mathbf{z}_{k+1} &= \mathbf{z}_k - \eta \left[2d_{k+1} I + \beta_{k+1} A\right] \mathbf{z}_{k+1} + \eta h(\mathbf{w}_{k+1}) \ . \end{aligned} \tag{3.3.6}$$

Note we can further simplify this equation by clubbing terms with $\mathbf{z}_{k+1}$ as:

$$\begin{aligned} \mathbf{w}_{k+1} &= \mathbf{w}_k + \eta \mathbf{z}_k \\ \mathbf{z}_{k+1} &= \left[(1 + 2\eta d_{k+1}) I + \eta \beta_{k+1} A\right]^{-1} \left[\mathbf{z}_k + \eta h(\mathbf{w}_{k+1})\right] \ . \end{aligned} \tag{3.3.7}$$

We will now substitute the $h(\mathbf{w}_{k+1})$ in the equation with values of A and $\mathbf{b}$ from (3.1.4). Also, note that $v^{(t)}(s_t) = \boldsymbol{\phi}^T(s_t)\mathbf{w}_t$ and $u^{(t)}(s_t) = \boldsymbol{\phi}^T(s_t)\mathbf{z}_t$. This gives us the following update equations for Nesterov TD(0) with Symplectic Euler discretization:

$$\begin{aligned} v^{(t+1)}(s_t) &= v^{(t)}(s_t) + \eta u^{(t)}(s_t) \\ u^{(t+1)}(s_t) &= u^{(t)}(s_t) - \eta[2d_{t+1} u^{(t+1)}(s_t)] + \eta\beta_{t+1}\left[\gamma u^{(t+1)}(s_{t+1}) - u^{(t+1)}(s_t)\right] \\ &\qquad + \eta\left[r_t + \gamma v^{(t+1)}(s_{t+1}) - v^{(t+1)}(s_t)\right] \ . \end{aligned} \tag{3.3.8}$$

The algorithm for Nesterov TD(0) with symplectic Euler discretization is given in Algorithm (4) as:

Algorithm 4: Nesterov TD(0) with Symplectic Euler discretization

Input: Policy π to be evaluated, Set η, β_k, γ
Initialize $\mathbf{w}_0 = \mathbf{0}$, $\mathbf{z}_0 = \mathbf{0}$
Repeat (for each episode):
    Initialize S; Compute $\phi(S)$; Set $\mathbf{w}_0 = \mathbf{0}$, $\mathbf{z}_0 = \mathbf{0}$, $k = 1$
    Repeat (for each step of the episode):
        $A \leftarrow$ action given by π for S
        Take action A, observe R, S' (new state)
        $\mathbf{w}_{k+1} = \mathbf{w}_k + \eta \mathbf{z}_k$
        $M = \left((1 + 2\eta d_{k+1}) I + \beta_{k+1}\eta(\phi(S) - \gamma\phi(S')\phi(S)^T\right)^{-1}$
        $\mathbf{z}_{k+1} = M\left[\mathbf{z}_k + \eta h(\mathbf{w}_{k+1})\right]$
        $V(S) \leftarrow \phi^T(S)\mathbf{w}_{k+1}$
        $k \leftarrow k + 1$
        $S \leftarrow S'$
    until S **is terminal**

Note that we chose to keep the update equations in the form of $\mathbf{w}_{k+1}$ as it is easier to perform computations of the inverse required for the symplectic Euler discretization in this form. Also, another important point to note is that the inverse of matrix $[(1 + 2\eta d_{k+1})I + \eta\beta_{k+1}A]$ always exists since A is assumed to be a positive-definite matrix.

This concludes the introduction of our discrete-time algorithms for the proposed Polyak and Nesterov TD(0) methods. Performing explicit Euler and symplectic Euler discretizations on both accelerated methods has given us four different discrete-time algorithms. In the next section, we will discuss and compare how the four discrete-time algorithms differ from each other.

3.4. Comparison between discrete-time algorithms of Polyak and Nesterov TD(0)

We start with summarizing the update equations of our Polyak and Nesterov TD(0) methods with explicit Euler and symplectic Euler discretizations respectively.

$$\textbf{TDPE} = \begin{cases} \mathbf{w}_{k+1} & = \mathbf{w}_k + \eta \mathbf{z}_k \\ \mathbf{z}_{k+1} & = \mathbf{z}_k - \eta\beta_k \mathbf{z}_k + \eta h(\mathbf{w}_k) \ . \end{cases} \tag{3.4.1}$$

$$\textbf{TDPS} = \begin{cases} \mathbf{w}_{k+1} & = \mathbf{w}_k + \eta \mathbf{z}_k \\ \mathbf{z}_{k+1} & = \frac{1}{(1+\eta\beta_{k+1})} \left[\mathbf{z}_k + \eta h(\mathbf{w}_{k+1})\right] \ . \end{cases} \tag{3.4.2}$$

$$\textbf{TDNE} = \begin{cases} \mathbf{w}_{k+1} & = \mathbf{w}_k + \eta \mathbf{z}_k \\ \mathbf{z}_{k+1} & = \mathbf{z}_k - \eta[2d_k I + \beta_k A]\mathbf{z}_k + \eta h(\mathbf{w}_k) \ . \end{cases} \tag{3.4.3}$$

$$\textbf{TDNS} = \begin{cases} \mathbf{w}_{k+1} & = \mathbf{w}_k + \eta \mathbf{z}_k \\ \mathbf{z}_{k+1} & = \left[(1+2\eta d_{k+1})I + \eta\beta_{k+1}A\right]^{-1} \left[\mathbf{z}_k + \eta h(\mathbf{w}_{k+1})\right] \ . \end{cases} \tag{3.4.4}$$

To be concise, we use the following acronyms for our proposed algorithms:

- **TDPE:** TD(0) Polyak with Explicit Euler discretization
- **TDPS:** TD(0) Polyak with Symplectic Euler discretization
- **TDNE:** TD(0) Nesterov with Explicit Euler discretization
- **TDNS**: TD(0) Nesterov with Symplectic Euler discretization

3.4.1. Polyak v/s Nesterov

From the above update equations, we can clearly see that the Nesterov versions are computationally more expensive than the Polyak versions of TD(0). TDNE has an additional matrix-vector ($\eta\beta_k A\mathbf{z}_k$) to perform when compared to the TDPE. On the other hand, TDNE has an entire matrix inverse to compute ($[(1+2\eta d_{k+1})I + \eta\beta_{k+1}A]^{-1}$)).

Also, note that the TDNS is computationally more expensive than the TDPS because of the presence of a matrix inverse term in the update equation.

3.4.2. Explicit Euler v/s Symplectic Euler

Looking at the above update equations, we see that the symplectic Euler versions of both Polyak and Nesterov TD(0) use h evaluated at the updated value of $\mathbf{w}$ at time step $k+1$, i.e., $h(\mathbf{w}_{k+1})$. This corresponds to performing the explicit Euler method on one equation and the implicit Euler method for discretization on the other equation as discussed in 2.7.1.

In general, symplectic Euler based discretizations are computationally intensive. This is more evident when looking at Nesterov TD(0). TDNS requires the computation of a matrix inverse.

Hence, we can see that the most computationally intensive methods out of all the four discrete-time algorithm is TDNS with the addition of the computational complexity of both Nesterov's momentum and the symplectic Euler discretization.

Concluding Remarks: In this chapter, we designed the counterparts to Polyak's Heavy Ball and Nesterov's acceleration in TD(0) which were discretized using two first-order Euler methods, explicit and symplectic, to give us four discrete-time accelerated TD(0) algorithms.

In the next chapter, we will give the convergence rates of the underlying dynamical system behind both Polyak and Nesterov TD(0). With the convergence analysis, we would also get the analytical values of the hyperparameters corresponding to the damping parameters, $\beta(t)$ and $d(t)$ to obtain the said accelerated convergence rate.

Chapter 4

Conservation Laws and Convergence Rates for Accelerated Temporal Difference Methods

This section aims to provide a framework to easily calculate convergence rates for accelerated methods in SA. Our work hinges upon Suh et al. [2022] and we now discuss their work in detail. Furthermore, we provide a proof sketch for calculating rates for TD methods.

The main goal of Suh et al. [2022] is to develop a unifying framework to recover existing continuous-time analysis for AGD. While acceleration has significantly sped-up first-order optimization methods, understanding the principles behind their success has remained shrouded in mystery. Ideas from various disciplines, like control theory and mathematical physics, were used to explain the phenomenon by looking at the physical interpretations of the underlying continuous-time system dynamics. However, these analyses often rely on establishing non-increasing energy functions (Lyapunov function) with unclear origins, often arrived at through a tedious trial-and-error process. Suh et al. [2022] provides a way for us to derive these energy functions naturally from the dynamics of the system in the form of conservation laws, which are analogous to the conservation of energy in physics.

We will now give the general proof sketch for finding conservation laws for accelerated TD methods which will then be employed to find conservation laws and corresponding convergence rates for Polyak TD(0) and Nesterov TD(0).

4.1. General Proof Sketch

We start by discussing the basics of deriving conserved quantities from a given ODE. Consider an ODE of the form:

$$\dot{\mathbf{m}}(t) + \mathbf{n}(t) = 0 \quad \text{which holds for all} \quad t > 0 \ , \tag{4.1.1}$$

where $\mathbf{m} : (0, \infty) \to \mathbb{R}$ is differentiable and $\mathbf{n} : (0, \infty) \to \mathbb{R}$ is integrable. Since the expression $\dot{\mathbf{m}}(t) + \mathbf{n}(t)$ equates to 0, integrating this expression over time would give us a

constant quantity. Formally,

$$\frac{d}{dt}\left(\mathbf{m}(t) + \int_{t_0}^{t} \mathbf{n}(s)ds\right) = \dot{\mathbf{m}}(t) + \mathbf{n}(t) = 0 \ ,$$

which implies that $\mathbf{m}(t) + \int_{t_0}^{t} \mathbf{n}(s)ds$ is constant with respect to time. This is therefore a conserved quantity. We can then get our conservation law to be:

$$E = \mathbf{m}(t_0) = \mathbf{m}(t) + \int_{t_0}^{t} \mathbf{n}(s)ds$$

$$E = \lim_{t_0 \to 0} \mathbf{m}(t_0) = \mathbf{m}(t) + \int_{0}^{t} \mathbf{n}(s)ds \ , \quad \text{if} \quad \lim_{t_0 \to 0} \mathbf{m}(t_0) \quad \text{exists} \ .$$

E here is independent of time. Note that a given ODE can exhibit several different conservation laws based on the coordinate system used. Here, since our focus is to derive the associated convergence rates for the system, the dilated coordinate system is chosen to obtain a particular conservation law from which extracting the rates for our dynamical system is pretty straightforward.

Consider the general form of SA ODE (or the ODE of TD(0) with linear function approximation):

$$\ddot{\mathbf{w}} + c\dot{\mathbf{w}} = h(\mathbf{w}) = b - A\mathbf{w} \ . \tag{4.1.2}$$

Note that our goal here is to convert (4.1.2) into the form given by (4.1.1) to derive conservation law for ODE of our interest. Consider a dilated coordinate system $\mathbf{y} = e^{\psi(t)}(\mathbf{w} - \mathbf{w}_\star)$. We can then write (4.1.2) in the dilated system as:

$$e^{-\psi(t)}\ddot{\mathbf{y}} + (c - 2\dot{\psi}(t))e^{-\mu(t)}\dot{\mathbf{y}} - (\ddot{\psi}(t) - \dot{\psi}^2(t) + c\dot{\psi}(t))e^{-\psi(t)}\mathbf{y} = h(e^{-\psi(t)}\mathbf{y} + \mathbf{w}_\star) \ .$$

Let $g(\mathbf{y},t) = -(\ddot{\psi}(t) - \dot{\psi}^2(t) + c\dot{\psi}(t))e^{-\psi(t)}\mathbf{y} - h(e^{-\psi(t)}\mathbf{y} + \mathbf{w}_\star))$.
We can therefore rewrite the equation in terms of $g(\mathbf{y},t)$ as:

$$e^{-\psi(t)}\ddot{\mathbf{y}} + (c - 2\dot{\psi}(t))e^{-\psi(t)}\dot{\mathbf{y}} + g(\mathbf{y},t) = 0 \ . \tag{4.1.3}$$

To write this equation as $\dot{A}(t) + B(t) = 0$, an integral step to perform is to take the inner product of (4.1.3) with $\dot{\mathbf{y}}$:

$$< e^{-\psi(t)}\ddot{\mathbf{y}} + (c - 2\dot{\psi}(t))e^{-\psi(t)}\dot{\mathbf{y}} + g(\mathbf{y},t), \dot{\mathbf{y}} >= 0$$

$$e^{-\psi(t)} < \ddot{\mathbf{y}}, \dot{\mathbf{y}} > + (c - 2\dot{\psi}(t))e^{-\psi(t)} < \dot{\mathbf{y}}, \dot{\mathbf{y}} > + < g(\mathbf{y},t), \dot{\mathbf{y}} >= 0 \ . \tag{4.1.4}$$

We can combine and simplify the first two terms of this expression using the following relation:

$$\frac{d}{dt}\left[\frac{e^{-\psi(t)}}{2}||\dot{\mathbf{y}}||^2\right] = e^{-\psi(t)} < \dot{\mathbf{y}}, \ddot{\mathbf{y}} > - \frac{\dot{\psi}(t)e^{-\psi(t)}}{2}||\dot{\mathbf{y}}||^2 \ .$$

Substituting this in (4.1.4), we get:

$$\frac{d}{dt}\left[\frac{e^{-\psi(t)}}{2}||\dot{\mathbf{y}}||^2\right] + \left(c - \frac{3\dot{\psi}(t)}{2}\right)e^{-\psi(t)} < \dot{\mathbf{y}}, \dot{\mathbf{y}} > + < g(\mathbf{y},t), \dot{\mathbf{y}} >= 0 \ . \tag{4.1.5}$$

Let us define $U(\mathbf{y},t) = \int_{\mathbf{y}} g(\mathbf{y},t)d\mathbf{y}$. We will discuss the validity of taking this integral in the section following this proof sketch. Note that giving the curve $\mathbf{y}(t)$ as the first input to U gives us $U(\mathbf{y}(t),t)$, which is a function solely dependent on t. Taking the total derivative of $U(\mathbf{y}(t),t)$ with respect to t and applying the chain rule of vector calculus, we get:

$$\frac{d}{dt}U(\mathbf{y}(t),t) =< g(\mathbf{y}(t),t), \dot{\mathbf{y}} > + \frac{\partial}{\partial t}U(\mathbf{y}(t),t) \ .$$

Here, $\frac{\partial}{\partial t}U(\mathbf{y}(t),t)$ means taking the partial derivative of $U(\mathbf{y},t)$ with respect to t and then plugging in $\mathbf{y} = \mathbf{y}(t)$.
The subterm $< g(\mathbf{y},t), \dot{\mathbf{y}} >$ can now be simplified with this relation as:

$$< g(\mathbf{y}(t),t), \dot{\mathbf{y}} >= \frac{d}{dt}U(\mathbf{y}(t),t) - \frac{\partial}{\partial t}U(\mathbf{y}(t),t) \ . \tag{4.1.6}$$

Substituting (4.1.6) in (4.1.5), we get:

$$\frac{d}{dt}\left[\frac{e^{-\psi(t)}}{2}||\dot{\mathbf{y}}||^2\right] + \left(c - \frac{3\dot{\psi}(t)}{2}\right)e^{-\psi(t)} < \dot{\mathbf{y}}, \dot{\mathbf{y}} > + \frac{d}{dt}U(\mathbf{y}(t),t) - \frac{\partial}{\partial t}U(\mathbf{y}(t),t) = 0$$

$$\frac{d}{dt}\left[\frac{e^{-\psi(t)}}{2}||\dot{\mathbf{y}}||^2 + U(\mathbf{y}(t),t)\right] + \left(c - \frac{3\dot{\psi}(t)}{2}\right)e^{-\psi(t)}||\dot{\mathbf{y}}||^2 - \frac{\partial}{\partial t}U(\mathbf{y}(t),t) = 0 \ . \tag{4.1.7}$$

Note we can now see (4.1.7) as our desired ODE $\dot{A}(t) + B(t) = 0$, where

$$A(t) = \frac{e^{-\psi(t)}}{2}||\dot{\mathbf{y}}||^2 + U(\mathbf{y}(t),t) \quad \text{and} \quad B(t) = \left(c - \frac{3\dot{\psi}(t)}{2}\right)e^{-\psi(t)}||\dot{\mathbf{y}}||^2 - \frac{\partial}{\partial t}U(\mathbf{y}(t),t) \ .$$

For $0 < t < t_0 < \infty$, integrating (4.1.7) from t_0 to t, we get:

$$\begin{aligned} E &= \frac{e^{-\psi(t_0)}}{2}||\dot{\mathbf{y}}(t_0)||^2 + U(\mathbf{y}(t_0),t_0) \\ &= \frac{e^{-\psi(t)}}{2}||\dot{\mathbf{y}}(t)||^2 + U(\mathbf{y}(t),t) + \int_{t_0}^{t}\left(c - \frac{3\dot{\psi}(s)}{2}\right)e^{-\psi(s)}||\dot{\mathbf{y}}(s)||^2 ds - \int_{t_0}^{t}\frac{\partial}{\partial s}U(\mathbf{y}(s),s)ds \ . \end{aligned} \tag{4.1.8}$$

This is our general form of the conservation law. Note that if $\mu(t) = 1$ and $U(\mathbf{y},t) = U(\mathbf{y})$, then this conservation law transforms into the usual conservation of energy in physics. Analogous to the conservation of energy in physics, we can say that the E comprises of the following terms: $(1/2)||\dot{\mathbf{y}}||^2$ which corresponds to the kinetic energy; $U(\mathbf{y},t)$ is the potential energy; $\int_{t_0}^{t} c||\dot{\mathbf{y}}||^2$ corresponds to the energy dissipated as heat due to friction, and the fourth term vanishes as potential U is independent of time (Suh et al. [2022]).

