Stochastic Variance Reduced Gradient for Policy Evaluation with Fewer Gradient Evaluations

Zilun Peng

Master of Science

School of Computer Science

McGill University

Montreal,Quebec

2019-12-15

A thesis submitted to McGill University in partial fulfillment of the requirements for the degree of Master of Science.

©Zilun Peng, 2019

ABSTRACT

Stochastic variance-reduced gradient (SVRG) is an optimization method originally designed for tackling machine learning problems with a finite sum structure. SVRG was later shown to work for policy evaluation, a problem in reinforcement learning in which one aims to estimate the value function of a given policy. SVRG makes use of gradient estimates at two scales. At the slower scale, SVRG computes a full gradient over the whole dataset, which could lead to prohibitive computation costs. In this work, we show that two variants of SVRG for policy evaluation could diminish the number of gradient calculations while preserving a linear convergence speed. Our experiments demonstrate computational savings of our methods.

ABRÉGÉ

Le gradient stochastique à réduction de variance (SVRG) est une méthode d'optimisation conçue à l'origine pour résoudre les problèmes d'apprentissage machine avec une structure à somme finie. Il a ensuite été démontré que SVRG fonctionnait pour l'évaluation des politiques, un problème d'apprentissage par renforcement dans lequel l'objectif est d'estimer la fonction de valeur d'une politique donnée. Le SVRG utilise des estimations de gradient à deux échelles. Sur une échelle plus lente, SVRG calcule un gradient complet sur l'ensemble du jeu de données, ce qui pourrait entraîner des coûts de calcul prohibitifs. Dans ce travail, nous montrons que deux variantes de SVRG pour l'évaluation des politiques pourraient réduire le nombre de calculs de gradients tout en préservant une vitesse de convergence linéaire. Nos expériences démontrent des économies de calcul de nos méthodes.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
Introduction

In Reinforcement Learning (RL), an agent continuously interacts with an environment by choosing actions as prescribed by a way of behaving called a policy. The agent observes its current state and performs an action based on its current policy (which is a probability distribution conditioned on state), then it reaches a new state and obtains a reward. The goal of the agent is to improve its policy, but a key requirement in this process is the ability to evaluate the expected long-term return of the current policy, called the value function. After evaluating the policy, the policy can be updated so that more valuable states are visited more often. Performing policy evaluation efficiently is thus imperative to the success of training a RL agent.

In Machine Learning (ML), a common task is to find the optimal solution to an objective function that is convex. Since computing the full gradient requires evaluating the entire dataset, convergent stochastic gradient methods are preferred due to their cheap computational costs, especially in situations where the size of the dataset is large. In the problem of interests here, policy evaluation with linear function approximation, the objective function is a saddle-point formulation of the empirical Mean Squared Projected Bellman Error (MSPBE). It is convex-concave and not strongly convex in the primal variable, so existing powerful convex optimization methods do not directly apply.

Despite this problem, Du et al [15] showed that SVRG and SAGA can be applied to solve the saddle-point version of MSPBE with linear convergence rates, leading to fast, convergent methods for policy evaluation. SVRG is particularly appealing when d is large, because it has low memory requirements. However, SVRG requires many gradient evaluations. In this thesis, we extend two methods, Batching SVRG and SCSG, for policy evaluation. These two methods were originally proposed to make SVRG computationally efficient for solving convex problems [17, 22], so they do not directly apply to our problem.

An important step of SVRG is to compute a full gradient at the beginning of every epoch. Subsequent stochastic gradient updates use this full gradient so that the variance of updating directions is reduced. Batching SVRG and SCSG use a batch of data to approximate the full gradient while maintaining a linear convergence rate, so they are computationally more efficient than SVRG. Our main contribution is that we prove Batching SVRG and SCSG converge linearly with respect to the number of epochs when solving the saddle-point formulation of MSPBE. This could potentially speed up policy evaluation with linear function approximation in terms of the number of gradient evaluations. Our experiment result shows that Batching SVRG and SCSG could achieve better policy evaluation and control performances than other gradient based methods in some standard benchmarks.

1.1 Related Works

Temporal difference (TD) learning [33] is a classic method for policy evaluation, which uses the Bellman equation to bootstrap the estimation process and continually update the value function. One disadvantage of TD is the data inefficiency [7]. Least Squares Temporal Difference (LSTD) method [8, 6] is a more data-efficient approach which uses the data to construct a linear system approximating the original problem, then solves this system. It also has the advantage of not requiring a learning rate parameter. However, LSTD is not computationally feasible when the number of features d is large, because it requires inverting a matrix of size $d \times d$. When d is large, stochastic gradient based approaches, such as GTD [35], GTD2 and TDC [36] are preferred because the amount of computation and storage during each update is linear in d. Compared to classical TD, these algorithms truly compute a gradient (instead of performing a fixed-point approximation which is in fact not a gradient update); as a result, they enjoy better theoretical guarantees, especially in the case of off-policy learning, in which the policy of interest for the evaluation is different from the policy generating the agent's experiences.

Liu et al [24] showed that GTD2 [36] can be interpreted as doing stochastic gradient updates on the saddle-point formulation of MSPBE. This allows various optimization techniques to be applied on accelerating GTD2, because the new objective function is convex-concave and has a finite sum structure. More specifically, SMP [20], SVRG [19] and SAGA [14] were applied to optimize the saddle-point formulation of MSPBE, which leads to new algorithms with linear convergence rates [24, 15]. Previous methods such as TD, GTD2 and TDC have slower sublinear convergence rates [5, 38, 12]. The new interpretation of MSPBE motivates finite sample analysis of GTD and GTD2 under different assumptions on the data [24, 40, 38], which is not possible under the original objective functions that GTD and GTD2 are designed for [16]. All previously mentioned methods focus on policy evaluation with linear function approximation, and saddle-point formulation of MSPBE has inspired new methods for other problems in RL, including: multi-agent RL [39, 10], off-policy evaluation [38, 42] and policy evaluation with nonlinear function approximation [11].

The idea of SVRG has been applied to RL by numerous previous works. Korda and Prashanth [21] incorporated SVRG updates into TD(0). Du et al [15] applied SVRG to optimize the saddle-point formulation of MSPBE. This work extends Du et al's work and shows that gradient evaluations of SVRG can be saved while preserving the linear convergence rate. In the control case, Xu et al [43] used SVRG in a subroutine of trust region policy optimization (TRPO) method [31]. Papini et al [26] optimized a policy by using a *SVRG policy gradient* instead of the traditional policy gradient, defined by taking the gradient with respect to the expected reward obtained by the policy [37]. The *SVRG policy gradient* is defined by snapshots of old and new policies' gradients, which follows the idea of SVRG. Additionally, *SVRG policy gradient* is incorporated with importance weights [28] because distributions of samples change as the policy evolves. Papini et al used approximations of full gradients in their updates due to cheaper computational costs, which is similar to our work, but there are two fundamental differences: (1) This work focuses on policy evaluation and our objective function is convex-concave. Papini et al proposed a policy gradient method, and their objective is non-concave (2) This work uses a linear function to approximate the value function, whereas any policy gradient estimators (e.g. REINFORCE [41]) can be plugged into Papini et al's method. These two differences result in differences in theoretical results. Papini et al's method converges to a stationary point with a sublinear rate and our methods converge to a globally optimal solution with a linear rate (under some assumptions about the data).

1.2 Thesis Contributions

This thesis is based on an Arxiv paper [27], titled "SVRG for Policy Evaluation with Fewer Gradient Evaluations", with Ahmed Touati, Pascal Vincent and Doina Precup. Some contents of Chapter 1, 2 and 3 are overlapped with [27] and were edited by Ahmed Touati, Doina Precup and Pascal Vincent. The proof in Chapter 3 was originally written by the author, and was later corrected by Ahmed Touati. In Chapter 4, Theorem 3 and Corollary 1 were proved by the author. Lemma 5 was proved by Ahmed Touati. Some parts of the analysis were proved by Ahmed Touati and were not included in the thesis. All experiments are conducted by the author.

CHAPTER 2
Background

In this chapter, we first introduce necessary RL preliminaries so that we can introduce our objective function, a convex-concave saddle-point version of MSPBE. Then, we introduce existing convex optimization algorithms. Finally, we show how existing convex optimization algorithms can be applied to policy evaluation with linear function approximation.

2.1 Markov Decision Process

In RL, a Markov Decision Process (MDP) is typically used to model the interaction between an agent and the environment. A MDP is defined by a tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$.

- $\mathcal{S}$ is the set of possible states.
- $\mathcal{A}$ is the set of actions
- P is the transition probability function where $P : \mathcal{S} \times \mathcal{A} \to (\mathcal{S} \to [0, 1])$. It maps a pair of state and action to distributions over next states.
- r is the reward function where $r : (\mathcal{S}, \mathcal{A}) \to \mathbb{R}$. It returns a reward that an agent will receive after performing an action $a \in \mathcal{A}$ at state $s \in \mathcal{S}$.
- γ is the discount factor used to discount rewards received farther in the future.

2.2 Policy Evaluation

An agent executes a policy when it is deployed into an environment. We define a policy π as a mapping from states to distributions over actions, i.e. $\pi : \mathcal{S} \to (\mathcal{A} \to [0, 1])$. For example, The probability of applying a at s is $\pi(a|s)$.

The value function for policy π, denoted as $V^\pi : \mathcal{S} \to \mathbb{R}$, represents the expected sum of discounted rewards along trajectories by executing π in the MDP. The value at state s is defined as:

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, \pi\right] \tag{2.1}$$

where the initial state s_0 is set to s and the reward that the agent receives at time t is r_t. Apply definitions of π and P on (2.1). We have:

$$V^{\pi}(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s,a)[r + \gamma V^{\pi}(s')] \tag{2.2}$$

A complete derivation can be found in [34]. Let V be a vector consisting of values at every state. From (2.2), we can express V as:

$$V = T^{\pi}V = r + \gamma P^{\pi} V \tag{2.3}$$

(2.3) is the Bellman equation. In (2.3), T^{π} is called the Bellman operator. r is a vector consisting of expected rewards at every state, and P^{π} is a matrix consisting of transition probabilities from s to s' for all $s, s' \in S$, i.e. $P^{\pi}(s,s') = \sum_{a \in A} \pi(a|s)P(s'|s,a)$.

V can be obtained by solving (2.3), but the transition probability function P is unknown in many situations. Policy evaluation is about estimating V. In this work, we use a linear function to approximate the value function at state s, so $V(s) = \phi(s)^{\top}\theta$. $\phi(s) \in \mathbb{R}^d$ is the feature representation of state s and θ is the parameter that we are trying to learn.

2.3 Mean Squared Projected Bellman Error

There are several objective functions that one can choose to optimize in order to find θ. In this work, we use the Mean Squared Projected Bellman Error (MSPBE). MSPBE is defined as:

$$\mathbf{MSPBE}(\theta) = \frac{1}{2}\left\|V - \Pi T^{\pi} V\right\|^2_{D^{\pi}} \tag{2.4}$$

We use $\Phi \in \mathbb{R}^{|S| \times d}$ to represent feature vectors of all states. In (2.4), $\Pi = \Phi(\Phi^{\top} D^{\pi} \Phi)^{-1} \Phi^{\top} D^{\pi}$ is a projection matrix which projects $T^{\pi}V$ on to the subspace spanned by $\{\phi(s) | s \in S\}$.

This projected error is weighted by $D^\pi \in \mathbb{R}^{|S|\times|S|}$, which is a diagonal matrix consisting of stationary distributions of all states. The stationary distribution of state s is defined as $d^\pi(s) = \sum_{s',a} P(s|s',a)\pi(a|s')d^\pi(s')$. A standard assumption in RL is the ergodicity of the MDP. We make this assumption so that the stationary distribution exists. From (2.4), we can derive an alternative form of MSPBE:

$$\mathbf{MSPBE}(\theta) = \frac{1}{2}\left\|\Phi^T D^\pi(\Phi\theta - \mathcal{T}^\pi(\Phi\theta))\right\|^2_{(\Phi^T D^\pi \Phi)^{-1}} \tag{2.5}$$

A complete derivation of (2.5) can be found in [36].

2.4 Convex Optimization Algorithms

In ML, the objective function typically has a finite sum structure:

$$\min_{x\in\mathbb{R}^d} f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x) \tag{2.6}$$

where each $f_i(x)$ is a convex and smooth function. Gradient Descent (GD) method, proposed by Cauchy [2], could be used to solve (2.6). Let x^t denote x at iteration t. GD updates x^t by $x^t - \alpha_t f'(x^t)$ where $\alpha_t \in \mathbb{R}$.

In situations where n is large, computing $f'(x^t) = \frac{1}{n}\sum_{i=1}^{n} f_i'(x^t)$ is an expensive operation. Robbins and Monro [29] propose Stochastic Gradient Descent method (SGD). SGD updates x^t by $x^t - \alpha_t f_{i_t}'(x^t)$, where i_t is uniformly chosen from $[n]$ at iteration t. Obviously, SGD is n times faster than GD in every iteration. More importantly, SGD is a convergent method. $f(x^t)$ converges to $f(x^*)$ if $\sum_t \alpha_t^2 < \infty$ and $\sum_t \alpha_t = \infty$, where $x^* \in \arg\min_x f(x)$.

The convergence of SGD is slow, but the computational cost per iteration is cheap. A series of fast stochastic methods were later proposed for solving (2.6) [30, 14, 19, 44, 23, 1]. They have linear convergence rates when (2.6) is strongly convex. Since this work focuses on Stochastic Variance Reduced Gradient method (SVRG) [19], we briefly introduce SVRG. SVRG consists of two loops. In each iteration of the outer loop, current value of x is stored and we denote it by $\tilde{x}$. The full gradient evaluated at $\tilde{x}$, $f'(\tilde{x})$, is also stored. In the t^{th}

iteration of the inner loop, $x^{t+1} = x^t - \alpha_t v_t$ where $v_t = f'_{i_t}(x_t) + f'(\tilde{x}) - f'_{i_t}(\tilde{x})$ and i_t is uniformly selected from $[n]$. SVRG is doing gradient descent updates in expectations, because $\mathbb{E}[v_t] = f'(x_t)$. The variance of SGD's updating directions is large, but the variance of SVRG's updating directions, v_t, reduces as both $\tilde{x}$ and x_t converge to x^* [19]. SVRG has a linear convergence rate when the objective function is strongly convex.

2.5 Saddle Point Formulation of Mean Squared Projected Bellman Error

We write MSPBE, presented in (2.5), in a more concise form:

$$\mathbf{MSPBE}(\theta) = \frac{1}{2} \|A\theta - b\|^2_{C^{-1}} \tag{2.7}$$

In (2.7), A, b and C are defined as:

$$\begin{aligned} A &= \mathbb{E}[\phi(s_t)(\phi(s_t) - \gamma\phi(s_{t+1}))^T] = \Phi^\top D^\pi (I - \gamma P^\pi)\Phi \\ b &= \mathbb{E}[\phi(s_t) r_t] = \Phi^\top D^\pi r \\ C &= \mathbb{E}[\phi(s_t)\phi(s_t)^T] = \Phi^\top D^\pi \Phi \end{aligned} \tag{2.8}$$

In (2.8), expectations are taken with respect to the stationary distribution, and (s_t, a_t, r_t, s_{t+1}) is the transition data generated by π at time t.

Let $\{(s_t, a_t, r_t, s_{t+1})\}_{t\in[n]}$ be the data set generated by π. We define empirical approximations of A, b and C as:

$$\hat{A} = \frac{1}{n}\sum_{t=1}^{n} \hat{A}_t, \quad \hat{b} = \frac{1}{n}\sum_{t=1}^{n} \hat{b}_t, \quad \hat{C} = \frac{1}{n}\sum_{t=1}^{n} \hat{C}_t.$$

where $\hat{A}_t, \hat{b}_t$ and $\hat{C}_t$ are defined as:

$$\hat{A}_t \triangleq \phi(s_t)(\phi(s_t) - \gamma\phi(s_{t+1}))^\top,$$
$$\hat{b}_t \triangleq r_t\phi(s_t)$$
$$\hat{C}_t \triangleq \phi(s_t)\phi(s_t)^\top$$

Replacing A, b, C in (2.7) with $\hat{A}, \hat{b}, \hat{C}$, we have the empirical MSPBE:

$$\mathbf{EM-MSPBE}(\theta) = \frac{1}{2}\left\|\hat{A}\theta - \hat{b}\right\|^2_{\hat{C}^{-1}} \tag{2.9}$$

(2.9) does not have a finite sum structure, so we cannot apply SGD and SVRG, introduced in Section 2.4, to solve (2.9). Du et al [15] show that minimizing (2.9) is equivalent as solving a convex-concave saddle-point problem:

$$\min_\theta \max_\omega f(\theta, \omega) = \langle \hat{b} - \hat{A}\theta, \omega\rangle - \frac{1}{2}||\omega||^2_{\hat{C}} \tag{2.10}$$

The idea of deriving (2.10) is by taking the conjugate of (2.9) and using the fact that the conjugate function of $g(x) = \frac{1}{2}\|x\|^2_{\hat{C}^{-1}}$ is $g^*(y) = \frac{1}{2}\|y\|^2_{\hat{C}}$. A complete derivation can be found in [15].

(2.10) has a finite sum structure, because we can write $f(\theta, \omega)$ as $f(\theta, \omega) = \frac{1}{n}\sum_{t=1}^n f_t(\theta, \omega) = \frac{1}{n}\sum_{t=1}^n \langle \hat{A}_t\theta - \hat{b}_t, \omega\rangle - \frac{1}{2}||\omega||^2_{\hat{C}_t}$. However, stochastic algorithms introduced in Section 2.4 cannot provide convergence guarantees when optimizing (2.10), because the objective is a saddle-point problem and there is no strong convexity in θ. Balamurugan and Bach [3] applied SVRG and SAGA to solve the general convex-concave problems, in addition to strong convexity, their method requires proximal mappings of variables, which are difficult to compute because it involves inverting $\hat{C}$. Du et al [15] successfully applied SVRG and SAGA to solve (2.10), with linear convergence rates. Du et al overcame the lack of strong convexity by exploiting spectral properties of the matrix, consisting of gradients taken of (2.10) with respect to θ and ω. More details on this will be given in Section 3.2.

Algorithm 1 SVRG for Policy Evaluation

```
Input: initial point $(\theta, \omega)$, $\sigma_\theta, \sigma_\omega$, $M$ and $K$
Output: $(\theta, \omega)$
 1: for m = 0 to M-1 do
 2:     Set $(\tilde{\theta}, \tilde{\omega})$ and $(\theta_{m,0}, \omega_{m,0})$ to $(\theta, \omega)$.
 3:     Set $\mu = F(\tilde{\theta}, \tilde{\omega})$
 4:     for j = 0 to K-1 do
 5:         Sample $t_j$ uniformly randomly from $[n]$
 6:         $v_{m,j} = F_{t_j}(\theta_{m,j}, \omega_{m,j}) - F_{t_j}(\tilde{\theta}, \tilde{\omega}) + \mu$
 7:         $$\begin{pmatrix} \theta_{m,j+1} \\ \omega_{m,j+1} \end{pmatrix} \leftarrow \begin{pmatrix} \theta_{m,j} \\ \omega_{m,j} \end{pmatrix} - \begin{pmatrix} \sigma_\theta & 0 \\ 0 & \sigma_\omega \end{pmatrix} v_{m,j}$$
 8:     end for
 9:     $(\theta, \omega) = (\theta_{m,K}, \omega_{m,K})$
10: end for
11: return $(\theta, \omega)$
```

2.6 SVRG for Policy Evaluation

We define the vector $F(\theta, \omega)$ obtained by stacking the primal and dual gradients.

$$F(\theta, \omega) \triangleq \begin{pmatrix} \nabla_\theta f(\theta, \omega) \\ \\ -\nabla_\omega f(\theta, \omega) \end{pmatrix} = \begin{pmatrix} -\hat{A}^\top \omega \\ \\ \hat{A}\theta - \hat{b} - \hat{C}\omega \end{pmatrix} \tag{2.11}$$

The primal and dual gradients are obtained by taking the gradient of (2.10) with respect to θ and ω. We have $F(\theta, \omega) = \frac{1}{n}\sum_{t=1}^{n} F_t(\theta, \omega)$ where $\forall t \in [n]$:

$$F_t(\theta, \omega) \triangleq \begin{pmatrix} -\hat{A}_t^\top \omega \\ \\ \hat{A}_t\theta - \hat{b}_t - \hat{C}_t\omega \end{pmatrix} \tag{2.12}$$

SVRG for policy evaluation [15] is given in Algorithm 1. In each iteration of the outer loop, current iterates (θ, ω) are saved as $(\tilde{\theta}, \tilde{\omega})$. Primal and dual graidents are computed and saved as $F(\tilde{\theta}, \tilde{\omega})$. A SVRG update is performed in each iteration of the inner loop.

CHAPTER 3
Batching SVRG for Policy Evaluation

SVRG is a fast and efficient method for solving problems with a finite sum structure. It has low memory costs, which makes it a practical choice when n is large. However, computations of the full gradient in every iteration of the outer loop could be expensive. Harikandeh et al [17] show that the number of gradient evaluations of SVRG can be reduced, and the idea is to approximate the full gradient and use it in inner loop updates. Harikandeh et al [17] propose Batching SVRG, a variant of SVRG where a batch of stochastic gradients is computed and averaged in an effort to approximate the full gradient. The error of approximation decreases as Batching SVRG runs. Batching SVRG converges linearly if the error of approximation decreases at an appropriate rate.

Batching SVRG is designed for solving strongly convex problems. In this chapter, we propose Batching SVRG for policy evaluation, where the objective function is convex-concave without strong convexity in the primal variable. We analyze the proposed method's convergence and show empirical results on some standard benchmarks.

3.1 Algorithm

Algorithm 2 presents batching SVRG for policy evaluation. It applies batching SVRG [17] on solving the saddle-point formulation of MSPBE. We propose Algorithm 2, similar to [17], estimating the full gradient in each epoch m using only a subset $\mathcal{B}_m$ (a mini-batch) of size $|\mathcal{B}_m| = B_m$ of training examples:

$$\mu_m = \frac{1}{B_m} \sum_{t \in \mathcal{B}_m} F_t(\tilde{\theta}, \tilde{\omega}) \tag{3.1}$$

where $F_t(\theta, \omega)$ is computed using (2.11).