Validity of the operation $U(\mathbf{y},t) = \int_Y g(\mathbf{y},t)d\mathbf{y}$ It has been long established that TD is not a Gradient Descent (GD) method. Simply put, the TD updates cannot be written as the gradient of some function. Since $g(\mathbf{y},t)$ includes $h(W,t)$ which is the TD objective, a common critique of performing the operation $U(\mathbf{y},t) = \int_{\mathbf{y}} g(\mathbf{y},t)d\mathbf{y}$ can be that taking this step could mean assuming that there exists a function $U(\mathbf{y},t)$ taking the gradient of which can give us the TD objective. The purpose of this section is to explain that performing this

operation is valid and does not go against the fact that TD is not gradient descent. Consider the TD objective given by

$$\begin{aligned} h(\mathbf{w}(t),t) &= \mathbb{E}_{x_d}[r_t\phi_t(s_t)] - \mathbb{E}_{x_d}[\boldsymbol{\phi}_t(s_t)(\boldsymbol{\phi}_t(s_t) - \gamma\boldsymbol{\phi}_t(s_{t+1}))^T]\mathbf{w}(t) \\ &= \Phi^T X R_d - \Phi^T X(I - \gamma P_d)\Phi\mathbf{w} \ . \end{aligned}$$

Taking derivative of $h(\mathbf{w}(t),t)$ with respect to w_i and w_j, we get:

$$\begin{aligned} \frac{\partial h(\mathbf{w},t)}{\partial w_i} &= \phi_i^T X R_d - \phi_i^T X(I - \gamma P_d)\Phi\mathbf{w} \ , \\ \frac{\partial h(\mathbf{w},t)}{\partial w_j} &= \phi_j^T X R_d - \phi_j^T X(I - \gamma P_d)\Phi\mathbf{w} \ . \end{aligned}$$

Taking the cross derivative of $h(\mathbf{w},t)$, we get:

$$\begin{aligned} \frac{\partial^2 h(\mathbf{w},t)}{\partial w_i \partial w_j} &= \phi_i^T X(I - \gamma P_d)\phi_j \ , \\ \frac{\partial^2 h(\mathbf{w},t)}{\partial w_j \partial w_i} &= \phi_j^T X(I - \gamma P_d)\phi_i \ . \end{aligned}$$

Since the second partial derivatives are not symmetric, according to Schwarz theorem, the second partial derivative is not continuous. Hence, this shows that TD method is not a gradient descent algorithm. However, this does not dispute the existence of the terms $\frac{\partial h(\mathbf{w},t)}{\partial w_i}$ and $\frac{\partial h(\mathbf{w},t)}{\partial w_j}$. While these first partial derivatives exist, they do not give a descent direction as the second partial derivatives are not symmetric in nature. Hence, $U(\mathbf{y},t) = \int_{\mathbf{y}} g(\mathbf{y},t)d\mathbf{y}$ is a valid operation to perform and does not imply that TD is a gradient descent method.

4.2. Preliminaries

Before starting our formal proof, we will define the general notation and definitions that will be revisited throughout the proof. For the sake of simplicity, we will focus our attention on finite-dimensional real-valued vector spaces. Consider an n-dimensional real vector space that is endowed with a norm $||.||$. We stick with the standard notation for Euclidean spaces, where $\langle . \rangle$ denotes the inner product, and $||.|| = ||.||_2$ is the Euclidean norm.

We consider the problem of policy evaluation for TD(0) with linear function approximation such that $v = \boldsymbol{\phi}^T\mathbf{w}$ where $\boldsymbol{\Phi} \in \mathbb{R}^{|S|\times k}$ is the concatentation of all feature vectors as rows and $\boldsymbol{\phi}(s)$ corresponds to the s^{th} feature vector. The update rule is therefore given as:

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \eta_t[\mathbf{r}_t + \gamma\boldsymbol{\phi}^T(s_{t+1})\mathbf{w}^{(t)} - \boldsymbol{\phi}^T(s_t)\mathbf{w}^{(t)}]\boldsymbol{\phi}_t \ .$$

We will be viewing this equation as a stochastic approximation algorithm used for finding the fixed points of the equation $h(\mathbf{w}) = 0$ where $\mathbf{w}_\star$ is the solution to this equation such that $h(\mathbf{w}_\star) = 0$. For this proof, we will solely be working with the underlying limiting ODE that

tracks the average of these iterates, which can be written as:

$$\dot{\mathbf{w}}(t) = h(\mathbf{w}(t)) = \mathbf{b} - A\mathbf{w}(t) \ , \tag{4.2.1}$$

$$\text{where} \quad A = \mathbb{E}_{x_d}[\boldsymbol{\phi}_t(s_t)\Big(\boldsymbol{\phi}_t(s_t) - \gamma\boldsymbol{\phi}_t(s_{t+1})\Big)^T] \quad \text{and} \quad \mathbf{b} = \mathbb{E}_{x_d}[\mathbf{r}_t\boldsymbol{\phi}_t(s_t)] \ .$$

We assume matrix A to be positive definite. Equation (4.2.1) is also known to converge to $\mathbf{w}_\star := A^{-1}\mathbf{b}$.

We will now outline some useful facts and definitions that would be used for both conservation law proofs.

Definition 4.2.1. *An operator $H : \mathbb{R}^n \to \mathbb{R}^n$ is said to be a monotone operator if:*

$$\langle H\boldsymbol{x} - H\boldsymbol{y}, \boldsymbol{x} - \boldsymbol{y}\rangle > 0 \quad \forall \quad \boldsymbol{x} \neq \boldsymbol{y} \ .$$

Definition 4.2.2 (Suh et al. [2022])**.** *Let $A : (0, \infty) \to \mathbb{R}$ be differentiable and $B : (0, \infty) \to \mathbb{R}$ be integrable. Suppose*

$$\dot{A}(t) + B(t) = 0$$

holds for all $t > 0$. Then, for $0 < t_0 < t < \infty$, integrating from t_0 to t gives us the conservation law:

$$E \equiv A(t_0) = A(t) + \int_{t_0}^{t} B(s)ds \ .$$

Also, if the $\lim_{t_0 \to 0} A(t_0)$ exists, then we have

$$E \equiv \lim_{t_0 \to 0} A(t_0) = A(t) + \int_{0}^{t} B(s)ds \ .$$

Note that this is a general definition and methodology for finding conservation laws for any dynamical system that does not apply any restricting conditions.

Definition 4.2.3 (Suh et al. [2022])**.** *Consider a function $U(\boldsymbol{y}, t)$ with variables $\boldsymbol{y} = (y_1, \cdots y_n) \in \mathbb{R}^n$ and $t \in \mathbb{R}$ defined as*

$$U(\boldsymbol{y}, t) := \int_{\boldsymbol{y}} g(Y,t)d\boldsymbol{y} \quad \in \quad \mathbb{R}^n \ ,$$

where $g(\boldsymbol{y}, t) : \mathbb{R}^n \to \mathbb{R}^n$ When $\boldsymbol{y}(t)$ is differentiable, the chain rule gives us

$$\frac{d}{dt}U(\boldsymbol{y},t) = \langle g(\boldsymbol{y}, t), \dot{\boldsymbol{y}}\rangle - \frac{\partial}{\partial t}U(\boldsymbol{y}(t), t) \ .$$

Note that taking the total derivative $\frac{d}{dt}$ corresponds to viewing $\mathbf{w}(t)$ as a curve dependent on t whereas taking the partial derivative $\frac{\partial}{\partial t}$ corresponds to viewing $\mathbf{w}$ as an input to U that is independent of t.

4.3. Conservation law for Polyak TD(0)

Consider the following ODE for accelerated TD(0) with Polyak's momentum:

$$\ddot{\mathbf{w}}(t) + \frac{\beta}{t}\dot{\mathbf{w}}(t) = h(\mathbf{w}(t)) = b - A\mathbf{w}(t) \ , \tag{4.3.1}$$

$$\text{where} \quad A = \mathbb{E}_{\mathbf{x}}[\boldsymbol{\phi}_t(s_t)(\boldsymbol{\phi}_t(s_t) - \gamma\boldsymbol{\phi}_t(s_t))] \quad \text{and} \quad b = \mathbb{E}_{\mathbf{x}}[\mathbf{r}_t\boldsymbol{\phi}_t(s_t)] \ .$$

Note that we are trying to solve to find the fixed-point for the function $h(\mathbf{w})$ where $\mathbf{w} = \mathbf{w}_\star$ is the solution to this equation such that $h(\mathbf{w}_\star) = 0$.

Consider the dilated coordinate system $\mathbf{y} = t^\alpha(\mathbf{y} - \mathbf{y}_\star)$ for some $\alpha \in \mathbb{R}$ to be defined later. We now express the ODE (4.3.1) in the $\mathbf{y}$- coordinate system as:

$$\begin{aligned}
\mathbf{y} &= t^\alpha(\mathbf{w} - \mathbf{w}_\star) \ , \\
\dot{\mathbf{y}} &= \alpha t^{\alpha-1}(\mathbf{w} - \mathbf{w}_\star) + t^\alpha\dot{\mathbf{w}} \ , \\
\ddot{\mathbf{y}} &= \alpha(\alpha-1)t^{\alpha-2}(\mathbf{w} - \mathbf{w}_\star) + 2\alpha t^{\alpha-1}\dot{\mathbf{w}} + t^\alpha\ddot{\mathbf{w}} \ .
\end{aligned}$$

Rewriting $\mathbf{w}, \dot{\mathbf{w}}, \ddot{\mathbf{w}}$ in terms of $\mathbf{y}, \dot{\mathbf{y}}, \ddot{\mathbf{y}}$, we get:

$$\begin{aligned}
\mathbf{w} &= t^{-\alpha}\mathbf{y} + \mathbf{w}_\star \ , \\
\dot{\mathbf{w}} &= t^{-\alpha}\dot{\mathbf{y}} - \alpha t^{-(\alpha+1)}\mathbf{y} \ , \\
\ddot{\mathbf{w}} &= t^{-\alpha}\ddot{\mathbf{y}} - 2\alpha t^{-(\alpha+1)}\dot{\mathbf{y}} + \alpha(\alpha+1)t^{-(\alpha+2)}\mathbf{y} \ .
\end{aligned}$$

Plugging $\mathbf{w}, \dot{\mathbf{w}}, \ddot{\mathbf{w}}$ in ODE (4.3.1), we get:

$$t^{-\alpha}\ddot{\mathbf{y}} - 2\alpha t^{-(\alpha+1)}\dot{\mathbf{y}} + \frac{\beta}{t}\left[t^{-\alpha}\dot{\mathbf{y}} - \alpha t^{-(\alpha+1)}\mathbf{y}\right] - h(t^{-\alpha}\mathbf{y} + \mathbf{w}_\star) = 0 \ .$$

$$t^{-\alpha}\ddot{\mathbf{y}} + (\beta - 2\alpha)t^{-(\alpha+1)}\dot{\mathbf{y}} + \alpha(\alpha - \beta + 1)t^{-(\alpha+2)}\mathbf{y} - h(t^{-\alpha}\mathbf{y} + \mathbf{w}_\star) = 0 \ .$$

Let us define $g(\mathbf{y},t) = \alpha(\alpha - \beta + 1)t^{-(\alpha+2)}\mathbf{y} - h(t^{-\alpha}\mathbf{y} + \mathbf{w}_\star)$. We can write the ODE as:

$$t^{-\alpha}\ddot{\mathbf{y}} + (\beta - 2\alpha)t^{-(\alpha+1)}\dot{\mathbf{y}} + g(\mathbf{y},t) = 0 \ . \tag{4.3.2}$$

Lemma 4.3.1. *Let $\boldsymbol{y}$ evolve according to* (4.3.2)*. Denote $U(\boldsymbol{y},t) = \int_{\boldsymbol{y}} g(\boldsymbol{y},t)d\boldsymbol{y}$ where $g(\boldsymbol{y},t) = \alpha(\alpha - \beta + 1)t^{-(\alpha+2)}\boldsymbol{y} - h(t^{-\alpha}\boldsymbol{y} + \boldsymbol{w}_\star)$. Then, $\forall t \geq 0$, we have*

$$U(\boldsymbol{y},t) = \alpha(\alpha - \beta + 1)t^{-(\alpha+2)}\frac{\boldsymbol{y}^T\boldsymbol{y}}{2} - b^T\boldsymbol{y} + t^{-\alpha}\boldsymbol{y}^T A\boldsymbol{y} + \boldsymbol{w}_\star^T A^T\boldsymbol{y} \ .$$

Proof. $U(\mathbf{y},t)$ is defined as $\int_{\mathbf{y}} g(\mathbf{y},t)d\mathbf{y}$. Performing this operation gives us:

$$\begin{aligned}
U(\mathbf{y},t) &= \int_{\mathbf{y}} g(\mathbf{y},t)d\mathbf{y} \\
&= \int_{\mathbf{y}} [\alpha(\alpha - \beta + 1)t^{-(\alpha+2)}\mathbf{y} - h(t^{-\alpha}\mathbf{y} + \mathbf{w}_\star)]d\mathbf{y} \\
&= \alpha(\alpha - \beta + 1)t^{-(\alpha+2)}\frac{\mathbf{y}^T\mathbf{y}}{2} - b^T\mathbf{y} + t^{-\alpha}\mathbf{y}^T A\mathbf{y} + \mathbf{w}_\star^T A^T\mathbf{y} \ .
\end{aligned}$$

□

Note that as we have discussed before in Section (4.1), performing the operation $U(\mathbf{y},t) = \int_{\mathbf{y}} g(\mathbf{y},t)d\mathbf{y}$ does not imply that we are assuming that TD is a gradient descent method. We will now use Lemma (4.3.1) to obtain a conservation law for Polyak TD(0).

Proposition 4.3.2. *Let $\boldsymbol{w}$ evolve according to (4.3.1), for an arbitrary but fixed $\boldsymbol{w}_0 \in \mathbb{R}^n$ such that $\boldsymbol{w}(0) = \boldsymbol{w}_0$ and $\dot{\boldsymbol{w}}(0) = 0$. Then, $\forall t \geq 0$, the conservation law can be given as:*

$$\begin{aligned} E =& \frac{||\alpha(\boldsymbol{w} - \boldsymbol{w}_\star + t\dot{\boldsymbol{w}}||^2)}{2t^{\alpha-2}} + \left(\frac{\alpha\beta - 2\alpha^2 + \alpha}{2t^{\alpha-2}}\right)||\boldsymbol{w} - \boldsymbol{w}_\star||^2 - t^\alpha\langle h(\boldsymbol{w}), \boldsymbol{w} - \boldsymbol{w}_\star\rangle \\ &+ \int_0^t \left(\beta - \frac{3\alpha}{2}\right) s^{\alpha-1}||\dot{\boldsymbol{w}}||^2 ds + \int_0^t \left[\left(\frac{\alpha^3 - 3\alpha^2 - \alpha^2\beta + 2\alpha\beta + 2\alpha}{2}\right) s^{\alpha-3}||\boldsymbol{w} - \boldsymbol{w}_\star||^2\right] ds \\ &+ \int_0^t \alpha s^{\alpha-1}(\boldsymbol{w} - \boldsymbol{w}_\star)^T A(\boldsymbol{w} - \boldsymbol{w}_\star) ds \ . \end{aligned}$$

Proof. Taking the inner product of (4.3.2) with $\dot{\mathbf{y}}$ gives:

$$\begin{aligned} \langle t^{-\alpha}\ddot{\mathbf{y}} + (\beta - 2\alpha)t^{-(\alpha+1)}\dot{\mathbf{y}} + g(\mathbf{y},t), \dot{\mathbf{y}}\rangle = 0 \ . \\ \langle t^{-\alpha}\ddot{\mathbf{y}}, \dot{\mathbf{y}}\rangle + \langle(\beta - 2\alpha)t^{-(\alpha+1)}\dot{\mathbf{y}}, \dot{\mathbf{y}}\rangle + \langle g(\mathbf{y},t), \dot{\mathbf{y}}\rangle = 0 \ . \end{aligned} \tag{4.3.3}$$

Now, we have,

$$\frac{d}{dt}\left(\frac{||\dot{\mathbf{y}}||^2}{2t^\alpha}\right) = < \frac{\ddot{\mathbf{y}}}{t^\alpha}, \dot{\mathbf{y}} > - \frac{\alpha||\dot{\mathbf{y}}||^2}{2t^{\alpha+1}} \ .$$

From Definition (4.2.3), we can say that:

$$< g(\mathbf{y},t), \dot{\mathbf{y}} > = \frac{d}{dt}U(\mathbf{y},t) - \frac{\partial}{\partial t}U(\mathbf{y}(t),t) \ .$$

Substituting these in (4.3.3), we get:

$$\frac{d}{dt}\left(\frac{||\dot{\mathbf{y}}||^2}{2t^\alpha} + U(\mathbf{y},t)\right) + \left(\beta - \frac{3\alpha}{2}\right)t^{-(\alpha+1)}||\dot{\mathbf{y}}||^2 - \frac{\partial}{\partial t}U(\mathbf{y},t) = 0 \ ,$$

$$\text{where} \quad A(t) = \frac{||\dot{\mathbf{y}}||^2}{2t^\alpha} + U(\mathbf{y},t) \quad \text{and} \quad B(t) = \left(\beta - \frac{3\alpha}{2}\right)t^{-(\alpha+1)}||\dot{\mathbf{y}}||^2 - \frac{\partial}{\partial t}U(\mathbf{y},t) \ .$$

Substituting $U(\mathbf{y},t)$ from Lemma (4.3.1), we will now find the partial derivative of $U(\mathbf{y},t)$ with respect to t:

$$\begin{aligned} \frac{\partial}{\partial t}U(\mathbf{y},t) &= \frac{\partial}{\partial t}\left[\alpha(\alpha - \beta + 1)t^{-(\alpha+2)}\frac{\mathbf{y}^T\mathbf{y}}{2} - b^T\mathbf{y} + t^{-\alpha}\mathbf{y}^T A\mathbf{y} + \mathbf{w}_\star^T A^T\mathbf{y}\right] \\ &= -\alpha(\alpha - \beta + 1)(\alpha + 2)t^{-(\alpha+3)}\frac{\mathbf{y}^T\mathbf{y}}{2} - \alpha t^{-(\alpha+1)}\mathbf{y}^T A\mathbf{y} \ . \end{aligned}$$

We will now substitute $\mathbf{y}(t) = t^\alpha(\mathbf{w} - \mathbf{w}_\star)$, we get:

$$\frac{\partial}{\partial t}U(\mathbf{y},t) = -\alpha(\alpha - \beta + 1)(\alpha + 2)t^{\alpha-3}\frac{||\mathbf{w} - \mathbf{w}_\star||^2}{2} - \alpha t^{\alpha-1}(\mathbf{w} - \mathbf{w}_\star)^T A(\mathbf{w} - \mathbf{w}_\star) \ .$$

Note that using Definition (4.2.2), we can write our conservation law as:

$$\begin{aligned}
E =& A(t) + \int_{t_0}^{t} B(s)ds \\
=& \frac{||\dot{\mathbf{y}}||^2}{2t^\alpha} + \alpha(\alpha-\beta+1)t^{-(\alpha+2)}\frac{\mathbf{y}^T\mathbf{y}}{2} - b^T\mathbf{y} + t^{-\alpha}\mathbf{y}^TA\mathbf{y} + \mathbf{w}_\star^TA^T\mathbf{y} \\
&+ \int_0^t \left(\beta - \frac{3\alpha}{2}\right) s^{-(\alpha+1)}||\dot{\mathbf{y}}||^2 ds \\
&+ \int_0^t \left(\alpha(\alpha-\beta+1)(\alpha+2)s^{\alpha-3}\frac{||\mathbf{w}-\mathbf{w}_\star||^2}{2} + \alpha s^{\alpha-1}(\mathbf{w}-\mathbf{w}_\star)^TA(\mathbf{w}-\mathbf{w}_\star)\right) \ .
\end{aligned}$$

Substituting $\mathbf{y} = t^\alpha(\mathbf{w}-\mathbf{w}_\star)$, we get:

$$\begin{aligned}
E =& \frac{t^\alpha||\alpha t^{-1}(\mathbf{w}-\mathbf{w}_\star) + t^\alpha\dot{\mathbf{w}}||^2}{2t^\alpha} + \alpha(\alpha-\beta+1)t^{\alpha-2}\frac{||\mathbf{w}-\mathbf{w}_\star||^2}{2} \\
&- t^\alpha b^T(\mathbf{w}-\mathbf{w}_\star) + t^\alpha(\mathbf{w}-\mathbf{w}_\star)^TA(\mathbf{w}-\mathbf{w}_\star) + t^\alpha\mathbf{w}_\star^TA^T(\mathbf{w}-\mathbf{w}_\star) \\
&+ \int_0^t \left(\left(\beta - \frac{3\alpha}{2}\right) s^{\alpha-1}||\alpha s^{-1}(\mathbf{w}-\mathbf{w}_\star) + \dot{\mathbf{w}}||^2\right) ds \\
&+ \int_0^t \left(\alpha(\alpha-\beta+1)(\alpha+2)s^{\alpha-3}\frac{||\mathbf{w}-\mathbf{w}_\star||^2}{2} + \alpha s^{\alpha-1}(\mathbf{w}-\mathbf{w}_\star)^TA(\mathbf{w}-\mathbf{w}_\star)\right) ds \ .
\end{aligned}$$

$$\begin{aligned}
\text{Let} \quad I =& \int_0^t \left(\left(\beta - \frac{3\alpha}{2}\right) s^{\alpha-1}||\alpha s^{-1}(\mathbf{w}-\mathbf{w}_\star) + \dot{\mathbf{w}}||^2\right) ds \\
&+ \int_0^t \left(\alpha(\alpha-\beta+1)(\alpha+2)s^{\alpha-3}\frac{||\mathbf{w}-\mathbf{w}_\star||^2}{2} + \alpha s^{\alpha-1}(\mathbf{w}-\mathbf{w}_\star)^TA(\mathbf{w}-\mathbf{w}_\star)\right) ds \ .
\end{aligned}$$

Solving for the integral, we get:

$$\begin{aligned}
I =& \left(\frac{2\alpha\beta - 3\alpha^2}{2}\right) t^{\alpha-2}||\mathbf{w}-\mathbf{w}_\star||^2 + \int_0^t \left(\beta - \frac{3\alpha}{2}\right) s^{\alpha-1}||\dot{\mathbf{w}}||^2 ds \\
&+ \int_0^t \left[\left(\frac{\alpha^3 - 3\alpha^2 - \alpha^2\beta + 2\alpha\beta + 2\alpha}{2}\right) s^{\alpha-3}||\mathbf{w}-\mathbf{w}_\star||^2 + \alpha s^{\alpha-1}(\mathbf{w}-\mathbf{w}_\star)^TA(\mathbf{w}-\mathbf{w}_\star)\right] ds \ .
\end{aligned}$$

We can now simplify and write our conservation law as:

$$\begin{aligned}
E =& \frac{t^\alpha||\alpha t^{-1}(\mathbf{w}-\mathbf{w}_\star) + \dot{\mathbf{w}}||^2}{2} + \left(\frac{\alpha\beta - 2\alpha^2 + \alpha}{2}\right) t^{\alpha-2}||\mathbf{w}-\mathbf{w}_\star||^2 - t^\alpha b^T(\mathbf{w}-\mathbf{w}_\star) \\
&+ t^\alpha(\mathbf{w}-\mathbf{w}_\star)^TA(\mathbf{w}-\mathbf{w}_\star) + t^\alpha\mathbf{w}_\star^TA^T(\mathbf{w}-\mathbf{w}_\star) + \int_0^t \left(\beta - \frac{3\alpha}{2}\right) s^{\alpha-1}||\dot{W}||^2 \\
&+ \int_0^t \left[\left(\frac{\alpha^3 - 3\alpha^2 - \alpha^2\beta + 2\alpha\beta + 2\alpha}{2}\right) s^{\alpha-3}||\mathbf{w}-\mathbf{w}_\star||^2 + \alpha s^{\alpha-1}(\mathbf{w}-\mathbf{w}_\star)^TA(\mathbf{w}-\mathbf{w}_\star)\right] ds \ .
\end{aligned}$$

Expressing quantities in terms of $h(\mathbf{w})$, we get:

$$\begin{aligned}E =&\frac{||\alpha(\mathbf{w}-\mathbf{w}_\star+t\dot{\mathbf{w}}||^2)}{2t^{\alpha-2}}+\left(\frac{\alpha\beta-2\alpha^2+\alpha}{2t^{\alpha-2}}\right)||\mathbf{w}-\mathbf{w}_\star||^2-t^\alpha\langle h(\mathbf{w}),\mathbf{w}-\mathbf{w}_\star\rangle\\&+\int_0^t\left(\beta-\frac{3\alpha}{2}\right)s^{\alpha-1}||\dot{\mathbf{w}}||^2ds+\int_0^t\left[\left(\frac{\alpha^3-3\alpha^2-\alpha^2\beta+2\alpha\beta+2\alpha}{2}\right)s^{\alpha-3}||\mathbf{w}-\mathbf{w}_\star||^2\right]ds\\&+\int_0^t\alpha s^{\alpha-1}(\mathbf{w}-\mathbf{w}_\star)^TA(\mathbf{w}-\mathbf{w}_\star)ds\ .\end{aligned}\tag{4.3.4}$$

□

Lemma 4.3.3. *Let $E_0=\lim_{t\to t_0}E$ denote the conservation law at the starting point t_0. Then,*

$$E_0=\frac{\alpha(2\alpha-\beta+1)}{2}t_0^{\alpha-2}||\boldsymbol{w}_0-\boldsymbol{w}_\star||^2-t_0^\alpha\langle h(\boldsymbol{w}_0),\boldsymbol{w}_0-\boldsymbol{w}_\star\rangle\ .$$

Proof. Using Defintion (4.2.2), E can be expressed at the initial start point t_0 as:

$$\begin{aligned}E_0&=\lim_{t\to t_0}A(t)\\&=\lim_{t\to t_0}\left(\frac{||\dot{\mathbf{y}}||^2}{2t^\alpha}+U(\mathbf{y},t)\right)\\&=\frac{t_0^{\alpha-2}||\alpha(\mathbf{w}-\mathbf{w}_\star)+t_0\dot{\mathbf{w}}||^2)}{2}+\frac{\alpha(\alpha-\beta+1)}{2}t_0^{\alpha-2}||\mathbf{w}-\mathbf{w}_\star||^2-t_0^\alpha b^T(\mathbf{w}-\mathbf{w}_\star)\\&\quad+t_0^\alpha(\mathbf{w}-\mathbf{w}_\star)^TA(\mathbf{w}-\mathbf{w}_\star)+t_0^\alpha\mathbf{w}_\star^TA^T(\mathbf{w}-\mathbf{w}_\star)\ .\end{aligned}$$

Simplifying the terms, we get

$$\begin{aligned}E_0=\frac{\alpha(2\alpha-\beta+1)}{2}t_0^{\alpha-2}||\mathbf{w}_0-\mathbf{w}_\star||^2-t_0^\alpha[&b^T(\mathbf{w}_0-\mathbf{w}_\star)+(\mathbf{w}_0-\mathbf{w}_\star)^TA(\mathbf{w}_0-\mathbf{w}_\star)\\&+\mathbf{w}_\star^TA^T(\mathbf{w}_0-\mathbf{w}_\star)]\end{aligned}$$

$$\begin{aligned}E_0=\frac{\alpha(2\alpha-\beta+1)}{2}t_0^{\alpha-2}||\mathbf{w}_0-\mathbf{w}_\star||^2-t_0^\alpha[&b^T(\mathbf{w}_0-\mathbf{w}_\star)+(\mathbf{w}_0-\mathbf{w}_\star)^TA(\mathbf{w}_0-\mathbf{w}_\star)\\&+\mathbf{w}_\star^TA^T(\mathbf{w}_0-\mathbf{w}_\star)]\end{aligned}$$

$$\begin{aligned}E_0&=\frac{\alpha(2\alpha-\beta+1)}{2}t_0^{\alpha-2}||\mathbf{w}_0-\mathbf{w}_\star||^2+t_0^\alpha[b^T(\mathbf{w}_0-\mathbf{w}_\star)+\mathbf{w}_0^TA(\mathbf{w}_0-\mathbf{w}_\star)]\\&=\frac{\alpha(2\alpha-\beta+1)}{2}t_0^{\alpha-2}||\mathbf{w}_0-\mathbf{w}_\star||^2+t_0^\alpha\langle b-A^T\mathbf{w}_0,\mathbf{w}_0-\mathbf{w}_\star\rangle\ .\end{aligned}$$

We can therefore write the conservation law as:

$$E_0=\frac{\alpha(2\alpha-\beta+1)}{2}t_0^{\alpha-2}||\mathbf{w}_0-\mathbf{w}_\star||^2-t_0^\alpha\langle h(\mathbf{w}_0),\mathbf{w}_0-\mathbf{w}_\star\rangle\ .\tag{4.3.5}$$

□

Since E remains conserved, we can therefore equate (4.3.4) and (4.3.5). Also, since $h(\mathbf{w}_\star)=0$, we can modify the equations and write them as:

$$\begin{aligned}
&\frac{\alpha(2\alpha-\beta+1)}{2}t_0^{\alpha-2}||\mathbf{w}_0-\mathbf{w}_\star||^2 - t_0^\alpha\langle h(\mathbf{w}_0)-h(\mathbf{w}_\star),\mathbf{w}_0-\mathbf{w}_\star\rangle \\
&\qquad = \left(\frac{\alpha\beta-2\alpha^2+\alpha}{2t^{\alpha-2}}\right)||\mathbf{w}-\mathbf{w}_\star||^2 - t^\alpha < h(\mathbf{w})-h(\mathbf{w}_\star),\mathbf{w}-\mathbf{w}_\star > \\
&\qquad\quad + \int_0^t \left(\beta-\frac{3\alpha}{2}\right)s^{\alpha-2}||\dot{\mathbf{w}}||^2 ds + \frac{||\alpha(\mathbf{w}-\mathbf{w}_\star)+t\dot{\mathbf{w}}||^2}{2t^{\alpha-2}} \\
&\qquad\quad + \int_0^t \left[\frac{\alpha^3-3\alpha^2-\alpha^2\beta+2\alpha\beta+2\alpha}{2s^{\alpha-3}}||\mathbf{w}-\mathbf{w}_\star||^2\right] ds \\
&\qquad\quad + \int_0^t \alpha s^{\alpha-1}(\mathbf{w}-\mathbf{w}_\star)^T A(\mathbf{w}-\mathbf{w}_\star) ds \ .
\end{aligned} \tag{4.3.6}$$

Lemma 4.3.4. *Let $h(\boldsymbol{w}) = b - A\boldsymbol{w}$ where A is positive definite. We can therefore say that:*

$$\langle h(\boldsymbol{w})-h(\boldsymbol{w}_\star),\boldsymbol{w}-\boldsymbol{w}_\star\rangle < 0$$

holds true for all $\boldsymbol{w}\neq\boldsymbol{w}_\star\in\mathbb{R}^n$.

Proof. Note that for a general affine function $f(\mathbf{x}) = A\mathbf{x}+b$, we can call it monotone iff $A+A^T \succeq 0$.. For monotone functions, we know from Definition (4.2.1) that $\langle \mathbf{x}-\mathbf{y}, f(\mathbf{x})-f(\mathbf{y})\rangle > 0 \quad \forall \mathbf{x}\neq\mathbf{y}$.

Here we have the affine function $h(\mathbf{w}) = b - A\mathbf{w}$ where A is positive definite and hence symmetric i.e., $A+A^T \succeq 0$. This makes $-A$ negative definite and hence, we can say that $-h(\mathbf{w})$ is a monotone function. Since $-h(\mathbf{w})$ is monotone, we can say that:

$$\langle h(\mathbf{w})-h(\mathbf{w}_\star),\mathbf{w}-\mathbf{w}_\star\rangle < 0 \ .$$

□

Using Lemma (4.3.4), we can show that the following terms are always positive:

$$\langle h(\mathbf{w}_0)-h(\mathbf{w}_\star),\mathbf{w}_0-\mathbf{w}_\star\rangle < 0 \implies -t_0^\alpha\langle h(\mathbf{w}_0)-h(\mathbf{w}_\star),\mathbf{w}_0-\mathbf{w}_\star\rangle > 0 \ .$$
$$\langle h(\mathbf{w})-h(\mathbf{w}_\star),\mathbf{w}-\mathbf{w}_\star\rangle < 0 \implies -t^\alpha\langle h(\mathbf{w})-h(\mathbf{w}_\star),\mathbf{w}-\mathbf{w}_\star\rangle > 0 \ .$$

We will now use this conservation law relation to derive the convergence rate for our accelerated TD methods. We can simplify and remove positive terms from the equation if the coefficients of those terms remain positive. This gives us the conditions needed for the ODE to be accelerated. The convergence rate depends on the choice of α. Since we observe t^α multiplied with the term $\langle h(\mathbf{w})-h(\mathbf{w}_\star),\mathbf{w}-\mathbf{w}_\star\rangle$, the choice of α determines our convergence rate. In anticipation for an accelerated rate of $\mathcal{O}(1/t^2)$, we take $\alpha = 2$.

Theorem 4.3.5. *Let $\boldsymbol{w}$ evolve according to* (4.3.1) *which solves for $h(\boldsymbol{w}) = \boldsymbol{b}-A\boldsymbol{w}$ where A is positive definite. Let $\boldsymbol{w}_\star$ be the minimizer for $h(\boldsymbol{w})$ such that $h(\boldsymbol{w}_\star)=0$. Let $\boldsymbol{w}(0)=\boldsymbol{w}_0\in\mathbb{R}^n$ be an arbitrary but fixed initial point, and $\dot{\boldsymbol{w}}_0 = 0$ denote the initial conditions. For the conservation law E and E_0 given by Proposition* (4.3.2) *and Lemma* (4.3.3) *respectively,*

setting $\alpha = 2$, we get the convergence rate expression for (4.3.1) *as:*

$$||\boldsymbol{w} - \boldsymbol{w}_\star||^2 \leq \frac{2}{t^2 \lambda^A_{min}} ||\boldsymbol{w}_0 - \boldsymbol{w}_\star||^2 \ .$$

PROOF. We can simplify (4.3.6) to be:

$$\begin{aligned} E = \frac{||2(\mathbf{w} - \mathbf{w}_\star) + t\dot{\mathbf{w}}||^2}{2} + (\beta - 3)||\mathbf{w} - \mathbf{w}_\star||^2 - t^2\langle h(\mathbf{w} - h(\mathbf{w}_\star), \mathbf{w} - \mathbf{w}_\star)\rangle \\ + \int_0^t \left(\frac{\beta - 3}{s}\right) ||\dot{\mathbf{w}}||^2 + \frac{2}{s}(\mathbf{w} - \mathbf{w}_\star)^T A(\mathbf{w} - \mathbf{w}_\star) \ . \end{aligned} \tag{4.3.7}$$

We can further simplify E_0 as $t_0 \to 0$ which would be applied to E after substituting $\alpha = 2$. We can therefore write E_0 as:

$$E_0 = (5 - \beta)||\mathbf{w}_0 - \mathbf{w}_\star||^2 \ . \tag{4.3.8}$$

Equating (4.3.7) and (4.3.8), we get:

$$\begin{aligned} (5 - \beta)||\mathbf{w}_0 - \mathbf{w}_\star||^2 = \frac{||2(\mathbf{w} - \mathbf{w}_\star) + t\dot{\mathbf{w}}||^2}{2} + (\beta - 3)||\mathbf{w} - \mathbf{w}_\star||^2 - t^2\langle h(\mathbf{w} - h(\mathbf{w}_\star), \mathbf{w} - \mathbf{w}_\star)\rangle \\ + \int_0^t \left(\frac{\beta - 3}{s}\right) ||\dot{\mathbf{w}}||^2 + \frac{2}{s}(\mathbf{w} - \mathbf{w}_\star)^T A(\mathbf{w} - \mathbf{w}_\star) \ . \end{aligned} \tag{4.3.9}$$

Since we are calculating the rate, we can neglect positive terms that do not contribute to the rate and transform (4.3.9) from an equality to the following inequality:

$$(\beta - 3)||\mathbf{w} - \mathbf{w}_\star||^2 - t^2\langle h(\mathbf{w}) - h(\mathbf{w}_\star), \mathbf{w} - \mathbf{w}_\star\rangle \leq (5 - \beta)||\mathbf{w}_0 - \mathbf{w}_\star||^2 \ . \tag{4.3.10}$$