Algorithm 2 Batching SVRG for Policy Evaluation

Input: initial point (θ, ω), $\sigma_\theta, \sigma_\omega$, M and K

Output: (θ, ω)

```
1:  for m = 0 to M-1 do
2:      Set $(\tilde{\theta}, \tilde{\omega})$ and $(\theta_{m,0}, \omega_{m,0})$ to $(\theta, \omega)$.
3:      Choose a mini-batch size $B_m$
4:      Sample a set $\mathcal{B}_m$ with $B_m$ elements uniformly from $[n]$
5:      Compute $\mu_m = \frac{1}{|B_m|} \sum_{t \in \mathcal{B}_m} F_t(\tilde{\theta}, \tilde{\omega})$
6:      for j = 0 to K-1 do
7:          Sample $t_j$ uniformly randomly from $[n]$
8:          $v_{m,j} = F_{t_j}(\theta_{m,j}, \omega_{m,j}) - F_{t_j}(\tilde{\theta}, \tilde{\omega}) + \mu_m$
9:
            $$\begin{pmatrix} \theta_{m,j+1} \\ \omega_{m,j+1} \end{pmatrix} \leftarrow \begin{pmatrix} \theta_{m,j} \\ \omega_{m,j} \end{pmatrix} - \begin{pmatrix} \sigma_\theta & 0 \\ 0 & \sigma_\omega \end{pmatrix} v_{m,j}$$
10:     end for
11:     $(\theta, \omega) = (\theta_{m,K}, \omega_{m,K})$
12: end for
13: return $(\theta, \omega)$
```

In each iteration j of the inner loop in Algorithm 2, it uses $v_{m,j}$ to update θ and ω. $v_{m,j}$ is the usual SVRG's updating direction, except that the full gradients is replaced with the mini-batch gradients μ_m.

$$v_{m,j} = F_{t_j}(\theta_{m,j}, \omega_{m,j}) - F_{t_j}(\tilde{\theta}, \tilde{\omega}) + \mu_m \tag{3.2}$$

where t_j is sampled uniformly from $[n]$.

3.2 Convergence Analysis

In order to characterize the convergence rate of Algorithm 2, we need to introduce some new notations and state new assumptions.

We denote by $\|A\| \triangleq \sup_{\|x\|=1} \|Ax\|$ the spectral norm of the matrix A and by $\kappa(A) = \|A\|\|A^{-1}\|$ its condition number. If the eigenvalues of a matrix A are real, we use $\lambda_{\max}(A)$ and $\lambda_{\min}(A)$ to denote respectively the largest and the smallest eigenvalue.

If we set $\sigma_\omega = \beta\sigma_\theta$ for a positive constant β, it is possible to write the inner loop update (line 9 in Algorithm 2) as an update for the vector $z_{m,j} \triangleq \begin{pmatrix} \theta_{m,j} \\ \frac{1}{\sqrt{\beta}}\omega_{m,j} \end{pmatrix}$ as follows :

$$z_{m,j+1} = z_{m,j} - \sigma_\theta \left(G_{t_j} z_{m,j} + (G_m - G_{t_j}) z_{m,0} - g_m \right)$$

where:

$$G_t \triangleq \begin{pmatrix} 0 & -\sqrt{\beta}\hat{A}_t^\top \\ \sqrt{\beta}\hat{A}_t & \beta\hat{C}_k \end{pmatrix}, \quad g_t \triangleq \begin{pmatrix} 0 \\ \sqrt{\beta}\hat{b}_t \end{pmatrix}$$

and their corresponding averages over the mini-batch B_m:

$$G_m \triangleq \frac{1}{B_m} \sum_{t \in \mathcal{B}_m} G_t, \quad g_m \triangleq \frac{1}{B_m} \sum_{t \in \mathcal{B}_m} g_t$$

Let's now define the matrix G (the vector g) as the average of matrices G_t (vectors g_t) over the entire dataset:

$$G \triangleq \begin{pmatrix} 0 & -\sqrt{\beta}\hat{A}^\top \\ \sqrt{\beta}\hat{A} & \beta\hat{C} \end{pmatrix} \quad \text{and} \quad g \triangleq \begin{pmatrix} 0 \\ \sqrt{\beta}\hat{b} \end{pmatrix}$$

To simplify notations, we overload the notation $\lambda_{\min} = \lambda_{\min}(G)$. Another important quantity that characterizes *smoothness* of our problem is L_G^2 defined below as:

$$L_G^2 = \left\| \frac{1}{n} \sum_{t=1}^{n} G_t^\top G_t \right\| \tag{3.3}$$

The matrix G will play a key role in the convergence analysis. Du et al [15] have already studied the spectral properties of G as it was critical for the convergence of SVRG for policy evaluation. The following lemma, restated from [15], shows the condition β should satisfy so that G is diagonalizable with all its eigenvalues real and positive.

Assumption 1. *$\hat{A}$ is nonsingular and $\hat{C}$ is positive definite. This implies that the saddle-point problem admits a unique solution $(\theta^\star, \omega^\star) = (\hat{A}^{-1}\hat{b}, 0)$ and we define $z^\star \triangleq (\theta^\star, \frac{1}{\sqrt{\beta}}\omega^\star)$.*

Lemma 1. *[15] Suppose assumption 1 holds and we choose $\beta = \frac{\sigma_\omega}{\sigma_\theta} = \frac{8\lambda_{\max}(\hat{A}^T\hat{C}^{-1}\hat{A})}{\lambda_{\min}(\hat{C})}$, then the matrix G is diagonalizable with all its eigenvalues real and positive.*

If assumptions of Lemma 1 hold, we can write G as $G = Q\Lambda Q^{-1}$ where Γ is a diagonal matrix whose diagonal entries are the eigenvalues of G, and Q consists of its eigenvectors. We define the residual vector $\Delta_m = z_m - z^\star$. To study the behaviour of our algorithms, we use the potential function $\|Q^{-1}\Delta_m\|^2$. As $\|Q\|^2 \|Q^{-1}\Delta_m\|^2 \geq \|\Delta_m\|^2 \geq \|\theta - \theta^\star\|^2$, the convergence of $\|Q^{-1}\Delta_m\|^2$ implies the convergence of $\|\theta - \theta^\star\|^2$.

In order to study the behavior of Algorithm 2, we define e_m, the error occurred at epoch m. This error comes from computing the gradients over a mini-batch $\mathcal{B}_m$.

$$e_m = (Gz_m - g) - (G_m z_m - g_m) \tag{3.4}$$

The stochastic update of the inner loop in Algorithm 2 could thus be written as follows:

$$z_{m,j+1} = z_{m,j} - \sigma_\theta \left(G_{t_j} z_{m,j} + (G - G_{t_j}) z_m - g + e_m \right) \tag{3.5}$$

Theorem 1. *Assume assumption 1 holds and we choose $\sigma_\theta = \frac{\lambda_{\min}}{6\kappa(Q)^2 L_G^2}$, $\beta = \frac{8\lambda_{\max}(\hat{A}^T\hat{C}^{-1}\hat{A})}{\lambda_{\min}(\hat{C})}$ and $K = \frac{2}{\sigma_\theta \lambda_{\min}} = \frac{12\kappa(Q)^2 L_G^2}{\lambda_{\min}^2}$, then we obtain:*

$$\mathbb{E}\left\|Q^{-1}\Delta_{m+1}\right\|^2 \leq \underbrace{\frac{2}{3}\mathbb{E}\left\|Q^{-1}\Delta_m\right\|^2}_{\textit{linear convergence term}} + \underbrace{\frac{2}{\lambda_{\min}^2}\mathbb{E}\left\|Q^{-1}e_m\right\|^2}_{\textit{additional error term}} \tag{3.6}$$

Proof. Define the residual vector Δ_m and $\Delta_{m,j}$ as:

$$\Delta_m = z_m - z^\star = \begin{pmatrix} \theta_m - \theta^* \\ \frac{1}{\sqrt{\beta}}(\omega_m - \omega^*) \end{pmatrix} \text{ and } \Delta_{m,j} = z_{m,j} - z^\star = \begin{pmatrix} \theta_{m,j} - \theta^* \\ \frac{1}{\sqrt{\beta}}(\omega_{m,j} - \omega^*) \end{pmatrix} \tag{3.7}$$

θ_m and ω_m are θ and ω at the beginning of epoch m. $\theta_{m,j}$ and $\omega_{m,j}$ are θ and ω at epoch m and j^{th} iteration of the inner loop . θ^* and ω^* are optimal solutions of (2.10). From the first order optimality condition given in (3.8), we know that

$$\begin{pmatrix} 0 & -\hat{A}^\top \\ \hat{A} & \hat{C} \end{pmatrix} \begin{pmatrix} \theta^* \\ \omega^* \end{pmatrix} = \begin{pmatrix} 0 \\ \hat{b} \end{pmatrix} \tag{3.8}$$

The above equality is obtained by setting (2.11) to a zero vector.

By writing out Algorithm 2's update, we have:

$$z_{m,j+1} = z_{m,j} - \sigma_\theta \left(G_{t_j} z_{m,j} + (G_m - G_{t_j}) z_m - g_m \right) \tag{3.9}$$

we defined e_m the error coming from using a mini-batch to compute the gradients at epoch m.

$$e_m = (G z_m - g) - (G_m z_m - g_m) \tag{3.10}$$

we can thus equivalently write (3.9) as:

$$z_{m,j+1} = z_{m,j} - \sigma_\theta \left(G_{t_j} z_{m,j} + (G - G_{t_j}) z_m - g - e_m \right) \tag{3.11}$$

Subtract both sides by $z^\star = (\theta^\star, \frac{1}{\sqrt{\beta}}\omega^\star)^\top$ and use the first order optimality condition given in (3.8). We obtain:

$$\begin{aligned} \Delta_{m,j+1} &= \Delta_{m,j} - \sigma_\theta (G\Delta_m + G_{t_j}\Delta_{m,j} - G_{t_j}\Delta_m) + \sigma_\theta e_m \\ &= (I - \sigma_\theta G)\Delta_{m,j} + \sigma_\theta (G - G_{t_j})(\Delta_{m,j} - \Delta_m) + \sigma_\theta e_m \end{aligned} \tag{3.12}$$

By our assumption, $\beta = \frac{8\lambda_{max}(\hat{A}^\top \hat{C}^{-1}\hat{A})}{\lambda_{min}(\hat{C})}$ so that G is diagonalizable by Lemma 1. Let $G = Q\Lambda Q^{-1}$ where Q contains eigenvectors and Λ contains eigenvalues of G. Multiply both sides of (3.12) by Q^{-1}, then take squared 2-norm and expectation. Set $\delta = \Delta_{m,j} - \Delta_m$. We get:

$$
\begin{aligned}
\mathbb{E}\left\|Q^{-1}\Delta_{m,j+1}\right\|^2 &= \mathbb{E}\|Q^{-1}(I-\sigma_\theta G)\Delta_{m,j} + \sigma_\theta Q^{-1}(G-G_{t_j})\delta + \sigma_\theta Q^{-1}e_m\|^2 \\
&= \mathbb{E}\left\|(I-\sigma_\theta\Lambda)Q^{-1}\Delta_{m,j}\right\|^2 + \sigma_\theta^2\,\mathbb{E}\left\|Q^{-1}(G-G_{t_j})\delta\right\|^2 + \sigma_\theta^2\,\mathbb{E}\left\|Q^{-1}e_m\right\|^2 \\
&+ 2\sigma_\theta\,\mathbb{E}\left\langle (I-\sigma_\theta\Lambda)Q^{-1}\Delta_{m,j}, Q^{-1}e_m\right\rangle \\
&\leq \|1-\sigma_\theta\Lambda\|^2\,\mathbb{E}\left\|Q^{-1}\Delta_{m,j}\right\|^2 + \sigma_\theta^2\,\mathbb{E}\left\|Q^{-1}(G-G_{t_j})\delta\right\|^2 + \sigma_\theta^2\,\mathbb{E}\left\|Q^{-1}e_m\right\|^2 \\
&+ 2\sigma_\theta\,\mathbb{E}\left\langle (I-\sigma_\theta\Lambda)Q^{-1}\Delta_{m,j}, Q^{-1}e_m\right\rangle \\
&\leq \|1-\sigma_\theta\Lambda\|^2\,\mathbb{E}\left\|Q^{-1}\Delta_{m,j}\right\|^2 + \sigma_\theta^2\,\mathbb{E}\left\|Q^{-1}G_{t_j}\delta\right\|^2 + \sigma_\theta^2\,\mathbb{E}\left\|Q^{-1}e_m\right\|^2 \\
&+ 2\sigma_\theta\,\mathbb{E}\left\langle (I-\sigma_\theta\Lambda)Q^{-1}\Delta_{m,j}, Q^{-1}e_m\right\rangle \qquad (3.13)
\end{aligned}
$$

The cross term in the second equality is simplified by using $\mathbb{E}[G_{t_j}] = G$ and G_{t_j} is independent with $\Delta_{m,j}$, Δ_m and e_m. We use in the last inequality that same independence and that the variance of a random variable is less than its second moment.