Any $\beta \in [3,5]$ ensures that all the terms remain positive. We choose $\beta = 3$ to get:

$$-t^2\langle h(\mathbf{w}) - h(\mathbf{w}_\star), \mathbf{w} - \mathbf{w}_\star\rangle \leq 2||\mathbf{w}_0 - \mathbf{w}_\star||^2 \ . \tag{4.3.11}$$

Expanding on the term $\langle h(\mathbf{w}) - h(\mathbf{w}_\star), \mathbf{w} - \mathbf{w}_\star\rangle$ in (4.3.11), we get:

$$\begin{aligned} -t^2\langle \cancel{b} - A\mathbf{w} - \cancel{b} + A\mathbf{w}_\star, \mathbf{w} - \mathbf{w}_\star\rangle &\leq 2||\mathbf{w}_0 - \mathbf{w}_\star||^2 \\ t^2\langle A(\mathbf{w} - \mathbf{w}_\star), \mathbf{w} - \mathbf{w}_\star\rangle &\leq 2||\mathbf{w}_0 - \mathbf{w}_\star||^2 \\ t^2(\mathbf{w} - \mathbf{w}_\star)^T A(\mathbf{w} - \mathbf{w}_\star) &\leq 2||\mathbf{w}_0 - \mathbf{w}_\star||^2 \ . \end{aligned} \tag{4.3.12}$$

Now, we know that for a positive definite matrix $A \in \mathbb{R}^{n \times n}$ and $\mathbf{x} \in \mathbb{R}^n$, we can write

$$\lambda^A_{min}||\mathbf{x}||^2 \leq \mathbf{x}^T A\mathbf{x} \leq \lambda^A_{max}||\mathbf{x}||^2 \ ,$$

where λ^A_{min} is the minimum eigenvalue of the matrix A and λ^A_{max} is the maximum eigenvalue of A. Applying this for (4.3.12), we get:

$$t^2\lambda^A_{min}||\mathbf{w} - \mathbf{w}_\star||^2 \leq 2||\mathbf{w}_0 - \mathbf{w}_\star||^2 \ .$$

Finally, rearranging the terms of the above expression, we get:

$$||\mathbf{w} - \mathbf{w}_\star||^2 \leq \frac{2}{t^2 \lambda_{min}^A} ||\mathbf{w}_0 - \mathbf{w}_\star||^2 \ . \tag{4.3.13}$$

□

Note that since A is a positive definite matrix, $\lambda_{min}^A > 0$. Hence, the coefficient of $||\mathbf{w}_0 - \mathbf{w}_\star||^2$ is always positive. Using the expression given by (4.3.13) and the whole conservation law analysis in general, we can make the following conclusions:

(1) **Choice of Damping Parameter.** For the case of Polyak TD(0), we have one damping parameter, β. One major advantage of performing this conservation law analysis is that we get the exact value to choose for β, i.e, $\beta = 3$. Hence, the only hyperparameter we need to optimize for is the learning rate.

(2) **Boundedness of the term $||\mathbf{w} - \mathbf{w}_\star||^2$**: Using this expression, we can say that $||\mathbf{w} - \mathbf{w}_\star||^2$ is always bounded by $||\mathbf{w}_0 - \mathbf{w}_\star||^2$.

(3) **Convergence Rate for Polyak's momentum.** Since the coefficient $\frac{2}{t^2 \lambda_{min}^A}$ is positive, (4.3.13) serves as the convergence rate equation for Polyak TD(0) with linear function approximation. Hence, we recover a convergence rate of $\mathcal{O}(1/(t^2 \lambda_{min}^A))$ for the underlying dynamical system from our analysis.

4.4. Conservation law for Nesterov TD(0)

Consider the underlying ODE for Nesterov TD(0) given by:

$$\ddot{\mathbf{w}}(t) + 2d(t)\dot{\mathbf{w}}(t) = h(\mathbf{w}(t) + \beta(t)\dot{\mathbf{w}}(t)) = \mathbf{b} - A(\mathbf{w}(t) + \beta(t)\dot{\mathbf{w}}(t)) \ , \tag{4.4.1}$$

where $A = \mathbb{E}_{\mathbf{x}}[\boldsymbol{\phi}_t(s_t)(\boldsymbol{\phi}_t(s_t) - \gamma \boldsymbol{\phi}_t(s_t))]$, $b = \mathbb{E}_{\mathbf{x}}[\mathbf{r}_t \boldsymbol{\phi}_t(s_t)]$, and $2d(t) + \beta(t) = 1$.

Note that parameters $d(t), \beta(t)$ are dependent on t. Simplifying (4.4.1), we get:

$$\ddot{\mathbf{w}}(t) + (2d(t)I + \beta(t)A)\dot{\mathbf{w}}(t) = \mathbf{b} - A\mathbf{w}(t) \ . \tag{4.4.2}$$

Let $\mathbf{w}_\star$ denote the minimizer for $h(\mathbf{w}(t))$. Consider the dilated coordinate system $\mathbf{y} = (t + c)^\alpha (\mathbf{w} - \mathbf{w}_\star)$ with yet undetermined $\alpha, c \in \mathbb{R}$. We can therefore express (4.4.2) in the $\mathbf{y}$-coordinate system.

$$\begin{aligned}
\mathbf{y} &= (t + c)^\alpha (\mathbf{w} - \mathbf{w}_\star) \ , \\
\dot{\mathbf{y}} &= \alpha (t + c)^{\alpha - 1} (\mathbf{w} - \mathbf{w}_\star) + (t + c)^\alpha \dot{\mathbf{w}} \ , \\
\ddot{\mathbf{y}} &= \alpha(\alpha - 1)(t + c)^{\alpha - 2} (\mathbf{w} - \mathbf{w}_\star) + 2\alpha (t + c)^{\alpha - 1} \dot{\mathbf{w}} + (t + c)^\alpha \ddot{\mathbf{w}} \ .
\end{aligned}$$

Rewriting $\mathbf{y}, \dot{\mathbf{y}}, \ddot{\mathbf{y}}$ in terms of $\mathbf{w}, \dot{\mathbf{w}}, \ddot{\mathbf{w}}$, we get:

$$\begin{aligned}
\mathbf{w} &= (t+c)^{-\alpha}\mathbf{y} + \mathbf{w}_\star \ , \\
\dot{\mathbf{w}} &= (t+c)^{-\alpha}\dot{\mathbf{y}} - \alpha(t+c)^{-(\alpha+1)}\mathbf{y} \ , \\
\ddot{\mathbf{w}} &= (t+c)^{-\alpha}\ddot{\mathbf{y}} - 2\alpha(t+c)^{-(\alpha+1)}\dot{\mathbf{y}} + (\alpha^2+\alpha)(t+c)^{-(\alpha+2)}\mathbf{y} \ .
\end{aligned}$$

Substituting $\mathbf{w}, \dot{\mathbf{w}}, \ddot{\mathbf{w}}$ in (4.4.2), we get:

$$\begin{aligned}
&(t+c)^{\alpha}\ddot{\mathbf{y}} - 2\alpha(t+c)^{-(\alpha+1)}\dot{\mathbf{y}} + (\alpha^2+\alpha)(t+c)^{-(\alpha+2)}\mathbf{y} \\
&\qquad + (2d(t)T + \beta(t)A)\left[(t+c)^{-\alpha}\dot{\mathbf{y}} - \alpha(t+c)^{-(\alpha+1)}\mathbf{y}\right] = \mathbf{b} - A\left[(t+c)^{-\alpha}\mathbf{y} + \mathbf{w}_\star\right] \ .
\end{aligned}$$

Let us define

$$g(\mathbf{y}(t)) = (\alpha^2+\alpha)(t+c)^{-(\alpha+2)}\mathbf{y} - \alpha(t+c)^{-(\alpha+1)}\left[2d(t)I + \beta(t)A\right]\mathbf{y} - \mathbf{b} + A\left[(t+c)^{-\alpha}\mathbf{y} + \mathbf{w}_\star\right] \ .$$

We can therefore rewrite the ODE in terms of $g(\mathbf{y}(t))$ as:

$$(t+c)^{\alpha}\ddot{\mathbf{y}} - 2\alpha(t+c)^{-(\alpha+1)}\dot{\mathbf{y}} + (t+c)^{-\alpha}\left[2d(t)I + \beta(t)A\right]\dot{\mathbf{y}} + g(\mathbf{y},t) = 0 \ . \tag{4.4.3}$$

Lemma 4.4.1. *Let $\boldsymbol{y}$ evolve according to* (4.4.3)*. Denote $U(\boldsymbol{y},t) = \int_{\boldsymbol{y}} g(\boldsymbol{y},t)d\boldsymbol{y}$ where we can express $g(\boldsymbol{y},t)$ as: $g(\boldsymbol{y}(t)) = (\alpha^2+\alpha)(t+c)^{-(\alpha+2)}\boldsymbol{y} - \alpha(t+c)^{-(\alpha+1)}\left[2d(t)I + \beta(t)A\right]\boldsymbol{y} - \boldsymbol{b} + A\left[(t+c)^{-\alpha}\boldsymbol{y} + \boldsymbol{w}_\star\right]$. Then $\forall t \geq 0$, we have:*

$$U(\boldsymbol{y},t) = \left[\frac{\alpha(\alpha+1)}{t+c} - 2\alpha d(t)\right](t+c)^{-(\alpha+1)}||\boldsymbol{y}||^2 + (t+c)^{-\alpha}\left[1 - \frac{\alpha\beta(t)}{t+c}\right]\boldsymbol{y}^T A\boldsymbol{y} - \langle \boldsymbol{b} - A\boldsymbol{w}_\star, \boldsymbol{y}\rangle \ .$$

Proof. Let us define $U(\mathbf{y},t) = \int_{\mathbf{y}} g(\mathbf{y},t)d\mathbf{y}$. Performing this operation, we get:

$$\begin{aligned}
U(\mathbf{y},t) &= \int_{\mathbf{y}} g(\mathbf{y},t)d\mathbf{y} \\
&= \int_{\mathbf{y}} \Big[(\alpha^2+\alpha)(t+c)^{-(\alpha+2)}\mathbf{y} - \alpha(t+c)^{-(\alpha+1)}\left[2d(t)I + \beta(t)A\right]\mathbf{y} \\
&\qquad\qquad - \mathbf{b} + A\left[(t+c)^{-\alpha}\mathbf{y} + \mathbf{w}_\star\right]d\mathbf{y} \\
&= (\alpha^2+\alpha)(t+c)^{-(\alpha+2)}||\mathbf{y}||^2 - 2\alpha d(t)(t+c)^{-(\alpha+1)}||\mathbf{y}||^2 - \alpha\beta(t)(t+c)^{-(\alpha+1)}\mathbf{y}^T A\mathbf{y} \\
&\qquad\qquad - \mathbf{b}^T\mathbf{y} + (t+c)^{-\alpha}\mathbf{y}^T A\mathbf{y} + \mathbf{w}_\star^T A\mathbf{y} \\
&= \left[\frac{\alpha(\alpha+1)}{t+c} - 2\alpha d(t)\right](t+c)^{-(\alpha+1)}||\mathbf{y}||^2 + (t+c)^{-\alpha}\left[1 - \frac{\alpha\beta(t)}{t+c}\right]\mathbf{y}^T A\mathbf{y} - \langle \mathbf{b} - A\mathbf{w}_\star, \mathbf{y}\rangle \ .
\end{aligned}$$

□

Note that we have described the validity of performing the operation $U(\mathbf{y},t) = \int_{\mathbf{y}} g(\mathbf{y},t)d\mathbf{y}$ in 4.1. We will now use Lemma (4.4.1) to obtain a conservation law for Nesterov TD(0).

Proposition 4.4.2. *Let $\boldsymbol{w}$ and $\boldsymbol{y}$ evolve according to* (4.4.2) *and* (4.4.3) *respectively. Consider an arbitrary but fixed $\boldsymbol{w}_0 \in \mathbb{R}^n$ such that $\boldsymbol{w}(0) = \boldsymbol{w}_0$ and $\dot{\boldsymbol{w}}(0) = 0$. Then $\forall t \geq 0$, the*

conservation law can be given as:

$$
\begin{aligned}
E = (t+c)^{\alpha-2}||\alpha(\boldsymbol{w}-\boldsymbol{w}_\star)+(t+c)\dot{\boldsymbol{w}}||^2 + \left[\frac{\alpha(\alpha+1)}{t+c} - 2\alpha d(t)\right](t+c)^{\alpha-1}||\boldsymbol{w}-\boldsymbol{w}_\star||^2 \\
+ \left[1-\frac{\alpha\beta(t)}{t+c}\right](t+c)^{\alpha}(\boldsymbol{w}-\boldsymbol{w}_\star)^T A(\boldsymbol{w}-\boldsymbol{w}_\star) \\
+ \int_0^t \left[2d(s) - \frac{3\alpha}{2(s+c)}\right](s+c)^{\alpha-2}||\alpha(\boldsymbol{w}-\boldsymbol{w}_\star)+(s+c)\dot{\boldsymbol{w}}||^2 ds \\
+ \int_0^t \beta(s)(s+c)^{\alpha-2}\left[\alpha(\boldsymbol{w}-\boldsymbol{w}_\star)+(s+c)\dot{\boldsymbol{w}}\right]^T A\left[\alpha(\boldsymbol{w}-\boldsymbol{w}_\star)+(s+c)\dot{\boldsymbol{w}}\right] ds \\
+ \int_0^t \left[-\frac{2\alpha(\alpha+1)d(s)}{s+c} + \frac{\alpha(\alpha+1)(\alpha+2)}{(s+c)^2} + 2\alpha\dot{d}(s)\right](s+c)^{\alpha-1}||\boldsymbol{w}-\boldsymbol{w}_\star||^2 ds \\
+ \int_0^t \left[-\frac{\alpha(\alpha+1)\beta(s)}{s+c} + \alpha\dot{\beta}(s) + \alpha\right](s+c)^{\alpha-1}(\boldsymbol{w}-\boldsymbol{w}_\star)^T A(\boldsymbol{w}-\boldsymbol{w}_\star) ds \ .
\end{aligned}
$$

Proof. Taking the inner product of (4.4.3) with $\dot{\mathbf{y}}$, we get:

$$
\begin{aligned}
(t+c)^{-\alpha}\langle \mathbf{y}, \ddot{\mathbf{y}}\rangle + 2d(t)(t+c)^{-\alpha}||\dot{\mathbf{y}}||^2 + \beta(t)(t+c)^{-\alpha}\langle \dot{\mathbf{y}}, A\dot{\mathbf{y}}\rangle - 2\alpha(t+c)^{-(\alpha+1)}||\dot{\mathbf{y}}||^2 \\
+ (\alpha^2+\alpha)(t+c)^{-(\alpha+2)}\langle \dot{\mathbf{y}}, \mathbf{y}\rangle - 2\alpha d(t)(t+c)^{-(\alpha+1)}\langle \dot{\mathbf{y}}, \mathbf{y}\rangle \\
- \alpha\beta(t)(t+c)^{-(\alpha+1)}\langle \dot{\mathbf{y}}, A\mathbf{y}\rangle - \langle \mathbf{b} - A\left[(t+c)^{-\alpha}\mathbf{y} + \mathbf{w}_\star\right], \dot{\mathbf{y}}\rangle = 0 \ .
\end{aligned}
$$

Now, we have

$$
\frac{d}{dt}\left(\frac{(t+c)^{-\alpha}}{2}||\dot{\mathbf{y}}||^2\right) = (t+c)^{-\alpha}\langle \ddot{\mathbf{y}}, \dot{\mathbf{y}}\rangle - \frac{\alpha(t+c)^{-(\alpha+1)}}{2}||\dot{\mathbf{y}}||^2 \ .
$$

From Definition (4.2.3), we can say that:

$$
< g(\mathbf{y},t), \dot{\mathbf{y}} >= \frac{d}{dt}U(\mathbf{y},t) - \frac{\partial}{\partial t}U(\mathbf{y}(t),t) \ .
$$

Substituting these in the above equation, we get:

$$
\begin{aligned}
\frac{d}{dt}\left(\frac{(t+c)^{-\alpha}}{2}||\dot{\mathbf{y}}||^2 + U(\mathbf{y},t)\right) - \frac{3\alpha}{2}(t+c)^{-(\alpha+1)}||\dot{\mathbf{y}}||^2 + 2d(t)(t+c)^{-\alpha}||\dot{\mathbf{y}}||^2 \\
+ \beta(t)(t+c)^{-\alpha}\langle \dot{\mathbf{y}}, A\dot{\mathbf{y}}\rangle - \frac{\partial}{\partial t}U(\mathbf{y},t) = 0 \ .
\end{aligned} \tag{4.4.4}
$$

We can therefore see that this equation is of the form $\dot{A}(t) + B(t) = 0$ as described in Definition (4.2.2), where we have:

$$
\begin{aligned}
A(t) &= \frac{(t+c)^{-\alpha}}{2}||\dot{\mathbf{y}}||^2 + U(\mathbf{y},t), \quad \text{and} \\
B(t) &= -\frac{3\alpha}{2}(t+c)^{-(\alpha+1)}||\dot{\mathbf{y}}||^2 + 2d(t)(t+c)^{-\alpha}||\dot{\mathbf{y}}||^2 + \beta(t)(t+c)^{-\alpha}\langle \dot{\mathbf{y}}, A\dot{\mathbf{y}}\rangle - \frac{\partial}{\partial t}U(\mathbf{y},t) \ .
\end{aligned}
$$

Substituting $U(\mathbf{y},t)$ from Lemma (4.4.1), we will now find the partial derivative of $U(\mathbf{y},t)$ with respect to t:

$$\begin{aligned}\frac{\partial U(\mathbf{y},t)}{\partial t} &= -(\alpha+2)(\alpha^2+\alpha)(t+c)^{-(\alpha+3)}||\mathbf{y}||^2 + 2\alpha(\alpha+1)d(t)(t+c)^{-(\alpha+2)}||\mathbf{y}||^2 \\ &\quad - 2\alpha\dot{d}(t)(t+c)^{-(\alpha+1)}||\mathbf{y}||^2 + \alpha(\alpha+1)\beta(t)(t+c)^{-(\alpha+2)}\mathbf{y}^T A\mathbf{y} \\ &\quad - \alpha\dot{\beta}(t)(t+c)^{-(\alpha+1)}\mathbf{y}^T A\mathbf{y} - \alpha(t+c)^{-(\alpha+1)}\mathbf{y}^T A\mathbf{y} \\ &= \left[\frac{2\alpha(\alpha+1)d(t)}{t+c} - \frac{(\alpha+2)(\alpha+1)\alpha}{(t+c)^2} - 2\alpha\dot{d}(t)\right](t+c)^{-(\alpha+1)}||\mathbf{y}^2|| \\ &\quad + \left[\frac{\alpha(\alpha+1)\beta(t)}{t+c} - \alpha\dot{\beta}(t) - \alpha\right](t+c)^{-(\alpha+1)}\mathbf{y}^T A\mathbf{y}\end{aligned}$$

Substituting these terms in (4.4.4), we get:

$$\begin{aligned}&\frac{d}{dt}\left[\frac{(t+c)^{-\alpha}}{2}||\dot{\mathbf{y}}||^2 + \left[\frac{\alpha(\alpha+1)}{t+c} - 2\alpha d(t)\right](t+c)^{-(\alpha+1)}||\mathbf{y}||^2\right] \\ &\quad + \frac{d}{dt}\left[\left(1 - \frac{\alpha\beta(t)}{t+c}\right)(t+c)^{-\alpha}\mathbf{y}^T A\mathbf{y} - \langle \mathbf{b} - A\mathbf{w}_\star, \mathbf{y}\rangle\right] \\ &\quad - \frac{3\alpha}{2}(t+c)^{-(\alpha+1)}||\dot{\mathbf{y}}||^2 + 2d(t)(t+c)^{-\alpha}||\dot{\mathbf{y}}||^2 + \beta(t)(t+c)^{-\alpha}\langle\dot{\mathbf{y}}, A\dot{\mathbf{y}}\rangle \\ &\quad - \left[\frac{2\alpha(\alpha+1)d(t)}{t+c} - \frac{(\alpha+2)(\alpha+1)\alpha}{(t+c)^2} - 2\alpha\dot{d}(t)\right](t+c)^{-(\alpha+1)}||\mathbf{y}^2|| \\ &\quad - \left[\frac{\alpha(\alpha+1)\beta(t)}{t+c} - \alpha\dot{\beta}(t) - \alpha\right](t+c)^{-(\alpha+1)}\mathbf{y}^T A\mathbf{y} = 0 \ .\end{aligned}$$

Note that using Definition (4.2.2), we can write our conservation law as:

$$\begin{aligned}E &= A(t) + \int_{t_0}^{t} B(s)ds \\ &= \frac{(t+c)^{-\alpha}}{2}||\dot{\mathbf{y}}||^2 + \left[\frac{\alpha(\alpha+1)}{t+c} - 2\alpha d(t)\right](t+c)^{-(\alpha+1)}||\mathbf{y}||^2 + \left[1 - \frac{\alpha\beta(t)}{t}\right](t+c)^{-\alpha}\mathbf{y}^T A\mathbf{y} \\ &\quad - \langle \mathbf{b} - A\mathbf{w}_\star, \mathbf{y}\rangle + \int_0^t \left[2d(s) - \frac{3\alpha}{2(s+c)}\right](s+c)^{-\alpha}||\dot{\mathbf{y}}||^2 + \beta(s)(s+c)^{-\alpha}\langle\dot{\mathbf{y}}, A\dot{\mathbf{y}}\rangle ds \\ &\quad - \int_0^t \left[\frac{2\alpha(\alpha+1)d(s)}{s+c} - \frac{\alpha(\alpha+1)(\alpha+2)}{(s+c)^2} - 2\alpha\dot{d}(s)\right](s+c)^{-(\alpha+1)}||\mathbf{y}||^2 ds \\ &\quad - \int_0^t \left[\frac{\alpha(\alpha+1)\beta(s)}{s+c} - \alpha\dot{\beta}(s) - \alpha\right](s+c)^{-(\alpha+1)}\mathbf{y}^T A\mathbf{y} ds \ .\end{aligned}$$

Substituting $\mathbf{y}, \dot{\mathbf{y}}$, we get:

$$
\begin{aligned}
E = (t+c)^{\alpha-2}||\alpha(\mathbf{w}-\mathbf{w}_\star)+(t+c)\dot{\mathbf{w}}||^2 &+ \left[\frac{\alpha(\alpha+1)}{t+c} - 2\alpha d(t)\right](t+c)^{\alpha-1}||\mathbf{w}-\mathbf{w}_\star||^2 \\
&+ \left[1-\frac{\alpha\beta(t)}{t+c}\right](t+c)^{\alpha}(\mathbf{w}-\mathbf{w}_\star)^T A(\mathbf{w}-\mathbf{w}_\star) \\
&+ \int_0^t \left[2d(s) - \frac{3\alpha}{2(s+c)}\right](s+c)^{\alpha-2}||\alpha(\mathbf{w}-\mathbf{w}_\star)+(s+c)\dot{\mathbf{w}}||^2 ds \\
&+ \int_0^t \beta(s)(s+c)^{\alpha-2}\left[\alpha(\mathbf{w}-\mathbf{w}_\star)+(s+c)\dot{\mathbf{w}}\right]^T A\left[\alpha(\mathbf{w}-\mathbf{w}_\star)+(s+c)\dot{\mathbf{w}}\right] ds \\
&+ \int_0^t \left[-\frac{2\alpha(\alpha+1)d(s)}{s+c} + \frac{\alpha(\alpha+1)(\alpha+2)}{(s+c)^2} + 2\alpha\dot{d}(s)\right](s+c)^{\alpha-1}||\mathbf{w}-\mathbf{w}_\star||^2 ds \\
&+ \int_0^t \left[-\frac{\alpha(\alpha+1)\beta(s)}{s+c} + \alpha\dot{\beta}(s) + \alpha\right](s+c)^{\alpha-1}(\mathbf{w}-\mathbf{w}_\star)^T A(\mathbf{w}-\mathbf{w}_\star) ds \ .
\end{aligned}
\tag{4.4.5}
$$

□

Lemma 4.4.3. *Let $E_0 = \lim_{t\to t_0} E$ denote the conservation law at the starting point t_0. Then,*

$$
\begin{aligned}
E_0 = \frac{(t+c)^{\alpha-2}}{2}||\alpha(\boldsymbol{w}_0-\boldsymbol{w}_\star)||^2 &+ \left[\frac{\alpha(\alpha+1)}{t+c} - 2\alpha d(t)\right](t+c)^{\alpha-1}||\boldsymbol{w}_0-\boldsymbol{w}_\star||^2 \\
&+ (t+c)^{\alpha}\left[1-\frac{\alpha\beta(t)}{t+c}\right](\boldsymbol{w}_0-\boldsymbol{w}_\star)^T A(\boldsymbol{w}_0-\boldsymbol{w}_\star) - (t+c)^{\alpha}\langle \boldsymbol{b}-A\boldsymbol{w}_\star, \boldsymbol{w}_0-\boldsymbol{w}_\star\rangle \ .
\end{aligned}
$$

Proof. Using Definition (4.2.2), E can be expressed at the initial start point t_0 as:

$$
\begin{aligned}
E_0 &= \lim_{t\to t_0} A(t) \\
&= \lim_{t\to t_0}\left[\frac{(t+c)^{-\alpha}}{2}||\dot{\mathbf{y}}||^2 + \left[\frac{\alpha(\alpha+1)}{t+c} - 2\alpha d(t)\right](t+c)^{-(\alpha+1)}||\mathbf{y}||^2 + \left[1-\frac{\alpha\beta(t)}{t+c}\right](t+c)^{-\alpha}\mathbf{y}^T A\mathbf{y}\right] \\
&\qquad - \lim_{t\to t_0}\left[\langle \mathbf{b}-A\mathbf{w}_\star, \mathbf{y}\rangle\right] \ .
\end{aligned}
$$

Substituting for $\dot{\mathbf{y}}$, we get:

$$
\begin{aligned}
E_0 = \frac{(t+c)^{\alpha-2}}{2}||\alpha(\mathbf{w}_0-\mathbf{w}_\star)+(t+c)\dot{\mathbf{w}}||^2 &+ \left[\frac{\alpha(\alpha+1)}{t+c} - 2\alpha d(t)\right](t+c)^{\alpha-1}||\mathbf{w}_0-\mathbf{w}_\star||^2 \\
&+ (t+c)^{\alpha}\left[1-\frac{\alpha\beta(t)}{t+c}\right](\mathbf{w}_0-\mathbf{w}_\star)^T A(\mathbf{w}_0-\mathbf{w}_\star) - (t+c)^{\alpha}\langle \mathbf{b}-A\mathbf{w}_\star, \mathbf{w}_0-\mathbf{w}_\star\rangle \ .
\end{aligned}
$$

For E_0, we have $\dot{w}_0 = 0$. Hence, we can simplify it further as

$$E_0 = \frac{(t+c)^{\alpha-2}}{2}||\alpha(\mathbf{w}_0 - \mathbf{w}_\star)||^2 + \left[\frac{\alpha(\alpha+1)}{t+c} - 2\alpha d(t)\right](t+c)^{\alpha-1}||\mathbf{w}_0 - \mathbf{w}_\star||^2 + (t+c)^\alpha\left[1 - \frac{\alpha\beta(t)}{t+c}\right](\mathbf{w}_0 - \mathbf{w}_\star)^T A(\mathbf{w}_0 - \mathbf{w}_\star) - (t+c)^\alpha\langle \mathbf{b} - A\mathbf{w}_\star, \mathbf{w}_0 - \mathbf{w}_\star\rangle \ . \tag{4.4.6}$$

□

We would now like to use this conservation law relation to derive the convergence rate for our NesterovTD(0) method. In anticipation for a $\mathcal{O}(1/t^2)$, we choose $\alpha = 2$. Note that to calculate the rate, we would like other non-contributing terms to be positive. To this end, we get certain inequalities determining the possible values of $d(t)$. We choose the corresponding $\beta(t)$ using the relation $2d(t) + \beta(t) = 0$. We also choose $a = 4$ to ensure that the derived expression would hold $\forall t \geq 0$. Hence, we have

$$\beta(t) = \frac{t+1}{t+4} \quad \text{and} \quad d(t) = \frac{3}{2(t+4)} \ .$$

Note that these expressions for $\beta(t)$ and $d(t)$ provide the exact values that we need to set our damping parameters to for obtaining the required convergent and accelerated dynamics. This is an added advantage of doing these conservation law-based analyses for our accelerated methods.

Theorem 4.4.4. *Let $\boldsymbol{w}$ evolve according to* (4.4.2) *which solves for $h(\boldsymbol{w}) = \boldsymbol{b} - A\boldsymbol{w}$ where A is positive definite. Let $\boldsymbol{w}_\star$ be the minimizer for $h(\boldsymbol{w})$ such that $h(\boldsymbol{w}_\star) = 0$. Let $\boldsymbol{w}(0) = \boldsymbol{w}_0 \in \mathbb{R}^n$ be an arbitrary but fixed initial point, and $\dot{\boldsymbol{w}}_0 = 0$ denote the initial conditions. For the conservation law E and E_0 given by Proposition* (4.4.2) *and Lemma* (4.4.3) *respectively, setting $\alpha = 2$, we get the convergence rate expression for* (4.4.2) *as:*

$$||\boldsymbol{w} - \boldsymbol{w}_\star||^2 \leq \frac{6}{t^2\lambda^A_{min}}||\boldsymbol{w}_0 - \boldsymbol{w}_\star||^2 \ ,$$

where λ^A_{min} is the minimum eigenvalue of matrix A.

Proof. Putting these values in (4.4.5), we get:

$$\begin{aligned} E =& ||2(\mathbf{w} - \mathbf{w}_\star) + (t+4)\dot{\mathbf{w}}||^2 + \left(t^2 + 6t + 14\right)(\mathbf{w} - \mathbf{w}_\star)^T A(\mathbf{w} - \mathbf{w}_\star) \\ &+ \int_0^t \left[\frac{s+1}{s+4}\right][2(\mathbf{w} - \mathbf{w}_\star) + (s+4)\dot{\mathbf{w}}]^T A\,[2(\mathbf{w} - \mathbf{w}_\star) + (s+4)\dot{\mathbf{w}}]\,ds \\ &+ \int_0^t \left[\frac{2s^2 + 10s + 29}{s+4}\right](\mathbf{w} - \mathbf{w}_\star)^T A(\mathbf{w} - \mathbf{w}_\star)ds \ . \end{aligned} \tag{4.4.7}$$

Substituting the values of $\alpha = 2$, $\beta(t)$, $d(t)$ in (4.4.6) and applying $t \to 0$ to the expression, E_0 simplifies to:

$$E_0 = 2||\mathbf{w}_0 - \mathbf{w}_\star||^2 + 14(\mathbf{w}_0 - \mathbf{w}_\star)^T A(\mathbf{w}_0 - \mathbf{w}_\star) \ . \tag{4.4.8}$$

For a positive definite matrix $A \in \mathbb{R}^{n\times n}$ and $\mathbf{x} \in \mathbb{R}^n$, we can write

$$\lambda^A_{min}||\mathbf{x}||^2 \leq \mathbf{x}^T A\mathbf{x} \leq \lambda^A_{max}||\mathbf{x}||^2 \ , \tag{4.4.9}$$

where λ^A_{min} is the minimum eigenvalue of matrix A and λ^A_{max} is the maximum eigenvalue of A.

We can therefore simplify E_0 from (4.4.8) to get:

$$\begin{aligned} E_0 &= 2||\mathbf{w}_0 - \mathbf{w}_\star||^2 + 14(\mathbf{w}_0 - \mathbf{w}_\star)^T A(\mathbf{w}_0 - \mathbf{w}_\star) \\ &\leq 2||\mathbf{w}_0 - \mathbf{w}_\star||^2 + 14\lambda^A_{max}||\mathbf{w}_0 - \mathbf{w}_\star||^2 \\ &= \left(2 + 14\lambda^A_{max}\right)||\mathbf{w}_0 - \mathbf{w}_\star||^2 \ . \end{aligned} \tag{4.4.10}$$

Since energy remains conserved throughout, we can write $E = E_0$. Equating (4.4.7) and (4.4.10), we get:

$$\begin{aligned} \left(2 + 14\lambda^A_{max}\right)||\mathbf{w}_0 - \mathbf{w}_\star||^2 \geq &||2(\mathbf{w} - \mathbf{w}_\star) + (t+4)\dot{\mathbf{w}}||^2 + \left(t^2 + 6t + 14\right)(\mathbf{w} - \mathbf{w}_\star)^T A(\mathbf{w} - \mathbf{w}_\star) \\ &+ \int_0^t \left[\frac{s+1}{s+4}\right][2(\mathbf{w} - \mathbf{w}_\star) + (s+4)\dot{\mathbf{w}}]^T A\,[2(\mathbf{w} - \mathbf{w}_\star) + (s+4)\dot{\mathbf{w}}]\,ds \\ &+ \int_0^t \left[\frac{2s^2 + 10s + 29}{s+4}\right](\mathbf{w} - \mathbf{w}_\star)^T A(\mathbf{w} - \mathbf{w}_\star)ds \ . \end{aligned} \tag{4.4.11}$$

Since we are calculating the rate, we can neglect positive terms that do not contribute to the rate and transform (4.4.11) into the following inequality:

$$\left(t^2 + 6t + 14\right)(\mathbf{w} - \mathbf{w}_\star)^T A(\mathbf{w} - \mathbf{w}_\star) \leq \left(2 + 14\lambda^A_{max}\right)||\mathbf{w}_0 - \mathbf{w}_\star||^2 \ . \tag{4.4.12}$$

Applying (4.4.9) for the LHS of (4.4.12), we get:

$$\left(t^2 + 6t + 14\right)\lambda^A_{min}||\mathbf{w} - \mathbf{w}_\star||^2 \leq \left(2 + 14\lambda^A_{max}\right)||\mathbf{w}_0 - \mathbf{w}_\star||^2 \ .$$

Note that $t^2+6t+14$ is a positive and monotonically increasing polynomial $\forall t \geq 0$. Therefore, we get:

$$||\mathbf{w} - \mathbf{w}_\star||^2 \leq \frac{\left(2 + 14\lambda^A_{max}\right)}{\left(t^2 + 6t + 14\right)\lambda^A_{min}}||\mathbf{w}_0 - \mathbf{w}_\star||^2 \ . \tag{4.4.13}$$

□

Again, since A is positive definite, we know that $\lambda^A_{min} > 0$ and $\lambda^A_{max} > 0$. Hence, the coefficient of $||\mathbf{w}_0 - \mathbf{w}_\star||^2$ is always positive. Similar to the case of Polyak TD(0), using the expression given by (4.4.13) and the conservation law based analysis in general, we can make the following conclusions:

(1) **Choice of the damping parameters.** For our Nesterov TD(0), we have two damping parameters, $\beta(t)$ and $d(t)$. One major advantage of performing this conservation law analysis is that we get exact expressions for the choice of these hyperparameters. Hence, the only HP we need to optimize for is the learning rate.

(2) **Boundedness of the term $||\mathbf{w} - \mathbf{w}_\star||^2$.** $||\mathbf{w} - \mathbf{w}_\star||^2$ is shown to be always bounded by $||\mathbf{w}_0 - \mathbf{w}_\star||^2$.

(3) **Convergence Rate of Nesterov TD(0)**. Since the coefficient $\frac{\left(2+14\lambda^A_{max}\right)}{(t^2+6t+14)\lambda^A_{min}}$ is always positive $\forall t \geq 0$, (4.4.13) serves as the convergence rate equation for Nesterov TD(0) with linear function approximation. Hence, we recover a convergence rate of $\mathcal{O}(1/(t^2\lambda^A_{min}))$ for the underlying dynamical system from our analysis.

This concludes our conservation-law based analyses for the proposed accelerated methods.

Concluding Remarks: In this chapter, we took the proposed accelerated methods and performed a conservation-law based analyses to get the convergence rates of the underlying dynamical system behind Polyak TD(0) and NesterovTD(0) introduced in the previous chapter. We also get the analytical expressions for the hyperparameters corresponding to the damping parameters, $\beta(t)$, and $d(t)$ needed to obtain the derived convergence rate.