We borrow the following useful inequalities from appendix C of [15].

$$
\mathbb{E}\left\|Q^{-1}G_{t_j}\delta\right\|^2 \leq 2\kappa(Q)^2 L_G^2(\mathbb{E}\left\|Q^{-1}\Delta_{m,j}\right\|^2 + \mathbb{E}\left\|Q^{-1}\Delta_m\right\|^2) \qquad (3.14)
$$

$$
\|I-\sigma_\theta\Lambda\|^2 = \max\{|1-\sigma_\theta\lambda_{\min}(G)|^2, |1-\sigma_\theta\lambda_{\max}(G)|^2\} \leq 1 - 2\sigma_\theta\lambda_{\min} + \sigma_\theta^2\kappa(Q)^2L_G^2 \qquad (3.15)
$$

Now we bound the cross term in (3.13):

$$
\begin{aligned}
\mathbb{E}\left\langle (I-\sigma_\theta\Lambda)Q^{-1}\Delta_{m,j}, Q^{-1}e_m\right\rangle &\leq \sqrt{\mathbb{E}\left\|(I-\sigma_\theta\Lambda)Q^{-1}\Delta_{m,j}\right\|^2}\sqrt{\mathbb{E}\left\|Q^{-1}e_m\right\|^2} \\
&\leq \frac{\lambda_{\min}}{4\|I-\sigma_\theta\Lambda\|^2}\,\mathbb{E}\left\|(I-\sigma_\theta\Lambda)Q^{-1}\Delta_{m,j}\right\|^2 \\
&+ \frac{\|I-\sigma_\theta\Lambda\|^2}{\lambda_{\min}}\,\mathbb{E}\left\|Q^{-1}e_m\right\|^2 \qquad (3.16)
\end{aligned}
$$

the first inequality is obtained by Cauchy-Schwartz inequality The last inequaliy follows from the fact that $2zw \leq a^{-1}z^2 + aw^2$ for any $a > 0$ and we select $a = \frac{2\|I-\sigma_\theta\Lambda\|^2}{\lambda_{\min}}$ in order for the inequality to hold.

Put (3.14), (3.15) and (3.16) back to (3.13). We obtain:

$$\begin{aligned}\mathbb{E}\left\|Q^{-1}\Delta_{m,j+1}\right\|^2 \leq{}& (1 - 2\sigma_\theta\lambda_{\min} + 3\sigma_\theta^2 k(Q)^2 L_G^2 + \frac{1}{2}\sigma_\theta\lambda_{\min})\,\mathbb{E}\left\|Q^{-1}\Delta_{m,j}\right\|^2 \\ &+ 2\sigma_\theta^2\kappa^2(Q)L_G^2\,\mathbb{E}\left\|Q^{-1}\Delta_m\right\|^2 \\ &+ 2\sigma_\theta\frac{1 - 2\sigma_\theta\lambda_{\min} + \sigma_\theta^2 k(Q)^2 L_G^2 + (1/2)\sigma_\theta\lambda_{\min}}{\lambda_{\min}}\,\mathbb{E}\left\|Q^{-1}e_m\right\|^2 \end{aligned} \tag{3.17}$$

If we choose $0 \leq \sigma_\theta \leq \frac{\lambda_{\min}}{6\kappa(Q)^2 L_G^2}$, then $3\sigma_\theta^2\kappa(Q)^2 L_G^2$ and $\sigma_\theta^2\kappa(Q)^2 L_G^2$ are smaller than $(1/2)\sigma_\theta\lambda_{\min}$ which implies that:

$$\begin{aligned}\mathbb{E}\left\|Q^{-1}\Delta_{m,j+1}\right\|^2 \leq{}& (1 - \sigma_\theta\lambda_{\min})\,\mathbb{E}\left\|Q^{-1}\Delta_{m,j}\right\|^2 \\ &+ 2\sigma_\theta^2\kappa^2(Q)L_G^2\,\mathbb{E}\left\|Q^{-1}\Delta_m\right\|^2 \\ &+ 2\sigma_\theta\frac{1 - \sigma_\theta\lambda_{\min}}{\lambda_{\min}}\,\mathbb{E}\left\|Q^{-1}e_m\right\|^2 \end{aligned} \tag{3.18}$$

Note that $\sigma_\theta\lambda_{min} \leq \frac{\lambda_{min}^2}{6\kappa(Q)^2 L_G^2} \leq 1$, because of the following inequalities cited from Appendix C in [15]:

$$\lambda_{min}^2 \leq \lambda_{max}^2 \leq \|G\|^2 = \|\mathbb{E}\,G_t\|^2 \leq \left\|\mathbb{E}\,G_t^T G_t\right\|^2 = L_G^2 \leq \kappa(Q)^2 L_G^2 \tag{3.19}$$

Now enrolling the above inequality (3.18) from $j = 1$ to $K - 1$, we obtain:

$$\begin{aligned}\mathbb{E}\left\|Q^{-1}\Delta_{m+1}\right\|^2 \leq (1-\sigma_\theta\lambda_{\min})^K\,\mathbb{E}\left\|Q^{-1}\Delta_m\right\|^2 \\ + 2\sigma_\theta^2\kappa^2(Q)L_G^2\sum_{j=0}^{K-1}(1-\sigma_\theta\lambda_{\min})^j\,\mathbb{E}\left\|Q^{-1}\Delta_m\right\|^2 \\ + 2\sigma_\theta\frac{1-\sigma_\theta\lambda_{\min}}{\lambda_{\min}}\sum_{j=0}^{K-1}(1-\sigma_\theta\lambda_{\min})^j\,\mathbb{E}\left\|Q^{-1}e_m\right\|^2 \end{aligned} \tag{3.20}$$

As

$$\sum_{j=0}^{K-1}(1-\sigma_\theta\lambda_{\min})^j = \frac{1-(1-\sigma_\theta\lambda_{\min})^K}{1-(1-\sigma_\theta\lambda_{\min})} \leq \frac{1}{\sigma_\theta\lambda_{\min}} \tag{3.21}$$

, we obtain:

$$\begin{aligned}\mathbb{E}\left\|Q^{-1}\Delta_{m+1}\right\|^2 \leq \left((1-\sigma_\theta\lambda_{\min})^K + \frac{2\sigma_\theta^2\kappa^2(Q)L_G^2}{\sigma_\theta\lambda_{\min}}\right)\mathbb{E}\left\|Q^{-1}\Delta_m\right\|^2 \\ + 2\frac{1-\sigma_\theta\lambda_{\min}}{\lambda_{\min}^2}\,\mathbb{E}\left\|Q^{-1}e_m\right\|^2\end{aligned} \tag{3.22}$$

We choose:

$$\sigma_\theta = \frac{\lambda_{\min}}{6\kappa(Q)^2L_G^2} \quad \text{and} \quad K = \frac{2}{\sigma_\theta\lambda_{\min}} = \frac{12\kappa(Q)^2L_G^2}{\lambda_{\min}^2} \tag{3.23}$$

then we get:

$$\mathbb{E}\left\|Q^{-1}\Delta_{m+1}\right\|^2 \leq (\exp(-2)+1/3)\,\mathbb{E}\left\|Q^{-1}\Delta_m\right\|^2 + 2\frac{1-\sigma_\theta\lambda_{\min}}{\lambda_{\min}^2}\,\mathbb{E}\left\|Q^{-1}e_m\right\|^2 \tag{3.24}$$

$$\leq \frac{2}{3}\,\mathbb{E}\left\|Q^{-1}\Delta_m\right\|^2 + \frac{2}{\lambda_{\min}^2}\,\mathbb{E}\left\|Q^{-1}e_m\right\|^2 \tag{3.25}$$

□

Note that if $B_m = n \quad \forall m$, the error is zero and we recover the convergence rate of SVRG in Theorem 1. If we use an approximation of the full gradient at epoch m, then $\mathbb{E}\left\|Q^{-1}e_m\right\|^2 > 0$. If the error term is not 0, we could still maintain the linear convergence rate if the error term decreases at a faster rate than the linear convergence term. Harikandeh et al [17] discuss some possible approaches of batching to control the error term.

We show that an exponentially increasing set of batch sizes would result in linear convergence of batching SVRG. Assume that the sample variance of the norms of the vectors F_t is bounded by a constant Ξ^2:

$$\frac{1}{n-1}\sum_{t=1}^{n}\left\|G_t z - g_t\right\|^2 - \left\|Gz - g\right\|^2 \leq \Xi^2 \quad \forall z \tag{3.26}$$

The following lemma gives a bound on the error term given (3.26) holds.

Lemma 2. *Suppose (3.26) holds.* $\mathbb{E}\left\|e_m\right\|^2$ *satisfies:*

$$\mathbb{E}\left\|e_m\right\|^2 \leq \frac{n - B_m}{nB_m}\Xi^2 \tag{3.27}$$

Proof. By the definition of e_m given in (3.4), we have

$$\begin{aligned}\mathbb{E}\left\|e_m\right\|^2 &= \mathbb{E}\left\|Gz_m - g - (G_m z_m - g_m)\right\|^2 \\ &= \mathbb{E}\left\|Gz_m - g - \left(\frac{1}{B_m}\sum_{t\in\mathcal{B}_m} G_t z_m - g_t\right)\right\|^2 \end{aligned} \tag{3.28}$$

Since $\sum_{t=1}^{n} Gz_m - g - (G_t z_m - g_t) = n(Gz_m - g) - n(Gz_m - g) = 0$, we can apply Lemma 3 to (3.28) and have:

$$
\begin{aligned}
\mathbb{E}\left\|e_m\right\|^2 &= \mathbb{E}\left\|Gz_m - g - \left(\frac{1}{B_m}\sum_{t\in\mathcal{B}_m} G_t z_m - g_t\right)\right\|^2 \\
&= \frac{n-B_m}{nB_m}\frac{1}{n-1}\sum_{t=1}^{n}\left\|Gz_m - g - (G_t z_m - g_t)\right\|^2 \\
&= \frac{n-B_m}{nB_m}\frac{1}{n-1}\left[\underbrace{\sum_{t=1}^{n}\left\|Gz_m - g\right\|^2}_{n\|Gz_m-g\|^2} - 2\underbrace{\sum_{t=1}^{n}\langle Gz_m - g\,, G_t z_m - g_t\rangle}_{n\|Gz_m-g\|^2} + \sum_{t=1}^{n}\left\|G_t z_m - g_t\right\|^2\right] \\
&= \frac{n-B_m}{nB_m}\frac{1}{n-1}\left[\sum_{t=1}^{n}\left\|G_t z_m - g_t\right\|^2 - \left\|Gz_m - g\right\|^2\right] \\
&\underset{\text{by (3.26)}}{\leq} \frac{n-B_m}{nB_m}\Xi^2
\end{aligned}
\tag{3.29}
$$

□

The above proof relies on a lemma that is cited from [22]:

Lemma 3. *Lemma B.1 in [22] restated: Let* $z_1 \ldots z_M \in \mathbb{R}^d$ *be an arbitrary population of* M *vectors with* $\sum_{j=1}^{M} z_j = 0$*. Let* $\mathcal{I}$ *be a uniform random subsets of* $[M]$ *with size* m*. Then,*

$$
\mathbb{E}\left\|\frac{1}{m}\sum_{i\in\mathcal{I}} z_i\right\|^2 = \frac{M-m}{(M-1)m} \times \frac{1}{M}\sum_{j=1}^{M}\left\|z_j\right\|^2 \leq \frac{I(m<M)}{m}\frac{1}{M}\sum_{j=1}^{M}\left\|z_j\right\|^2
$$

where $I(m<M) = 1$ *if* $m < M$ *and zero otherwise*

By using the bound on e_m (Lemma 2) and setting an exponentially increasing sequence of batch sizes, we can show that batching SVRG has a linear convergence rate.