In the next chapter, we will take the proposed accelerated methods stated in Chapter 3 along with the analytical expressions for hyperparameters derived in this chapter to state the discrete-time implementable accelerated algorithms. We would then report and analyse the empirical performance of our proposed algorithms as compared to other baselines on small linear prediction tasks.

Chapter 5

Experiments

We will now illustrate the performance of the accelerated versions of TD(0) algorithm proposed in this work compared to other variants. While these experiments do not provide an exhaustive characterization of the empirical performance, they nevertheless serve as proof-of-concept results and highlight the convergent behaviour of our method.

We start by stating our final ODEs and the corresponding discrete-time update equations after incorporating the analytical values of the damping parameters obtained through our convergence analysis in Chapter 4. For a complete review of the experiments, we then give a detailed description of our problem setting and the experimental setup before we show how our proposed accelerated TD(0) algorithms perform when compared with existing TD(0) algorithms designed for linear problem settings.

5.1. Choice of the damping parameter

In this section, we use the analytical damping parameters obtained during the conservation law-based convergence analysis to state the final form of our Polyak and Nesterov TD(0) ODE and the corresponding discrete-time algorithms.

Recall from Chapter 3, our proposed Polyak TD(0) ODE is given as:

$$\ddot{\mathbf{w}}(t) + \beta(t)\dot{\mathbf{w}}(t) = h(\mathbf{w}(t)) = \mathbf{b} - A\mathbf{w}(t) \ ,$$

where $\beta(t) = \beta/t$. From Section 4.3, we know that we can choose $\beta \in [3,5]$. To obtain an accelerated rate of $\mathcal{O}(1/t^2)$, we choose $\beta = 3$. Hence, we can write our Polyak TD(0) ODE as:

$$\ddot{\mathbf{w}}(t) + \frac{3}{t}\dot{\mathbf{w}}(t) = h(\mathbf{w}(t)) = \mathbf{b} - A\mathbf{w}(t) \ .$$

We can therefore modify the update equations stated in Chapter 3 for the discrete-time Polyak TD(0) method to be:

$$\textbf{TDPE} = \begin{cases} \mathbf{w}_{k+1} & = \mathbf{w}_k + \eta \mathbf{z}_k \\ \mathbf{z}_{k+1} & = \left(\frac{k-3}{k}\right) \mathbf{z}_k + \eta h(\mathbf{w}_k) \end{cases} , \qquad \text{where} \qquad \beta_k = \frac{3}{\eta k} \ . \tag{5.1.1}$$

$$\textbf{TDPS} = \begin{cases} \mathbf{w}_{k+1} & = \mathbf{w}_k + \eta \mathbf{z}_k \\ \mathbf{z}_{k+1} & = \left(\frac{k+1}{k+4}\right) [\mathbf{z}_k + \eta h(\mathbf{w}_{k+1})] \end{cases} , \qquad \text{where} \qquad \beta_k = \frac{3}{\eta(k+1)} \ . \tag{5.1.2}$$

Similarly, our proposed ODE for Nesterov is given as:

$$\ddot{\mathbf{w}}(t) + 2d(t)\dot{\mathbf{w}}(t) = h(\mathbf{w}(t) + \beta(t)\dot{\mathbf{w}}(t)) = \mathbf{b} - A[\mathbf{w}(t) + \beta(t)\dot{\mathbf{w}}(t)] \ ,$$

where $d(t)$ and $\beta(t)$ are akin to damping parameters that are dependent on time. From Section 4.4, we get the parameters to be:

$$d(t) = \frac{3}{2t}, \quad \text{and} \quad \beta(t) = \frac{t-3}{t} \ .$$

This gives us the Nesterov TD(0) to take the following specific form:

$$\ddot{\mathbf{w}}(t) + \left(\frac{3}{t}\right) \dot{\mathbf{w}}(t) = h(\mathbf{w}(t) + \left(\frac{t-3}{t}\right) \dot{\mathbf{w}}(t)) = \mathbf{b} - A\left[\mathbf{w}(t) + \left(\frac{t-3}{t}\right) \dot{\mathbf{w}}(t)\right] \ .$$

We can therefore modify the update equation stated in Chapter 3 for the discrete-time Nesterov methods as:

$$\textbf{TDNE} = \begin{cases} \mathbf{w}_{k+1} & = \mathbf{w}_k + \eta \mathbf{z}_k \\ \mathbf{z}_{k+1} & = \eta \left(\frac{k\eta+4-3\eta}{k\eta+4}\right) \mathbf{z}_k - \eta \left(\frac{k\eta+1}{k\eta+4}\right) A\mathbf{z}_k \\ & \qquad +\eta h(\mathbf{w}_k) + \eta h(\mathbf{w}_k + \left(\frac{k\eta+1}{k\eta+4}\right)) \end{cases} , \text{where} \quad \beta_k = \frac{\eta k + 1}{\eta k + 4}, d_k = \frac{3}{2(\eta k + 4)} \ . \tag{5.1.3}$$

$$\textbf{TDNS} = \begin{cases} \mathbf{w}_{k+1} & = \mathbf{w}_k + \eta \mathbf{z}_k \\ \mathbf{z}_{k+1} & = \left[\left(\frac{\eta(k+1)+4+3\eta}{\eta(k+1)+4}\right) I + \eta \left(\frac{\eta(k+1)+1}{\eta(k+1)+4}\right) A\right]^{-1} \\ & \qquad [\eta \mathbf{z}_k + \eta h(\mathbf{w}_{k+1})] \end{cases} , \beta_k = \frac{\eta(k+1)+1}{\eta(k+1)+4}, d_k = \frac{3}{2(\eta(k+1)+4)} \ . \tag{5.1.4}$$

These specific update equations will be used in our experiments. Note that in our discrete-time Nesterov methods, for the update equations of $\mathbf{z}_{k+1}$, we had to multiply the previous iterate, $\mathbf{z}_k$ with an additional learning rate. This facilitated the ability to use a wider range of step sizes. We will explain this choice further when discussing the results seen in our experimental set-up. We will now describe our experimental set-up.

5.2. Prediction Problems

We report the performance of Polyak TD(0) and Nesterov TD(0) across small linear prediction tasks. We use the exact experimental setup used in prior works of Sutton et al.

[2009] and Ghiassian et al. [2020]. For the sake of completeness, we describe the setup here in detail.

We work on four small linear problems: three random-walk problems and a Boyan-chain problem (Boyan [2002]). All of these problems are episodic, undiscounted, and have only on-policy training with a fixed policy.

The three random walk problems are essentially a five-state random walk MDP with three different feature representations: tabular, inverted, and dependent features. These problems are based on the standard Markov Chain proposed in Sutton [1988] where we have a linear arrangement with five states and two absorbing terminal states. All episodes start with the same initial state, the center state of the five, and then transition randomly with equal probability to a neighboring state until a terminal state is reached. The rewards are zero everywhere except on transition into the right terminal state, upon which the reward assigned is +1.

We will now describe the three feature representations that are often taken in these experiments. The first feature representation is tabular features. These characterize the lookup-table setting which we are all familiar with. The second representation is called inverted features which have classically been chosen to cause extensive inappropriate generalization between states. The third representation called dependent features uses fewer features than are sufficient to solve the problem exactly. For more details on each individual experiment, check out Sutton et al. [2009].

The fourth problem we work on is the classic Boyan-chain problem (Boyan [2002]) which is a standard episodic task often used in existing literature for comparing different TD algorithms with linear function approximation. Here, we have a 13-states Markov chain where each state is represented by a compact feature representation. While this encoding causes inappropriate generalization during learning, $\mathbf{v}_\pi$ can be represented perfectly with the given features (Ghiassian et al. [2020]).

5.3. Results

In this section, we report the empirical performance of our proposed algorithms and compare them with some existing TD algorithms. To benchmark the performance of our methods, we use the following algorithms as baselines:

- **TD**: (Sutton [1988])
- **GTD2**: (Sutton et al. [2009])
- **HTD**: (Hackman [2013])
- **VTrace**: (Espeholt et al. [2018])
- **TDRC**: (Ghiassian et al. [2020])
- **TDC**: (Sutton et al. [2009])

Note that we use the acronyms introduced in Chapter 3 for the four proposed algorithms: TDPE, TDPS, TDNE, and TDNS.

We run the baselines along with our proposed algorithms on four different problems which were described in Section (2.6). We first provide plots that show the sensitivity of the RMSPBE reported by different methods to the step size parameter. Note that the choice of the step size parameter critically influences the performances of all the methods. We vary the step size parameter η in powers of 2, i.e., $\eta = 2^{-x}$ where the choice of $x \in \{8,7,6,5,4,3,2,1\}$ varies depending on the problem setting. All the baselines that we compare with have some tunable parameters. We use the best-reported parameters from the work of Ghiassian et al. [2020]. For the proposed algorithms, we just have one hyperparameter, i.e., the step size η. While our accelerated methods do include other damping parameters, they have been derived analytically using the conservation law analysis, and hence, do not need any tuning. Note that for Nesterov's accelerated version, we rely on a heuristic way of discretization proposed by Muehlebach and Jordan [2019] that allows us to define a relationship between damping parameters. This constrains the hyperparameter search space, making it easier to arrive at stable solutions by simply tuning the step size. Hence, the step size is an important parameter to tune and determines how slow or fast we traverse down the loss landscape. We observe that the proposed accelerated methods perform best for higher values of the step size parameter, while other methods typically require smaller step sizes to achieve comparable performances. This demonstrates the advantages of accelerated methods as these methods can take bigger steps and converge faster towards the optimal value.

Following this, we note the best step size choices for each method from the above parameter sensitivity plot and use them to report the averaged RMSPBE over 100 independent runs. For each run, all the methods complete 5,000 steps. We sample RMSPBE every 50 steps and observe the convergence curves of each method (as given in (c) of every figure).

One of the most important observations here is the tradeoff between speed of convergence and accuracy. This tradeoff often appears in the setting of stochastic optimization as well. Generally, we observe that our methods enjoy a noticeably faster decrease in the RMSPBE value, however, it does end up converging to a slightly larger neighborhood than is otherwise possible with the other methods. This indicates the existence of a tradeoff: Are we okay with tolerating a slightly higher error range to get a much faster method? This tradeoff is captured in the Relative RMSPBE plot (as given in part (b) of every figure). In this plot, we report the normalized average area under the RMSPBE learning curve relative to TDNS for each method. If the tradeoff between the speed of convergence or the RMSPBE value is not well-balanced, we would see the relative RMSPBE bar shoot up in comparison to the other methods. This helps us to effectively compare the relative performance of baselines as well as our proposed methods compared to TDNS.

Across the various problem settings, we observe that the gain in speed attained due to a faster method is usually worth the slight increase in the size of the neighborhood of values we converge to. This is especially true in all of the Random Walk environments where the proposed methods are the best possible methods to go with for a well-balanced tradeoff between speed of convergence and accuracy. The one environment where our methods seem to struggle with this tradeoff is the Boyan chain problem.

The methods that seem to perform the best in the Boyan chain problem are GTD2 and TDC, where we observe a fast reduction in the RMSPBE values as shown in Figure 5.1. This could be attributed to the fact that our proposed methods and other baselines suffer from the variance in h. The features in this environment cause bigger changes in h than in the other environments. Since we use the concept of momentum to bias our methods to past iterations, this causes our methods to suffer greatly from the variance observed in h. However, we see that our proposed methods perform better than TD and TDRC. Hence, our methods still manage to fall in the middle of the two extreme groups, proving that we are still relatively performing better than the worst methods. Note that all the methods do eventually converge to a similar neighborhood of the RMSPBE value.

For the Random Walk environment with Tabular feature representations shown in Figure 5.2 (c), we again notice that there isn't a considerable difference in the optimal value achieved by most methods. We do observe, however, that our methods reached these values faster than other methods. While TDNS and TDPS went fast, they managed to stay at that value for the rest of the runs. TDNE on the other hand, initially showed the fastest descent down the curve, but then ended up converging to an RMSPBE value slightly higher than the optimal value achieved by the other methods. However, this increase in the error is almost negligible in this case, as is also visible in the Relative RMSPBE curve.

This effect is more pronounced in the random walk environments with Inverted and Dependent feature representations. In the case of both these representations as seen in Figure 5.3 (c) and Figure 5.4 (c), we see a similar trend for our proposed methods. TDPE, TDPS, and TDNE go down the fastest but converge to a bigger neighborhood, i.e., they do not arrive at the most optimal value of RMSPBE possible. TDNS converges a bit slower but also ends up arriving and converging to the same bigger neighborhood as other proposed methods.

However, in the case of inverted feature representations, many methods converge to better values of RMSPBE, especially TDRC, albeit taking a lot more steps than our proposed accelerated methods. In Figure 5.3 (b), we can see that the proposed methods are able to converge a lot faster than other methods if we are willing to tolerate the associated increase in the neighborhood of RMSPBE values that we end up in.

In the case of Dependent feature representations as seen in Figure 5.4 (c), we observe that other baselines, like TD, TDRC, and HTD are better in terms of the RMSPBE value

achieved than our methods. But the speed of reduction in the RMSPBE value observed in the case of TDPE, TDPS, and TDNE is drastic and there is indeed not much difference in the neighborhood of RMSPBE value it ends up converging to at the end of 100 steps. This can be seen by looking at the Relative RMSPBE bars in Figure 5.4 (b).

We will now discuss the computational complexity of our proposed methods. We observe that TDNE performs the best compared to all other algorithms in terms of relative RMSPBE in all four problems. However, in terms of convergence, it is slightly slower as compared to other algorithms. However, this negligibly slow convergence is balanced by the gains in computational complexity. In terms of computational time, it is slightly more expensive than TDPS and TDPE as the Nesterov-based method requires an extra matrix-vector product computation as compared to Polyak. However, compared to TDNS, it is less expensive because TDNS requires the computation of matrix inverse (see Section 3.4). In most cases, it would be advisable to go with TDNE method if the user has decided to tolerate a slightly slower method.

Note that while TDNS shows comparable performance to other baselines for the Relative RMSPBE, its performance compared to our other proposed algorithms is not good. In terms of convergence, it seems to be slower than the other methods and similar to TDNE. Computationally, it seems to be the most expensive method as it involves the computation of an inverse. So, it would generally not be advisable to go for this approach when we have other accelerated methods which are computationally better.

TDPE and TDPS are very similar to each other in terms of Relative RMSPBE as well as convergence. They perform better in terms of Relative RMSPBE and are on par in terms of convergence compared to most of the baselines. They are faster in convergence compared to TDNE and TDNS and on par with TDNE in terms of Relative RMSPBE. Therefore, out of all the four proposed algorithms, TDPE and TDPS perform the best (i.e. both in terms of Relative RMSPBE as well as faster convergence). We can therefore say that these algorithms handle the tradeoff between speed of convergence and accuracy the best.

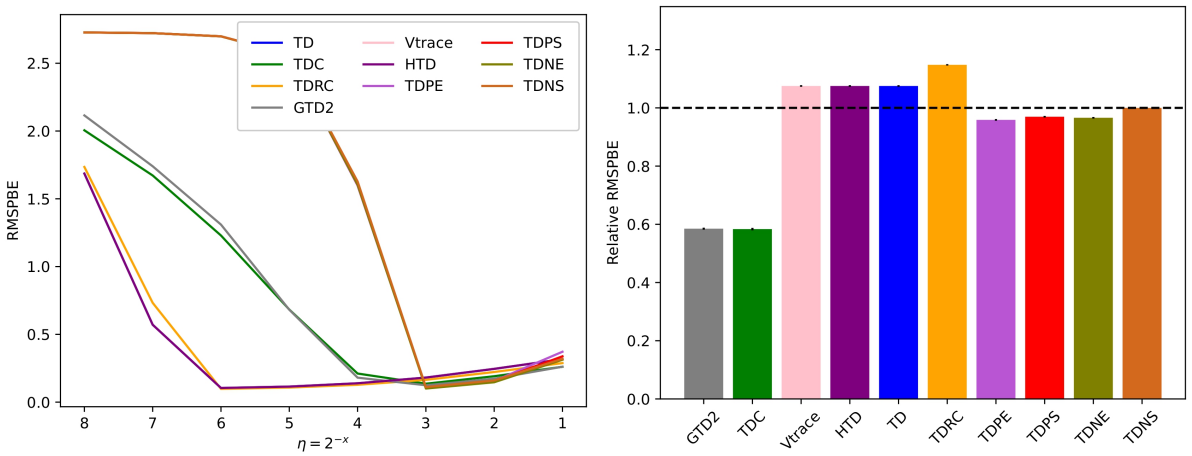
These observations fall in line with the general fact that Nesterov is often known to not perform well with stochastic objectives. Note that we had to introduce an additional learning rate in the update equations to stabilize the dynamics and allow for the exploration of a wider range of step size values. Specifically, the behavior observed with symplectic discretization is not surprising. While our symplectic discretization doesn't exactly preserve the symplectic structure, we see a similar trend to the one observed by Betancourt [2015]. This could be attributed to the fact that stochastic variations in the objective incur a bias in the symplectic integrator which reduces their accuracy by deviating the numerical approximations away from the true dynamics Betancourt et al. [2018]. Intuitively, we can say that the dynamical evolution moves so quickly that the variations in stochastic objective doesn't have sufficient

time to average out. This explains the relative success of slower methods, like Robbins-Monro in the case of stochastic objectives.

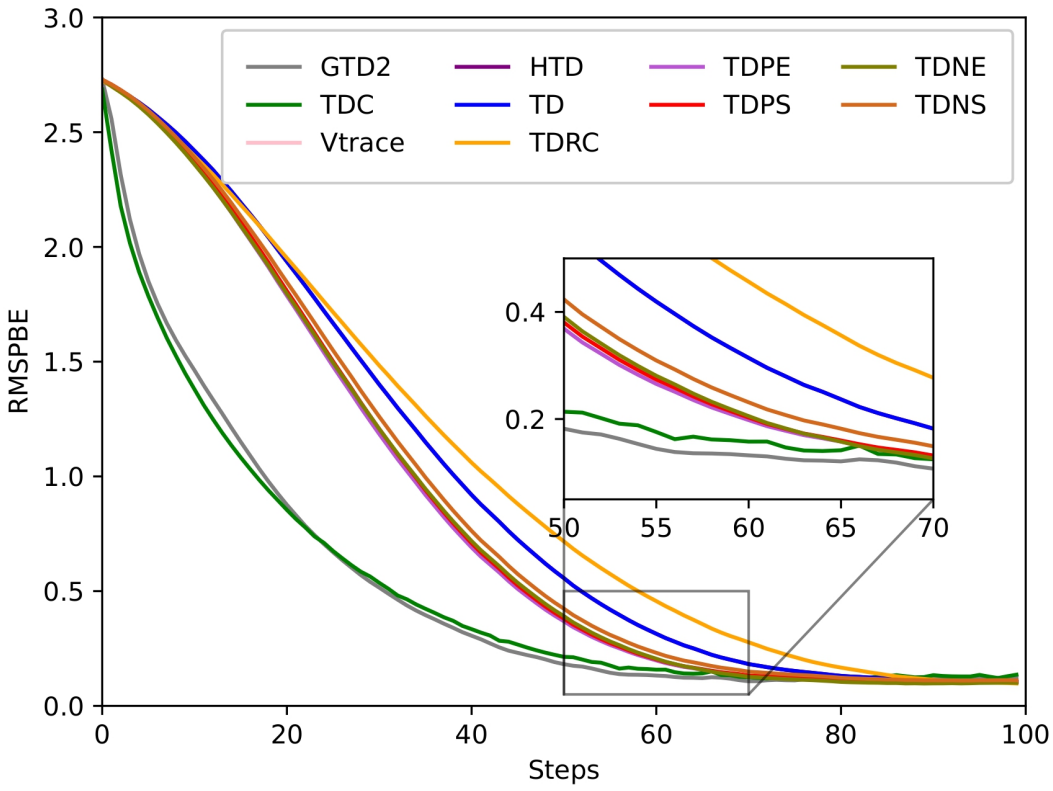
In the hyperparameter study for the step sizes, we observed that increasing the step sizes for our proposed accelerated methods resulted in a steep increase in the RMSPBE value that we end up converging to. Hence, the performance and stability of the proposed methods depend critically on the choice of the step size parameter. In other words, we can go faster (up to a limit in the discrete setting), but doing so adversely affects the stability of these systems. In such cases, the solution seemed to be diverging, suggesting that we achieved acceleration at the cost of stability. This indicates that although we can go faster, we might end up in a situation where we do not arrive at a solution or converge to a larger neighborhood than would otherwise be possible.

Note that we do not observe the $\mathcal{O}(1/t^2)$ rate observed in the convergence rate analysis done in Chapter 4. This is because the accelerated rates obtained correspond to the continuous-time system where we can traverse the curve at any speed. However, the discrete-time system has a bottleneck in how fast we can go down the curve stably. To translate these rates to the discrete-time system, we would have to construct symplectic integration schemes that preserve the underlying symplectic structure of our dynamical system, thereby preserving our rates. However, our accelerated algorithms do see an increase in the speed of convergence.

In future work, we would like to investigate other discretization schemes that would preserve the accelerated rates obtained in the continuous-time systems. One interesting line of work would be to explore symplectic integration that considers the geometry of the underlying dissipative system by taking a Hamiltonian perspective on discretization (Diakonikolas and Jordan [2021], França et al. [2021]). These discretization schemes would allow our methods to be more robust to step size schedules and therefore, allow for a higher range of step size parameters.

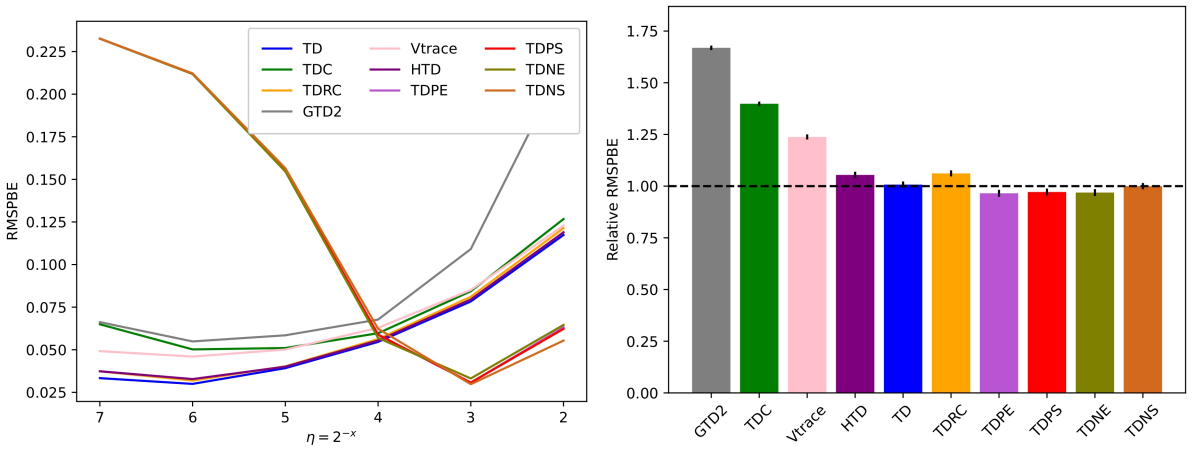


(a) **(b)**



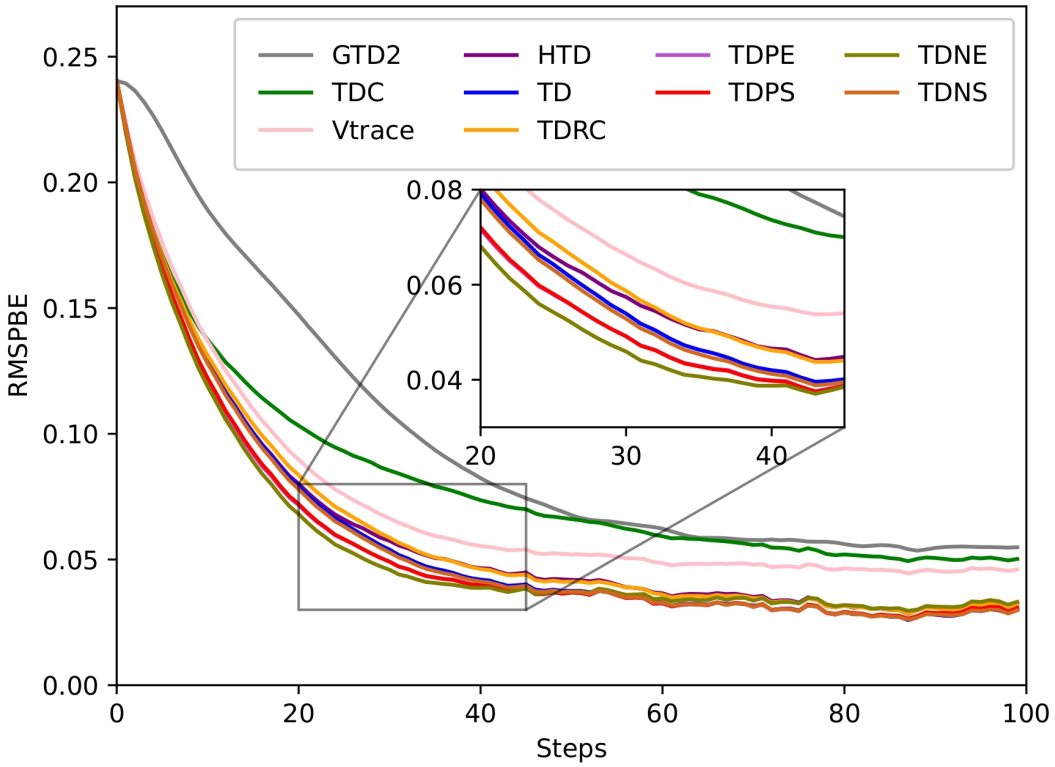
(c)

Fig. 5.1. **(a)**: Step-size sensitivity of RMSPBE for each method on the Boyan-chain problem. Here, η is the step size that varies in powers of 2, i.e., $\eta = 2^{-x}$. **(b)**: The normalized average area under the RMSPBE learning curve. The step sizes for each method are set such that each method achieves the best possible RMSPBE. Each bar is normalized by TDNS's performance so that each problem can be shown in the same range. All results are averaged over 100 independent runs with standard error bars shown at the top of each rectangle. The black horizontal line demonstrates the performance of TDNS for easier comparison between other methods. **(c)**: RMSPBE vs steps for each method. Note that the step-size parameters are tuned according to the recommendations made by the step-size sensitivity curve shown in **(a)**.



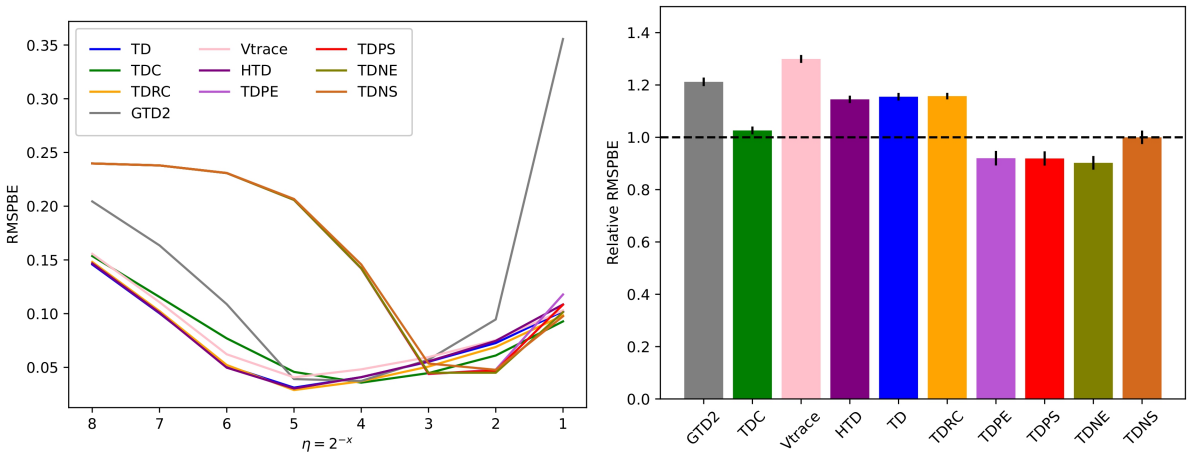
(a)

(b)



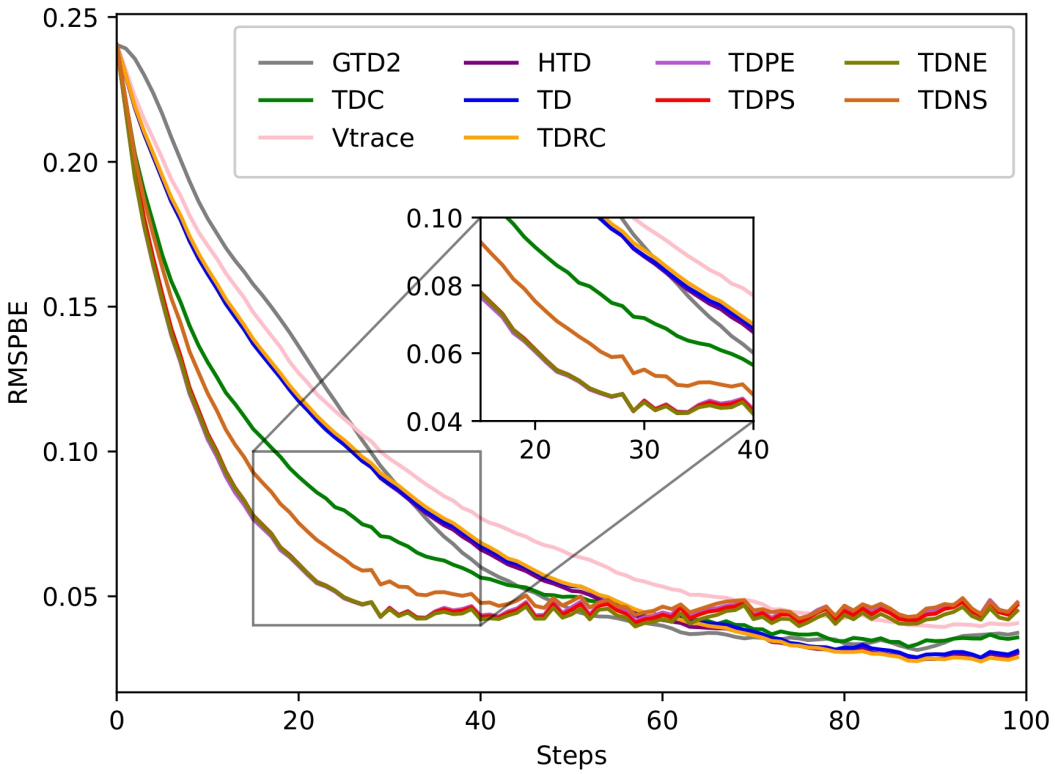
(c)

Fig. 5.2. **(a)**: Step-size sensitivity of RMSPBE for each method on Random Walk with Tabular feature representation. Here, η is the step size that varies in powers of 2, i.e., $\eta = 2^{-x}$. **(b)**: The normalized average area under the RMSPBE learning curve. The step sizes for each method are set such that each method achieves the best possible RMSPBE. Each bar is normalized by TDNS's performance so that each problem can be shown in the same range. All results are averaged over 100 independent runs with standard error bars shown at the top of each rectangle. The black horizontal line demonstrates the performance of TDNS for easier comparison between other methods. **(c)**: RMSPBE vs steps for each method. Note that the step-size parameters are tuned according to the recommendations made by the step-size sensitivity curve shown in **(a)**.



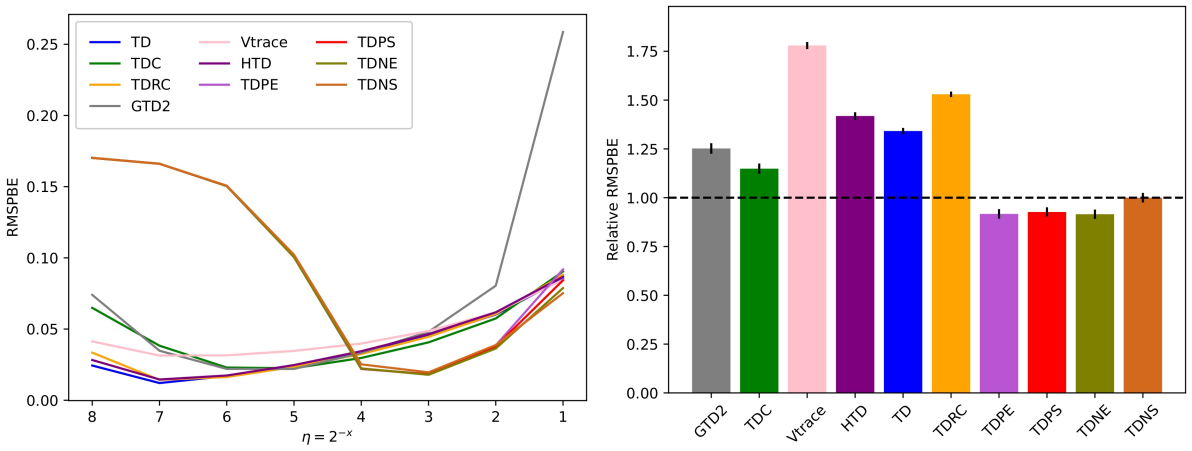
(a)

(b)

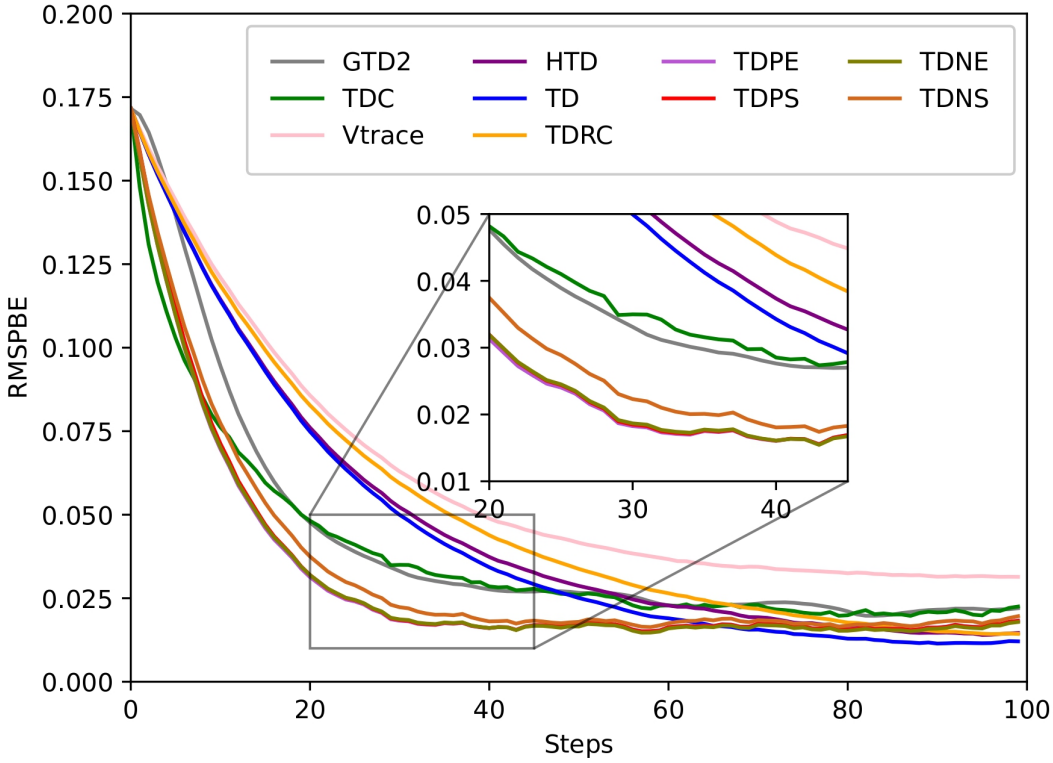


(c)

Fig. 5.3. **(a)**: Step-size sensitivity of RMSPBE for each method on Random Walk with Inverted feature representation. Here, η is the step size that varies in powers of 2, i.e., $\eta = 2^{-x}$. **(b)**: The normalized average area under the RMSPBE learning curve. The step sizes for each method are set such that each method achieves the best possible RMSPBE. Each bar is normalized by TDNS's performance so that each problem can be shown in the same range. All results are averaged over 100 independent runs with standard error bars shown at the top of each rectangle. The black horizontal line demonstrates the performance of TDNS for easier comparison between other methods. **(c)**: RMSPBE vs steps for each method. Note that the step-size parameters are tuned according to the recommendations made by the step-size sensitivity curve shown in **(a)**.