Theorem 2. *For* $m = 0 \ldots M$*, set* $B_m = \frac{n\Xi^2}{\Xi^2 + n\alpha\rho^m}$*, where* $\rho \leq 2/3$ *and* $\alpha > 0$*. Epoch* M *of batching SVRG satisfies:*

$$
\mathbb{E}\left\|Q^{-1}\Delta_M\right\|^2 \leq \left(\frac{2}{3}\right)^M \mathbb{E}\left\|Q^{-1}\Delta_0\right\|^2 + \frac{3\alpha}{\lambda_{min}^2(1-3\rho/2)}\left\|Q^{-1}\right\|^2\left(\frac{2}{3}\right)^M
\tag{3.30}
$$

Proof. From the result in Lemma 2, we have: $B_m \leq \frac{n\Xi^2}{n\mathbb{E}\|e_m\|^2+\Xi^2}$. Given our setting of B_m, we have:

$$\mathbb{E}\left\|e_m\right\|^2 \leq \alpha\rho^m \tag{3.31}$$

Combine the above inequality with the result in Theorem 1. We have:

$$\mathbb{E}\left\|Q^{-1}\Delta_{m+1}\right\|^2 \leq \frac{2}{3}\mathbb{E}\left\|Q^{-1}\Delta_m\right\|^2 + \frac{2\left\|Q^{-1}\right\|^2}{\lambda_{min}^2}\alpha\rho^m \tag{3.32}$$

Enrol the above inequality M times. We have:

$$\begin{aligned}
\mathbb{E}\left\|Q^{-1}\Delta_{m+1}\right\|^2 &\leq \left(\frac{2}{3}\right)^M \mathbb{E}\left\|Q^{-1}\Delta_0\right\|^2 + \frac{2\alpha\left\|Q^{-1}\right\|^2}{\lambda_{min}^2}\sum_{k=0}^{M-1}\frac{2}{3}^{M-1-k}\rho^k \\
&= \left(\frac{2}{3}\right)^M \mathbb{E}\left\|Q^{-1}\Delta_0\right\|^2 + \frac{2\alpha\left\|Q^{-1}\right\|^2}{\lambda_{min}^2}\left(\frac{2}{3}\right)^{M-1}\sum_{k=0}^{M-1}\left(\frac{3\rho}{2}\right)^k \\
&\leq \left(\frac{2}{3}\right)^M \mathbb{E}\left\|Q^{-1}\Delta_0\right\|^2 + \frac{2\alpha\left\|Q^{-1}\right\|^2}{\lambda_{min}^2}\left(\frac{2}{3}\right)^{M-1}\frac{1}{1-3\rho/2} \\
&= \left(\frac{2}{3}\right)^M \mathbb{E}\left\|Q^{-1}\Delta_0\right\|^2 + \frac{3\alpha}{\lambda_{min}^2(1-3\rho/2)}\left\|Q^{-1}\right\|^2\left(\frac{2}{3}\right)^M
\end{aligned} \tag{3.33}$$

The second inequality is derived by using the fact that $\frac{3\rho}{2} \leq 1$. $\frac{3\rho}{2} \leq 1$ because we set $\rho \leq \frac{2}{3}$. □

3.3 Experiments

3.3.1 Experiment Setups

We evaluate our algorithms on four benchmark problems:

1. **Random MDP** [13]. A randomly generated MDP with 400 states, 10 actions. Each state has a 201 dimensional feature vector where each entry except the last one is uniformly sampled from $[0, 1]$. The last entry of every state's feature vector is 1. For every state-action pair, we randomly select 5 states that are reachable from it. We add 10^{-5} to all transition probabilities. Rewards are uniformly sampled from $[-10, 10]$.

2. **Mountain Car** [25]. A car is initially positioned at a valley and its goal is to move to the mountain's top. A car has three possible actions: throttle forward, throttle reverse and zero throttle. States consist of positions and velocities of the car. The terminal state is the top of the mountain. The car gets a reward of -1 until it reaches the terminal state. Details of the state transition dynamics can be found in [34].

3. **Cart Pole** [4]. A pole is located on a cart and the cart is allowed to move left or right. States consist of cart's position, velocity, and pole's angle and velocity at tip. Reward is +1 if the pole does not fall. OpenAI Gym library [9] is used to simulate this environment and details of the dynamics can be found in CartPole-v1.

4. **Acrobot** [32]. The system consists of two joints and two links. Links are initially positioned downward and the goal is to swing them up. States are two rotational joints' angles and angular velocities. Reward is always -1 before reaching the terminal state. OpenAI Gym library [9] is used to simulate this environment and details of the dynamics can be found in Acrobot-v1.

Note that the transition probability function and reward function in our Random MDP domain are different from what is described in Dann et al [13]. We make this modification so that policy can be measured more effectively in the control case.

3.3.2 Experiment Details

1. **Data collection process.** In all four environments, we evaluate the data generated by a policy which performs random actions. We collect $n = 20000$ data samples in Random MDP and Cart Pole environments, and we collect $n = 5000$ data samples in Mountain car and Acrobot environments. For discount factor, we use $\gamma = 0.95$ in Random MDP environment. We use $\gamma = 0.99$ in Cart Pole and Mountain Car environments and we use $\gamma = 0.9$ in Acrobot environment.
2. **Feature engineering process.** We do not modify state's feature vector in Random MDP environment. In Mountain Car and Acrobot environments, we normalize state's features to $[0, 1]$ and apply state's features to 10 (in Mountain Car environment) and 3 (in Acrobot environment) evenly spaced RBF kernels, which is defined as $f(x) = \exp(-r/\sigma)$. We follow [18] and set $\sigma = 0.01$ in Mountain Car environment and we set $\sigma = 0.1$ in Acrobot environment. In Cart Pole environment, we follow [18] and bound velocity values by $f(x) = \tanh(x/10)$. We use raw observations for position values.
3. **Parameter selection** For SVRG, We set step sizes and number of SVRG's inner loop iteration based on grid searching results. We did not tune step sizes for Batching SVRG, and the number of Batching SVRG's inner loop iteration is set to n. Grid searching results can be found in Appendix A.
4. **Initialization** In Random MDP and Mountain Car environments, we initialize θ and ω to zero vectors. In Car Pole and Acrobot environments, we initialize each element of θ and ω by randomly sampling from $[0, 10]$. This is because zero vectors give a relatively small objective value in these two environments.
5. **Description of how experiments were run.** We generate data and set parameters by following procedures that we described above. Then we aggregate results of each method for 10 independent runs.

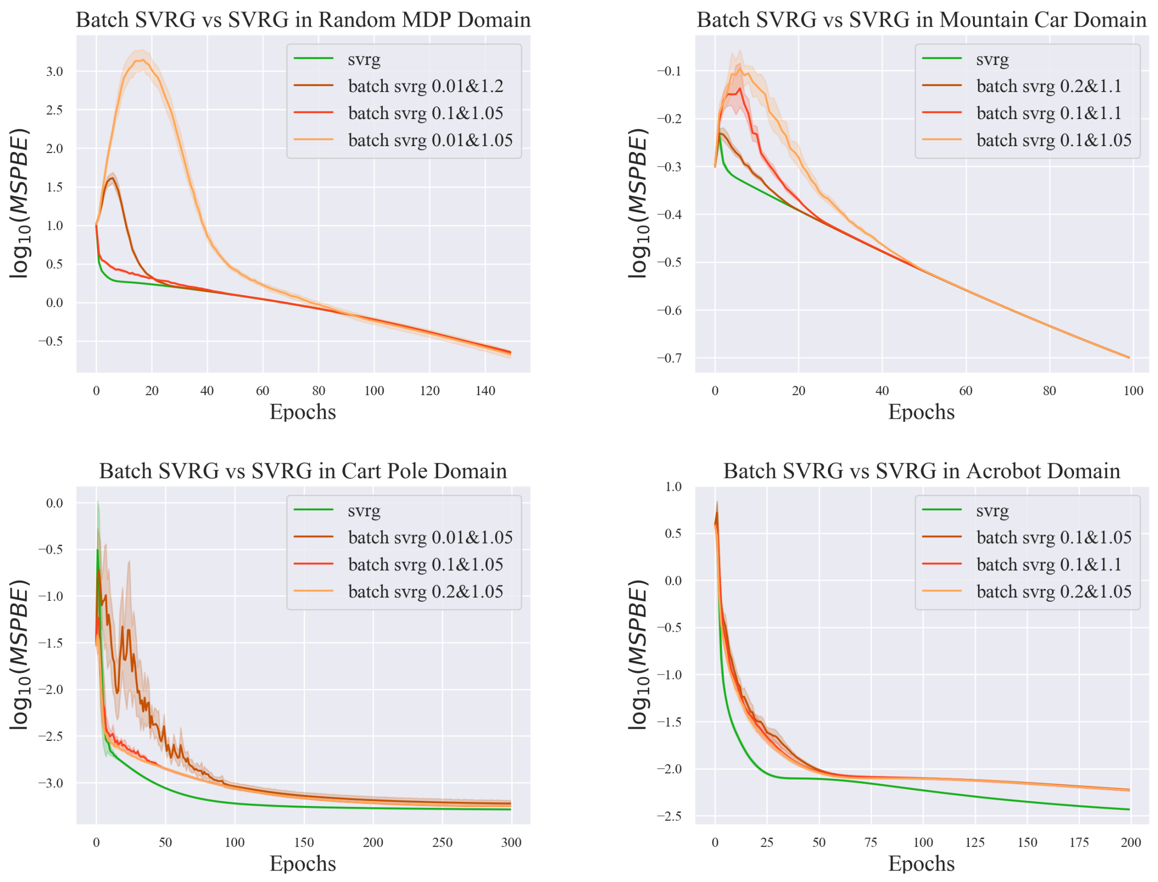

Figure 3–1: The above figures show performances of batching SVRG and SVRG in different environments. Batching SVRG is evaluated with different parameter settings. Two numbers following *batch svrg* are the initial batch size and the growth rate of batch size. For example, *batch svrg 0.01&1.2* means that batching SVRG is started with a batch size of $n * 0.01$ and increase the batch size by 20% in every epoch.