(a) (b)



(c)

Fig. 5.4. **(a)**: Step-size sensitivity of RMSPBE for each method on Random Walk with Dependent feature representation. Here, η is the step size that varies in powers of 2, i.e., $\eta = 2^{-x}$. **(b)**: The normalized average area under the RMSPBE learning curve. The step sizes for each method are set such that each method achieves the best possible RMSPBE. Each bar is normalized by TDNS's performance so that each problem can be shown in the same range. All results are averaged over 100 independent runs with standard error bars shown at the top of each rectangle. The black horizontal line demonstrates the performance of TDNS for easier comparison between other methods. **(c)**: RMSPBE vs steps for each method. Note that the step-size parameters are tuned according to the recommendations made by the step-size sensitivity curve shown in **(a)**.

Chapter 6

Conclusions and Future Work

This thesis is centered around exploring the concept of acceleration and momentum in the setting of temporal difference methods. Accelerated methods are known to play a central role in optimization, achieving the optimal rate in many settings. Although momentum is a heavily studied and well-understood concept in optimization, it doesn't directly translate to semi-gradient TD methods because TD methods do not minimize an objective by performing gradient descent (Barnard [1993]). Hence, this concept is often under appreciated in the RL setting. Few works exploring acceleration in TD methods have done this in the setting of gradient TD methods (Sutton et al. [2009], Meyer et al. [2014], Pan et al. [2017], Ghiassian et al. [2020], Deb and Bhatnagar [2022]). The introduction of gradient descent in the TD objective makes the process of adding momentum similar to the one seen in optimization which is easier to work with, however, it happens to be a very limiting viewpoint of TD methods. Hence, there is a need to come up with the right counterpart to momentum in the RL setting.

We take another approach and attempt to introduce acceleration in temporal difference methods from first principles. We hypothesize that this could serve as the true counterpart to acceleration in the SA setting. Specifically, inspired by the inception of accelerated algorithms in optimization using the dynamical systems approach (Polyak [1964]), we extend this recipe to intuitively derive accelerated algorithms in SA. Note that the dynamical systems is one of the primary method for analysis in SA (Borkar and Meyn [2000]) and has been used for designing new algorithms (Meyn [2022]). This thesis aims to understand the acceleration phenomenon for TD while advocating for the dynamical systems approach for designing new algorithms. We introduce two second-order ODEs for TD(0) with linear function approximation: Polyak TD(0) and Nesterov TD(0), which are the 'correct' momentum counterparts. We also discretize these ODEs using two discretization schemes: explicit Euler, and symplectic Euler to give us four different accelerated algorithms for TD(0). We

show that these accelerated algorithms have comparable results to existing TD algorithms in linear prediction tasks.

Inspired by Suh et al. [2022]'s methodology for analyzing continuous-time versions of accelerated gradient methods in dilated coordinates, we extend this framework to temporal difference methods. This is the main contribution of this thesis: it allows us to establish the convergence of our methods along with providing their continuous-time rates.

We obtain novel continuous-time rates for the proposed accelerated methods. Note that deriving convergence rates in any setting are known to be complex, often requiring the need to come up with particular Lyapunov functions. We provide an intuitive and fundamental way of deriving rates for SA methods. Another major advantage of this methodology is that it also provides a recommendation for the choice of the damping/momentum parameter to get an accelerated rate. Note that without these conservation laws to guide the choice of these damping hyperparameters, we would have to manually control and optimize them. This, in turn, would make the process of obtaining better convergent results more cumbersome. In addition to that, the introduction of a general dilated coordinate framework can further be used to derive different conservation laws to better understand the behavior and properties of TD methods.

6.1. Limitations and Future Work

The rates derived from the continuous-time analysis framework might not always translate to the discrete-time system as is seen in the experiments we conducted. Working with the continuous-time dynamical system allows us to traverse the curve at any speed, whereas this breaks when we move into the discrete-time system. A major line of future work is to explore more discretization schemes that would preserve the convergence rates obtained in the continuous-time dynamical system. Specifically, we would like to explore symplectic integration schemes that consider the geometry of the underlying dissipative system. A Hamiltonian perspective on discretization would focus on the conserved quantities of the dissipative system and allow us to think about how to use these quantities to design more robust and stable discrete accelerated methods (Diakonikolas and Jordan [2021], França et al. [2021]).

We would also need to perform detailed experiments for more complex problems, settings, and environments. We would specifically like to explore control problems and extend the concept of acceleration to the non-linear setting of Q-Learning. This would allow us to investigate how accelerated algorithms fare in non-linear settings and if acceleration proves to be beneficial in such settings. Another line of work could be to extend this concept of acceleration to TD(λ). Note that momentum also has an update that is similar to the

eligibility trace update and hence it would be interesting to understand its interplay with classical accumulating traces.

Note that accelerated methods in gradient-based optimization can be seen as instances of a single underlying framework, called the Bregman Lagrangian (Wibisono et al. [2016]). The ability to do so has allowed for the development of a unifying framework to analyze and characterize their physical behavior. An interesting line of work would be to develop accelerated methods in TD learning with the viewpoint of them being instances of a single underlying concept.

While the ODE approach has long been used in RL for performing stability and convergence analysis, the field is yet to embrace its full potential. With this thesis, we hope to motivate the understanding of acceleration from first principles in the RL setting. To this end, we design a framework to create such accelerated methods and further study their properties.

References

Samir Adly and Hedy Attouch. First-order inertial algorithms involving dry friction damping. *Mathematical Programming*, 193(1):405–445, 2022.

Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2): 251–276, 1998.

Hedy Attouch and Felipe Alvarez. The heavy ball with friction dynamical system for convex constrained minimization problems. In *Optimization*, pages 25–35. Springer, 2000.

Hedy Attouch and Alexandre Cabot. Convergence rates of inertial forward-backward algorithms. *SIAM Journal on Optimization*, 28(1):849–874, 2018.

Hedy Attouch, Xavier Goudou, and Patrick Redont. The heavy ball with friction method, i. the continuous dynamical system: global exploration of the local minima of a real-valued function by asymptotic analysis of a dissipative dynamical system. *Communications in Contemporary Mathematics*, 2(01):1–34, 2000.

Hedy Attouch, Alexandre Cabot, Zaki Chbani, and Hassan Riahi. Inertial forward–backward algorithms with perturbations: Application to tikhonov regularization. *Journal of Optimization Theory and Applications*, 179(1):1–36, 2018a.

Hedy Attouch, Zaki Chbani, Juan Peypouquet, and Patrick Redont. Fast convergence of inertial dynamics and algorithms with asymptotic vanishing viscosity. *Mathematical Programming*, 168(1):123–175, 2018b.

Konstantin Avrachenkov, Kishor Patil, and Gugan Thoppe. Online algorithms for estimating change rates of web pages. *Performance Evaluation*, 153:102261, 2022.

Mohammad Gheshlaghi Azar, Remi Munos, M Ghavamzadaeh, and Hilbert J Kappen. Speedy q-learning. 2011.

Etienne Barnard. Temporal-difference methods and markov models. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(2):357–365, 1993.

Michel Benaïm. Dynamics of stochastic approximation algorithms. In *Seminaire de probabilites XXXIII*, pages 1–68. Springer, 1999.

Emmanuel Bengio, Joelle Pineau, and Doina Precup. Correcting momentum in temporal difference learning. *arXiv preprint arXiv:2106.03955*, 2021.

Albert Benveniste, Michel Métivier, and Pierre Priouret. *Adaptive algorithms and stochastic approximations*, volume 22. Springer Science & Business Media, 2012.

Michael Betancourt, Michael I Jordan, and Ashia C Wilson. On symplectic optimization. *arXiv preprint arXiv:1802.03653*, 2018.

MJ Betancourt. The fundamental incompatibility of hamiltonian monte carlo and data subsampling. *arXiv preprint arXiv:1502.01510*, 2015.

Jalaj Bhandari, Daniel Russo, and Raghav Singal. A finite time analysis of temporal difference learning with linear function approximation. In *Conference on learning theory*, pages 1691–1692. PMLR, 2018.

Jiri Blazek. *Computational fluid dynamics: principles and applications*. Butterworth-Heinemann, 2015.

Vivek S Borkar. *Stochastic approximation: a dynamical systems viewpoint*, volume 48. Springer, 2009.

Vivek S Borkar and Sean P Meyn. The ode method for convergence of stochastic approximation and reinforcement learning. *SIAM Journal on Control and Optimization*, 38(2): 447–469, 2000.

Justin A Boyan. Technical update: Least-squares temporal difference learning. *Machine learning*, 49(2):233–246, 2002.

Sébastien Bubeck, Yin Tat Lee, and Mohit Singh. A geometric alternative to nesterov's accelerated gradient descent. *arXiv preprint arXiv:1506.08187*, 2015.

Shuhang Chen, Adithya M Devraj, Fan Lu, Ana Busic, and Sean Meyn. Zap q-learning with nonlinear function approximation. *Advances in Neural Information Processing Systems*, 33:16879–16890, 2020.

Shuhang Chen, Adithya Devraj, Andrey Bernstein, and Sean Meyn. Accelerating optimization and reinforcement learning with quasi stochastic approximation. In *2021 American Control Conference (ACC)*, pages 1965–1972. IEEE, 2021.

Gal Dalal, Balázs Szörényi, Gugan Thoppe, and Shie Mannor. Finite sample analyses for td (0) with function approximation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018a.

Gal Dalal, Gugan Thoppe, Balázs Szörényi, and Shie Mannor. Finite sample analysis of two-timescale stochastic approximation with applications to reinforcement learning. In *Conference On Learning Theory*, pages 1199–1233. PMLR, 2018b.

Rohan Deb and Shalabh Bhatnagar. Gradient temporal difference with momentum: Stability and convergence. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 6488–6496, 2022.

Adithya M Devraj and Sean Meyn. Zap q-learning. *Advances in Neural Information Processing Systems*, 30, 2017.

Adithya M Devraj, Ana Bušić, and Sean Meyn. On matrix momentum stochastic approximation and applications to q-learning. In *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 749–756. IEEE, 2019.

AM Devraj. *Reinforcement learning design with optimal learning rate*. PhD thesis, PhD thesis, University of Florida, 2019.

Jelena Diakonikolas and Michael I Jordan. Generalized momentum-based methods: A hamiltonian perspective. *SIAM Journal on Optimization*, 31(1):915–944, 2021.

Yoel Drori and Marc Teboulle. Performance of first-order methods for smooth convex minimization: a novel approach. *Mathematical Programming*, 145(1):451–482, 2014.

Dmitriy Drusvyatskiy, Maryam Fazel, and Scott Roy. An optimal first order method based on optimal quadratic averaging. *SIAM Journal on Optimization*, 28(1):251–271, 2018.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, pages 1407–1416. PMLR, 2018.

Amir-Massoud Farahmand and Mohammad Ghavamzadeh. Pid accelerated value iteration algorithm. In *International Conference on Machine Learning*, pages 3143–3153. PMLR, 2021.

Guilherme França, Jeremias Sulam, Daniel Robinson, and René Vidal. Conformal symplectic and relativistic optimization. *Advances in Neural Information Processing Systems*, 33: 16916–16926, 2020.

Guilherme França, Michael I Jordan, and René Vidal. On dissipative symplectic integration with applications to gradient-based optimization. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(4):043402, 2021.

Sina Ghiassian, Andrew Patterson, Shivam Garg, Dhawal Gupta, Adam White, and Martha White. Gradient temporal-difference learning with regularized corrections. In *International Conference on Machine Learning*, pages 3524–3534. PMLR, 2020.

Leah M Hackman. Faster gradient-td algorithms. 2013.

Ernst Haier, Christian Lubich, and Gerhard Wanner. *Geometric Numerical integration: structure-preserving algorithms for ordinary differential equations*. Springer, 2006.

Bin Hu and Laurent Lessard. Dissipativity theory for nesterov's accelerated method. In *International Conference on Machine Learning*, pages 1549–1557. PMLR, 2017.

Chi Jin, Praneeth Netrapalli, and Michael I Jordan. Accelerated gradient descent escapes saddle points faster than gradient descent. In *Conference On Learning Theory*, pages 1042–1085. PMLR, 2018.

Michael I Jordan. Dynamical, symplectic and stochastic perspectives on gradient-based optimization. In *Proceedings of the International Congress of Mathematicians: Rio de Janeiro 2018*, pages 523–549. World Scientific, 2018.

Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

Walid Krichene, Alexandre Bayen, and Peter L Bartlett. Accelerated mirror descent in continuous and discrete time. *Advances in neural information processing systems*, 28, 2015.

Harold Kushner and G George Yin. *Stochastic approximation and recursive algorithms and applications*, volume 35. Springer Science & Business Media, 2003.

Chandrashekar Lakshminarayanan and Csaba Szepesvari. Linear stochastic approximation: How far does constant step-size and iterate averaging go? In *International Conference on Artificial Intelligence and Statistics*, pages 1347–1355. PMLR, 2018.

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.

Bo Liu, Ji Liu, Mohammad Ghavamzadeh, Sridhar Mahadevan, and Marek Petrik. Proximal gradient temporal difference learning algorithms. In *IJCAI*, pages 4195–4199, 2016.

Chris J Maddison, Daniel Paulin, Yee Whye Teh, Brendan O'Donoghue, and Arnaud Doucet. Hamiltonian descent methods. *arXiv preprint arXiv:1809.05042*, 2018.

Dominik Meyer, Rémy Degenne, Ahmed Omrane, and Hao Shen. Accelerated gradient temporal difference learning algorithms. In *2014 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, pages 1–8. IEEE, 2014.

Sean Meyn. *Control Systems and Reinforcement Learning*. Cambridge University Press, 2022.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Wenlong Mou, Chris Junchi Li, Martin J Wainwright, Peter L Bartlett, and Michael I Jordan. On linear stochastic approximation: Fine-grained polyak-ruppert and non-asymptotic concentration. In *Conference on Learning Theory*, pages 2947–2997. PMLR, 2020.

Michael Muehlebach and Michael Jordan. A dynamical systems perspective on nesterov acceleration. In *International Conference on Machine Learning*, pages 4656–4662. PMLR, 2019.

Michael Muehlebach and Michael I Jordan. Optimization with momentum: Dynamical, control-theoretic, and symplectic perspectives. *J. Mach. Learn. Res.*, 22(73):1–50, 2021.

Y Nesterov. A method of solving a convex programming problem with convergence rate mathcal {O}(1/k^ {2}). In *Sov. Math. Dokl*, volume 27.

Yu Nesterov. Smooth minimization of non-smooth functions. *Mathematical programming*, 103(1):127–152, 2005.

Yangchen Pan, Adam White, and Martha White. Accelerated gradient temporal difference learning. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics*, 4(5):1–17, 1964.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

Bin Shi, Simon S Du, Weijie Su, and Michael I Jordan. Acceleration via symplectic discretization of high-resolution differential equations. *Advances in Neural Information Processing Systems*, 32, 2019.

Bin Shi, Simon S Du, Michael I Jordan, and Weijie J Su. Understanding the acceleration phenomenon via high-resolution differential equations. *Mathematical Programming*, 195 (1):79–148, 2022.

Rayadurgam Srikant and Lei Ying. Finite-time error bounds for linear stochastic approximation andtd learning. In *Conference on Learning Theory*, pages 2803–2830. PMLR, 2019.

Weijie Su, Stephen Boyd, and Emmanuel Candes. A differential equation for modeling nesterov's accelerated gradient method: theory and insights. *Advances in neural information processing systems*, 27, 2014.

Jaewook J Suh, Gyumin Roh, and Ernest K Ryu. Continuous-time analysis of accelerated gradient methods via conservation laws in dilated coordinate systems. In *International Conference on Machine Learning*, pages 20640–20667. PMLR, 2022.

Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

Richard S Sutton, Hamid Maei, and Csaba Szepesvári. A convergent $o(n)$ temporal-difference algorithm for off-policy learning with linear function approximation. *Advances in neural information processing systems*, 21, 2008.

Richard S Sutton, Hamid Reza Maei, Doina Precup, Shalabh Bhatnagar, David Silver, Csaba Szepesvári, and Eric Wiewiora. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *Proceedings of the 26th annual international conference on machine learning*, pages 993–1000, 2009.

Csaba Szepesvári. The asymptotic convergence-rate of q-learning. *Advances in neural information processing systems*, 10, 1997.

Paul Tseng. On accelerated proximal gradient methods for convex-concave optimization. *submitted to SIAM Journal on Optimization*, 2(3), 2008.

John Tsitsiklis and Benjamin Van Roy. Analysis of temporal-diffference learning with function approximation. *Advances in neural information processing systems*, 9, 1996.

Gerhard Wanner and Ernst Hairer. *Solving ordinary differential equations II*, volume 375. Springer Berlin Heidelberg New York, 1996.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.

Christopher John Cornish Hellaby Watkins. Learning from delayed rewards. 1989.

Andre Wibisono and Ashia C Wilson. On accelerated methods in optimization. *arXiv preprint arXiv:1509.03616*, 2015.

Andre Wibisono, Ashia C Wilson, and Michael I Jordan. A variational perspective on accelerated methods in optimization. *proceedings of the National Academy of Sciences*, 113(47):E7351–E7358, 2016.

Ashia C Wilson, Lester Mackey, and Andre Wibisono. Accelerating rescaled gradient descent: Fast optimization of smooth functions. *Advances in Neural Information Processing Systems*, 32, 2019.

Ashia C Wilson, Ben Recht, and Michael I Jordan. A lyapunov analysis of accelerated methods in optimization. *J. Mach. Learn. Res.*, 22:113–1, 2021.

Samuel Yang-Zhao et al. Sufficient conditions for divergence in projected bellman equation methods. 2018.

Jingzhao Zhang, Suvrit Sra, and Ali Jadbabaie. Acceleration in first order quasi-strongly convex optimization by ode discretization. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 1501–1506. IEEE, 2019.