	Random MDP	Mountain Car	Cart Pole	Acrobot
SVRG	300	200	600	600
Batching SVRG	280	191	583	385
Batching SVRG	271	185	571	383
Batching SVRG	225	171	525	371

Table 3–1: Computational Costs of Batching SVRG and SVRG. This table shows the number of passes through the dataset for SVRG and three variants of Batching SVRG when generating results given in Figure 3–1

3.3.3 Results

Figure 3–1 shows performances of SVRG and variants of Batching SVRG in different environments. Their computational costs can be found in Table 3–1. In Random MDP and Mountain Car environments, Batching SVRG's objective values first increase. This is because we use a initial batch size that is too small, so batch gradients in early epochs are not good estimates of the full gradient. Objective values later start to decrease because Batching SVRG starts to use full gradients and it eventually reaches the same level of performances with SVRG. In Cart Pole and Acrobot environments, Batching SVRG's performance is not stable in early epochs, but objective values keep decreasing. This means that full gradients are estimated reasonably well by batch gradients in early epochs. As Batching SVRG starts to use full gradients in later epochs, variances of objective values are much smaller and objective values decrease as fast as SVRG.

It is clear that batching SVRG is able to achieve the same level of performances with SVRG while using a significantly less amount of data. Batching SVRG's performance is worse than SVRG in early epochs, but it later reaches the same level of objective values and has the same convergence speed with SVRG. This is expected because our theoretical result suggests that having an approximation error will not affect the overall convergence rate if the error decreases properly.

CHAPTER 4
SCSG for Policy Evaluation

Stochastically Controlled Stochastic Gradient (SCSG) [22], similar with Batching SVRG, uses a batch of stochastic gradients to approximate the full gradient in order to reduce computational costs. Unlike Batching SVRG, the convergence of SCSG does not rely on the decrease of approximation errors. Depending on the problem, the rate of SCSG's convergence could be independent of n, which makes SCSG a very practical choice for solving large scale problems. SCSG has a linear convergence rate when the objective function is strongly convex. In this chapter, we propose SCSG for policy evaluation, where the objective function is convex-concave without strong convexity in the primal variable. We show that the proposed method has a linear convergence rate, and we evaluate the proposed method on standard benchmarks.

4.1 Algorithm

Algorithm 3 presents SCSG for Policy Evaluation. Similar to Bachting SVRG for policy evaluation in Algorithm 2, Algorithm 3 implements the gradient computation on a subset $\mathcal{B}$ of training examples at each epoch, but the mini-batch size is fixed in advance. Moreover, instead of being fixed, the number of iteration for the inner loop in Algorithm 3 is sampled from a geometrically distributed random variable: $K_m \sim \text{Geom}(\frac{B}{B+1}) \quad \text{for each epoch} \quad m$. The number of iteration for the inner loop of Algorithm 3 is equal in expectation to $\frac{\frac{B}{B+1}}{1-\frac{B}{B+1}} = B$.

Algorithm 3 SCSG for Policy Evaluation

Input: initial point (θ, ω), $\sigma_\theta, \sigma_\omega$, M, K and mini-batch size B
Output: (θ, ω)

```
1: for m = 0 to M-1 do
2:     Set $(\tilde{\theta}, \tilde{\omega})$ and $(\theta_{m,0}, \omega_{m,0})$ to $(\theta, \omega)$.
3:     Sample a set $\mathcal{B}$ with $B$ elements uniformly from $[n]$
4:     Compute $\mu_m = \frac{1}{B}\sum_{t\in\mathcal{B}} F_t(\tilde{\theta}, \tilde{\omega})$
5:     $K_m \sim$ Geom$(\frac{B}{B+1})$
6:     for j = 0 to $K_m - 1$ do
7:         Sample $t_j$ uniformly randomly from $[n]$
8:         $v_{m,j} = F_{t_j}(\theta_{m,j}, \omega_{m,j}) - F_{t_j}(\tilde{\theta}, \tilde{\omega}) + \mu_m$
9:         $\begin{pmatrix}\theta_{m,j+1}\\ \omega_{m,j+1}\end{pmatrix} \leftarrow \begin{pmatrix}\theta_{m,j}\\ \omega_{m,j}\end{pmatrix} - \begin{pmatrix}\sigma_\theta & 0\\ 0 & \sigma_\omega\end{pmatrix} v_{m,j}$
10:    end for
11:    $(\theta, \omega) = (\theta_{m,K_m}, \omega_{m,K_m})$
12: end for
13: return $(\theta, \omega)$
```

4.2 Convergence Analysis

The convergence of Algorithm 3 relies on a property of geometric random variables. The following lemma is cited from Lei and Jordan [22] and will be used in our analysis.

Lemma 4. *Lemma A.2 in [22]. Let $N \sim Geom(\gamma)$ for some $\gamma > 0$. Then any sequence $\{D_n\}$ with $\mathbb{E}\,|D_N| < \infty$ satisfies*

$$\mathbb{E}[D_N - D_{N+1}] = \left(\frac{1}{\gamma} - 1\right)(D_0 - \mathbb{E}\, D_N) \tag{4.1}$$

Before stating the convergence result, we introduce *the complexity measure* defined as follows:

$$\mathcal{H} = \frac{1}{n}\sum_{t=1}^{n} \|G_t z^\star - g_t\|^2 \tag{4.2}$$

This quantity is equivalent to the complexity measure that is introduced by Lei and Jordan [22] to motivate and analyze SCSG for convex finite sum minimization problems.

Theorem 3. *Suppose assumption 1 holds. Set $\sigma_\theta \le \min\{\frac{\lambda_{min}}{20\kappa(Q)^2 L_G^2}, \frac{5}{28B\lambda_{max}}\}$, $\sigma_\omega = \beta\sigma_\theta$ and $B \ge \frac{70\kappa(Q)^2 L_G^2}{\lambda_{min}^2}$. The last iterate θ_{M-1} satisfies:*

$$\mathbb{E}\left\|\theta_{M-1}-\theta^\star\right\|^2 \leq \frac{(1+0.7\sigma_\theta B\lambda_{max})\kappa(Q)^2}{(1+0.8\sigma_\theta B\lambda_{min})^M}\mathbb{E}\left\|\Delta_0\right\|^2 + \frac{60\kappa(Q)^2\mathcal{H}I(B<n)}{B\lambda_{min}^2} \tag{4.3}$$

Proof. Algorithm 3's inner loop update is same with Algorithm 2. By (3.9), we have:

$$z_{m,j+1} = z_{m,j} - \sigma_\theta\left(G_{t_j}z_{m,j} + (G_m - G_{t_j})z_m - g_m\right) \tag{4.4}$$

Define e_m same as give in (3.10). We can write the above update equivalently as:

$$z_{m,j+1} = z_{m,j} - \sigma_\theta\left(G_{t_j}z_{m,j} + (G - G_{t_j})z_m - g - e_m\right) \tag{4.5}$$

Subtract both sides by $z^\star = (\theta^\star, \frac{1}{\sqrt{\beta}}\omega^\star)^\top$ and use the first order optimality condition given in (3.8). Define the residual vector Δ_m and $\Delta_{m,j}$ same as (3.7). We obtain:

$$\Delta_{m,j+1} = \Delta_{m,j} - \sigma_\theta(G\Delta_m + G_{t_j}\Delta_{m,j} - G_{t_j}\Delta_m) + \sigma_\theta e_m \tag{4.6}$$

By assumption 1 and our setting of β, conditions of Lemma 1 are satisfied, so G is diagonalizable with all its eigenvalues real and positive. Let $G = Q\Lambda Q^{-1}$ where Q contains eigenvectors and Λ contains eigenvalues of G. Multiply both sides of (4.6) by Q^{-1}, then take squared 2-norm and expectation. We get:

$$\begin{aligned}\mathbb{E}\left\|Q^{-1}\Delta_{m_j+1}\right\|^2 &= \mathbb{E}\left\|Q^{-1}\Delta_{m,j} - \sigma_\theta(\Lambda Q^{-1}\Delta_m + Q^{-1}G_{t_j}\Delta_{m,j} - Q^{-1}G_{t_j}\Delta_m - Q^{-1}e_m)\right\|^2 \\ &= \mathbb{E}\left\|Q^{-1}\Delta_{m,j}\right\|^2 - 2\sigma_\theta\,\mathbb{E}\left\langle \Lambda Q^{-1}\Delta_{m,j} - Q^{-1}e_m, Q^{-1}\Delta_{m,j}\right\rangle \\ &+ \sigma_\theta^2\underbrace{\mathbb{E}\left\|\Lambda Q^{-1}\Delta_m + Q^{-1}G_{t_j}\Delta_{m,j} - Q^{-1}G_{t_j}\Delta_m - Q^{-1}e_m\right\|^2}_{(1)}\end{aligned} \tag{4.7}$$

We first bound (1) in (4.7).

$$\begin{aligned}
&\mathbb{E}\left\|\Lambda Q^{-1}\Delta_m + Q^{-1}G_{t_j}\Delta_{m,j} - Q^{-1}G_{t_j}\Delta_m - Q^{-1}e_m\right\|^2 \\
&= \mathbb{E}\left\|Q^{-1}G_{t_j}\Delta_{m,j} - Q^{-1}G_{t_j}\Delta_m - \Lambda Q^{-1}\Delta_{m,j} + \Lambda Q^{-1}\Delta_m\right\|^2 + \left\|\Lambda Q^{-1}\Delta_{m,j} - Q^{-1}e_m\right\|^2 \\
&= \mathbb{E}\left\|Q^{-1}(G - G_{t_j})(\Delta_{m,j} - \Delta_m)\right\|^2 + \left\|\Lambda Q^{-1}\Delta_{m,j} - Q^{-1}e_m\right\|^2 \\
&\leq \mathbb{E}\left\|Q^{-1}G_{t_j}(\Delta_{m,j} - \Delta_m)\right\|^2 + 2\left\|\Lambda Q^{-1}\Delta_{m,j}\right\|^2 + 2\left\|Q^{-1}e_m\right\|^2 \\
&\leq 2\kappa(Q)^2L_G^2\mathbb{E}\left\|Q^{-1}\Delta_{m,j}\right\|^2 + 2\kappa(Q)^2L_G^2\mathbb{E}\left\|Q^{-1}\Delta_m\right\|^2 + 2\left\|\Lambda Q^{-1}\Delta_{m,j}\right\|^2 + 2\left\|Q^{-1}e_m\right\|^2
\end{aligned} \tag{4.8}$$

To derive the first equality and the first inequality, we use the fact that $\mathbb{E}\|X\|^2 = \mathbb{E}\|X - \mathbb{E}X\|^2 + \|\mathbb{E}X\|^2$ for any random variable X. We use (3.14) to derive the last inequality. Put (4.8) back to (4.7) and rearrange terms in (4.7). We have:

$$\begin{aligned}
2\sigma_\theta\,\mathbb{E}\left\langle\Lambda Q^{-1}\Delta_{m,j}, Q^{-1}\Delta_{m,j}\right\rangle \leq\ & \mathbb{E}\left\|Q^{-1}\Delta_{m,j}\right\|^2 - \mathbb{E}\left\|Q^{-1}\Delta_{m,j+1}\right\|^2 + 2\sigma_\theta\,\mathbb{E}\left\langle Q^{-1}e_m, Q^{-1}\Delta_{m,j}\right\rangle \\
&+ 2\sigma_\theta^2\kappa(Q)^2L_G^2\mathbb{E}\left\|Q^{-1}\Delta_{m,j}\right\|^2 + 2\sigma_\theta^2\kappa(Q)^2L_G^2\mathbb{E}\left\|Q^{-1}\Delta_m\right\|^2 \\
&+ 2\sigma_\theta^2\left\|\Lambda Q^{-1}\Delta_{m,j}\right\|^2 + 2\sigma_\theta^2\left\|Q^{-1}e_m\right\|^2
\end{aligned} \tag{4.9}$$

Set j to K_m in the above inequality and take an expectation with respect to $K_m \sim \mathrm{Geom}(\frac{B}{B+1})$, we obtain using Lemma 4:

$$\begin{aligned}
2\sigma_\theta\,\mathbb{E}\left\langle\Lambda Q^{-1}\Delta_{m,K_m}, Q^{-1}\Delta_{m,K_m}\right\rangle \leq\ & \frac{1}{B}\left(\mathbb{E}\left\|Q^{-1}\Delta_{m,0}\right\|^2 - \mathbb{E}\left\|Q^{-1}\Delta_{m,K_m}\right\|^2\right) \\
&+ 2\sigma_\theta\,\mathbb{E}\left\langle Q^{-1}e_m, Q^{-1}\Delta_{m,K_m}\right\rangle + 2\sigma_\theta^2\kappa(Q)^2L_G^2\mathbb{E}\left\|Q^{-1}\Delta_{m,K_m}\right\|^2 \\
&+ 2\sigma_\theta^2\kappa(Q)^2L_G^2\mathbb{E}\left\|Q^{-1}\Delta_m\right\|^2 + 2\sigma_\theta^2\,\mathbb{E}\left\|\Lambda Q^{-1}\Delta_{m,K_m}\right\|^2 + 2\sigma_\theta^2\left\|Q^{-1}e_m\right\|^2
\end{aligned} \tag{4.10}$$

Multiply both sides of the above inequality with B. We get:

$$
\begin{aligned}
2\sigma_\theta B\,\mathbb{E}\left\langle \Lambda Q^{-1}\Delta_{m+1}, Q^{-1}\Delta_{m+1}\right\rangle \le{}& \mathbb{E}\left\|Q^{-1}\Delta_m\right\|^2 - \mathbb{E}\left\|Q^{-1}\Delta_{m+1}\right\|^2 \\
&+ 2\sigma_\theta B\,\underbrace{\mathbb{E}\left\langle Q^{-1}e_m, Q^{-1}\Delta_{m+1}\right\rangle}_{(1)} + 2B\sigma_\theta^2\kappa(Q)^2L_G^2\mathbb{E}\left\|Q^{-1}\Delta_{m+1}\right\|^2 \\
&+ 2B\sigma_\theta^2\kappa(Q)^2L_G^2\mathbb{E}\left\|Q^{-1}\Delta_m\right\|^2 + 2B\sigma_\theta^2\,\underbrace{\mathbb{E}\left\|\Lambda Q^{-1}\Delta_{m+1}\right\|^2}_{(2)} \\
&+ 2B\sigma_\theta^2\left\|Q^{-1}e_m\right\|^2 \qquad (4.11)
\end{aligned}
$$

Note that $\Delta_{m,0} = \Delta_m$ and $\Delta_{m,K_m} = \Delta_{m+1}$ by definitions given in (3.7). We will derive an inequality and use it repeatedly in the rest of the proof:

$$
\begin{aligned}
\mathbb{E}\left\|Q^{-1}\Delta_{m+1}\right\|^2 &= \mathbb{E}\left\|\Lambda^{-\frac{1}{2}}\Lambda^{\frac{1}{2}}Q^{-1}\Delta_{m+1}\right\|^2 \\
&\le \left\|\Lambda^{-\frac{1}{2}}\right\|^2\mathbb{E}\left\|\Lambda^{\frac{1}{2}}Q^{-1}\Delta_{m+1}\right\|^2 \\
&\le \frac{1}{\lambda_{min}}\,\mathbb{E}\,\Delta_{m+1}^\top Q^{-\top}\Lambda Q^{-1}\Delta_{m+1} \qquad (4.12)
\end{aligned}
$$

$\left\|\Lambda^{-\frac{1}{2}}\right\|^2 \le \frac{1}{\lambda_{min}}$ because Λ is a diagonal matrix consisting of eigenvalues of G. We now give a bound of (1) in (4.11)

$$
\begin{aligned}
\mathbb{E}\left\langle Q^{-1}e_m, Q^{-1}\Delta_{m+1}\right\rangle &\le \sqrt{\mathbb{E}\left\|Q^{-1}e_m\right\|^2}\sqrt{\mathbb{E}\left\|Q^{-1}\Delta_{m+1}\right\|^2} \\
&\le \frac{\lambda_{min}}{10}\mathbb{E}\left\|Q^{-1}\Delta_{m+1}\right\|^2 + \frac{10}{\lambda_{min}}\mathbb{E}\left\|Q^{-1}e_m\right\|^2 \\
&\le 0.1\,\mathbb{E}\,\Delta_{m+1}^\top Q^{-\top}\Lambda Q^{-1}\Delta_m + \frac{10}{\lambda_{min}}\mathbb{E}\left\|Q^{-1}e_m\right\|^2 \qquad (4.13)
\end{aligned}
$$

The first inequality follows from Cauchy-Schwartz inequality. The second inequaliy follows from the fact that $2zw \le a^{-1}z^2 + aw^2$ for any $a > 0$. The third inequality is derived by using (4.12).

(2) in (4.11) can be bounded as:

$$
\begin{aligned}
\mathbb{E}\left\|\Lambda Q^{-1}\Delta_{m+1}\right\|^2 &= \mathbb{E}\,\Delta_{m+1}^\top Q^{-\top}\Lambda^2 Q^{-1}\Delta_{m+1} \\
&\le \lambda_{max}\,\mathbb{E}\,\Delta_{m+1}^\top Q^{-\top}\Lambda Q^{-1}\Delta_{m+1} \\
&= \lambda_{max}\,\mathbb{E}\left\langle \Lambda Q^{-1}\Delta_{m+1}, Q^{-1}\Delta_{m+1}\right\rangle
\end{aligned} \tag{4.14}
$$

Put (4.12), (4.13) and (4.14) back to (4.11), then rearrange terms. We have:

$$
\begin{aligned}
&\sigma_\theta B\left(1.8 - \frac{2\sigma_\theta\kappa(Q)^2L_G^2}{\lambda_{min}} - 2\sigma_\theta\lambda_{max}\right)\mathbb{E}\left\langle \Lambda Q^{-1}\Delta_{m+1}, Q^{-1}\Delta_{m+1}\right\rangle + \mathbb{E}\left\|Q^{-1}\Delta_{m+1}\right\|^2 \\
&\le \mathbb{E}\left\|Q^{-1}\Delta_m\right\|^2 + 2\sigma_\theta^2 B\kappa(Q)^2L_G^2\mathbb{E}\left\|Q^{-1}\Delta_m\right\|^2 + \sigma_\theta B\left(2\sigma_\theta + \frac{20}{\lambda_{min}}\right)\mathbb{E}\left\|Q^{-1}e_m\right\|^2
\end{aligned} \tag{4.15}
$$

We can lower bound the left hand side of the above inequality since $\frac{\lambda_{min}}{2}\mathbb{E}\left\|Q^{-1}\Delta_{m+1}\right\|^2 \le \frac{1}{2}\Delta_{m+1}^\top Q^{-\top}\Lambda Q^{-1}\Delta_{m+1}$. Thus, we have:

$$
\begin{aligned}
&\sigma_\theta B\left(1.8 - \frac{2\sigma_\theta\kappa(Q)^2L_G^2}{\lambda_{min}} - 2\sigma_\theta\lambda_{max}\right)\left[\frac{1}{2}\mathbb{E}\,\Delta_{m+1}^\top Q^{-\top}\Lambda Q^{-1}\Delta_{m+1} + \frac{\lambda_{min}}{2}\mathbb{E}\left\|Q^{-1}\Delta_{m+1}\right\|^2\right] \\
&+\mathbb{E}\left\|Q^{-1}\Delta_{m+1}\right\|^2 \le \mathbb{E}\left\|Q^{-1}\Delta_m\right\|^2 + 2\sigma_\theta^2 B\kappa(Q)^2L_G^2\mathbb{E}\left\|Q^{-1}\Delta_m\right\|^2 + \sigma_\theta B\left(2\sigma_\theta + \frac{20}{\lambda_{min}}\right)\mathbb{E}\left\|Q^{-1}e_m\right\|^2
\end{aligned} \tag{4.16}
$$

From (3.19), we know that $-\frac{\kappa(Q)^2L_G^2}{\lambda_{min}} \le -\lambda_{max}$. We can further lower bound the left hand side of the above inequality:

$$
\begin{aligned}
&\sigma_\theta B\left(0.9 - \frac{2\sigma_\theta\kappa(Q)^2L_G^2}{\lambda_{min}}\right)\left[\mathbb{E}\,\Delta_{m+1}^\top Q^{-\top}\Lambda Q^{-1}\Delta_{m+1} + \lambda_{min}\mathbb{E}\left\|Q^{-1}\Delta_{m+1}\right\|^2\right] + \mathbb{E}\left\|Q^{-1}\Delta_{m+1}\right\|^2 \\
&\le \mathbb{E}\left\|Q^{-1}\Delta_m\right\|^2 + 2\sigma_\theta^2 B\kappa(Q)^2L_G^2\mathbb{E}\left\|Q^{-1}\Delta_m\right\|^2 + \sigma_\theta B\left(2\sigma_\theta + \frac{20}{\lambda_{min}}\right)\mathbb{E}\left\|Q^{-1}e_m\right\|^2 \\
&\le \mathbb{E}\left\|Q^{-1}\Delta_m\right\|^2 + 2\sigma_\theta^2 B\kappa(Q)^2L_G^2\mathbb{E}\left\|Q^{-1}\Delta_m\right\|^2 \\
&+\sigma_\theta B\left(4\sigma_\theta + \frac{40}{\lambda_{min}}\right)\left[\frac{I(B<n)}{B}\kappa(Q)^2L_G^2\,\mathbb{E}\left\|Q^{-1}\Delta_m\right\|^2 + \frac{I(B<n)\|Q^{-1}\|^2}{B}\mathcal{H}\right]
\end{aligned} \tag{4.17}
$$

In the second inequality, we use the result in Lemma 5 to bound $\mathbb{E}\left\|Q^{-1}e_m\right\|^2$. We can use (4.12) to upper bound $\mathbb{E}\left\|Q^{-1}\Delta_m\right\|^2$ in the above inequality and we obtain:

$$\begin{aligned}
&\sigma_\theta B\left(0.9-\frac{2\sigma_\theta\kappa(Q)^2L_G^2}{\lambda_{min}}\right)\left[\mathbb{E}\,\Delta_{m+1}^\top Q^{-\top}\Lambda Q^{-1}\Delta_{m+1}+\lambda_{min}\mathbb{E}\left\|Q^{-1}\Delta_{m+1}\right\|^2\right]+\mathbb{E}\left\|Q^{-1}\Delta_{m+1}\right\|^2\\
&\leq\mathbb{E}\left\|Q^{-1}\Delta_m\right\|^2+\sigma_\theta B\left(\frac{2\sigma_\theta\kappa(Q)^2L_G^2}{\lambda_{min}}+\frac{I(B<n)\kappa(Q)^2L_G^2}{B\lambda_{min}}\left(4\sigma_\theta+\frac{40}{\lambda_{min}}\right)\right)\mathbb{E}\,\Delta_m^\top Q^{-\top}\Lambda Q^{-1}\Delta_m\\
&+\sigma_\theta\left(4\sigma_\theta+\frac{40}{\lambda_{min}}\right)I(B<n)\left\|Q^{-1}\right\|^2\mathcal{H} \qquad (4.18)
\end{aligned}$$

By our setting, $\sigma_\theta\leq\frac{\lambda_{min}}{20\kappa(Q)^2L_G^2}$, so $\frac{2\sigma_\theta\kappa(Q)^2L_G^2}{\lambda_{min}}\leq\frac{1}{10}$. This implies that $0.9-\frac{2\sigma_\theta\kappa(Q)^2L_G^2}{\lambda_{min}}\geq 0.8$. We also know that $\lambda_{min}^2\leq\kappa(Q)^2L_G^2$ by (3.19), so $2\sigma_\theta\leq\frac{\lambda_{min}}{10\kappa(Q)^2L_G^2}\leq\frac{1}{10\lambda_{min}}\leq\frac{1}{\lambda_{min}}$. Now we can lower bound and upper bound (4.18):

$$\begin{aligned}
&0.8\sigma_\theta B\,\mathbb{E}\,\Delta_{m+1}^\top Q^{-\top}\Lambda Q^{-1}\Delta_{m+1}+(1+0.8\sigma_\theta B\lambda_{min})\,\mathbb{E}\left\|Q^{-1}\Delta_{m+1}\right\|^2\\
&\leq\mathbb{E}\left\|Q^{-1}\Delta_m\right\|^2+\sigma_\theta B\left(0.1+\frac{I(B<n)42\kappa(Q)^2L_G^2}{B\lambda_{min}^2}\right)\mathbb{E}\,\Delta_m^\top Q^{-\top}\Lambda Q^{-1}\Delta_m\\
&+\frac{I(B<n)42\sigma_\theta}{\lambda_{min}}\left\|Q^{-1}\right\|^2\mathcal{H}\\
&\leq\mathbb{E}\left\|Q^{-1}\Delta_m\right\|^2+0.7\sigma_\theta B\,\mathbb{E}\,\Delta_m^\top Q^{-\top}\Lambda Q^{-1}\Delta_m+\frac{I(B<n)42\sigma_\theta}{\lambda_{min}}\left\|Q^{-1}\right\|^2\mathcal{H} \qquad (4.19)
\end{aligned}$$

The last inequality of the above derivations follows from the fact that we set $B\geq\frac{70\kappa(Q)^2L_G^2}{\lambda_{min}^2}$.

Since we have set $\sigma_\theta\leq\frac{5}{28B\lambda_{max}}\leq\frac{5}{28B\lambda_{min}}$, we have $\frac{4}{5}\sigma_\theta B\lambda_{min}\leq\frac{1}{7}$. This implies that $1+0.8\sigma_\theta B\lambda_{min}\leq\frac{8}{7}$. We thus have $(1+0.8\sigma_\theta B\lambda_{min})\times 0.7\leq\frac{8}{7}\times 0.7=0.8$. Now we can lower bound the left hand side of (4.19):

$$\begin{aligned}
&(1+0.8\sigma_\theta B\lambda_{min})\left(\mathbb{E}\left\|Q^{-1}\Delta_{m+1}\right\|^2+0.7\sigma_\theta B\,\mathbb{E}\,\Delta_{m+1}^\top Q^{-\top}\Lambda Q^{-1}\Delta_{m+1}\right)\\
&\leq\mathbb{E}\left\|Q^{-1}\Delta_m\right\|^2+0.7\sigma_\theta B\,\mathbb{E}\,\Delta_m^\top Q^{-\top}\Lambda Q^{-1}\Delta_m+\frac{I(B<n)42\sigma_\theta}{\lambda_{min}}\left\|Q^{-1}\right\|^2\mathcal{H} \qquad (4.20)
\end{aligned}$$

Enrolling (4.20) from $m = 0$ to $M - 1$, we have:

$$\begin{aligned}
&(1 + 0.8\sigma_\theta B\lambda_{min})^M \left(\mathbb{E}\left\|Q^{-1}\Delta_{M-1}\right\|^2 + 0.7\sigma_\theta B\,\mathbb{E}\,\Delta_{M-1}^\top Q^{-\top}\Lambda Q^{-1}\Delta_{M-1}\right) \\
&\le \mathbb{E}\left\|Q^{-1}\Delta_0\right\|^2 + 0.7\sigma_\theta B\,\mathbb{E}\,\Delta_0^\top Q^{-\top}\Lambda Q^{-1}\Delta_0 + \frac{I(B<n)42\sigma_\theta}{\lambda_{min}}\left\|Q^{-1}\right\|^2\mathcal{H}\sum_{j=0}^{M-1}(1+0.8\sigma_\theta B\lambda_{min})^j \\
&\le (1 + 0.7\sigma_\theta B\lambda_{max})\mathbb{E}\left\|Q^{-1}\Delta_0\right\|^2 + \frac{I(B<n)42\sigma_\theta}{\lambda_{min}}\left\|Q^{-1}\right\|^2\mathcal{H}\frac{(1+0.8\sigma_\theta B\lambda_{min})^M}{0.8\sigma_\theta B\lambda_{min}} \qquad (4.21)
\end{aligned}$$

In the second inequality, we use the fact that Λ is a diagonal matrix consisting of eigenvalues, which implies that: $\mathbb{E}\,\Delta_0^\top Q^{-\top}\Lambda Q^{-1}\Delta_0 \le \lambda_{max}\mathbb{E}\left\|Q^{-1}\Delta_0\right\|^2$. Divide both sides of the above inequality by $(1 + 0.8\sigma_\theta B\lambda_{min})^M$. We have:

$$\begin{aligned}
&\mathbb{E}\left\|Q^{-1}\Delta_{M-1}\right\|^2 + 0.7\sigma_\theta B\,\mathbb{E}\,\Delta_{M-1}^\top Q^{-\top}\Lambda Q^{-1}\Delta_{M-1} \\
&\le \frac{(1+0.7\sigma_\theta B\lambda_{max})}{(1+0.8\sigma_\theta B\lambda_{min})^M}\mathbb{E}\left\|Q^{-1}\Delta_0\right\|^2 + \frac{60\left\|Q^{-1}\right\|^2\mathcal{H}I(B<n)}{B\lambda_{min}^2} \qquad (4.22)
\end{aligned}$$

Obviously, $\mathbb{E}\left\|Q^{-1}\Delta_{M-1}\right\|^2 \le \mathbb{E}\left\|Q^{-1}\Delta_{M-1}\right\|^2 + 0.7\sigma_\theta B\,\mathbb{E}\,\Delta_{M-1}^\top Q^{-\top}\Lambda Q^{-1}\Delta_{M-1}$, so we have:

$$\mathbb{E}\left\|Q^{-1}\Delta_{M-1}\right\|^2 \le \frac{(1+0.7\sigma_\theta B\lambda_{max})}{(1+0.8\sigma_\theta B\lambda_{min})^M}\mathbb{E}\left\|Q^{-1}\Delta_0\right\|^2 + \frac{60\left\|Q^{-1}\right\|^2\mathcal{H}I(B<n)}{B\lambda_{min}^2} \qquad (4.23)$$

To conclude:

$$\begin{aligned}
&\mathbb{E}\left\|\theta_{M-1} - \theta^\star\right\|^2 \le \mathbb{E}\left\|QQ^{-1}\Delta_{M-1}\right\|^2 \le \left\|Q\right\|^2\mathbb{E}\left\|Q^{-1}\Delta_{M-1}\right\|^2 \\
&\le \frac{(1+0.7\sigma_\theta B\lambda_{max})\left\|Q\right\|^2}{(1+0.8\sigma_\theta B\lambda_{min})^M}\mathbb{E}\left\|Q^{-1}\Delta_0\right\|^2 + \frac{60\left\|Q\right\|^2\left\|Q^{-1}\right\|^2\mathcal{H}I(B<n)}{B\lambda_{min}^2} \\
&\le \frac{(1+0.7\sigma_\theta B\lambda_{max})\kappa(Q)^2}{(1+0.8\sigma_\theta B\lambda_{min})^M}\mathbb{E}\left\|\Delta_0\right\|^2 + \frac{60\kappa(Q)^2\mathcal{H}I(B<n)}{B\lambda_{min}^2} \qquad (4.24)
\end{aligned}$$

□

We used Lemma 4 in the proof above, so we need to show that $\mathbb{E}\|Q^{-1}\Delta_{m,K_m}\|^2 < \infty$ for any m. See [27] for the proof of this property. We used a bound on the error term $\mathbb{E}\left\|Q^{-1}e_m\right\|^2$ in the proof above. Now we present the proof of this bound.

Lemma 5.

$$\mathbb{E}\left\|Q^{-1}e_m\right\|^2 \le \frac{2I(B<n)}{B}\kappa(Q)^2 L_G^2\,\mathbb{E}\left\|Q^{-1}\Delta_m\right\|^2 + \frac{2I(B<n)\|Q^{-1}\|^2}{B}\mathcal{H}$$

Proof.

$$\begin{aligned}
\mathbb{E}\left\|Q^{-1}e_m\right\|^2 &= \mathbb{E}\left\|Q^{-1}e_m + Q^{-1}(G_m z^\star - g_m) - Q^{-1}(G_m z^\star - g_m)\right\|^2\\
&\le 2\,\mathbb{E}\left\|Q^{-1}e_m + Q^{-1}(G_m z^\star - g_m)\right\|^2 + 2\,\mathbb{E}\left\|Q^{-1}(G_m z^\star - g_m)\right\|^2\\
&= 2\,\mathbb{E}\left\|Q^{-1}e_m + Q^{-1}(G_m z^\star - g_m) - \underbrace{(Gz^\star - g)}_{=0}\right\|^2 + 2\,\mathbb{E}\left\|Q^{-1}(G_m z^\star - g_m)\right\|^2\\
&= 2\,\mathbb{E}\left\|Q^{-1}(G_m - G)\Delta_m\right\|^2 + 2\,\mathbb{E}\left\|Q^{-1}(G_m z^\star - g_m)\right\|^2\\
&= 2\,\mathbb{E}\,\mathbb{E}_{\mathcal{B}}\left\|\frac{1}{B}\sum_{t\in\mathcal{B}} Q^{-1}(G_t - G)\Delta_m\right\|^2 + 2\,\mathbb{E}\,\mathbb{E}_{\mathcal{B}}\left\|\frac{1}{B}\sum_{t\in\mathcal{B}} Q^{-1}(G_t z^\star - g_t)\right\|^2 \qquad (4.25)
\end{aligned}$$

Where $\mathbb{E}$ is expectation over $\mathcal{B}$. As

$$\frac{1}{n}\sum_{t=1}^{n} Q^{-1}(G_t - G)\Delta_m = Q^{-1}(G-G)\Delta_m = 0 \quad \text{and} \quad \frac{1}{n}\sum_{t=1}^{n} Q^{-1}(G_t z^\star - g_t) = Gz^\star - g = 0 \qquad (4.26)$$

then, by applying Lemma 3, we obtain

$$\begin{aligned}
\mathbb{E}\left\|Q^{-1}e_m\right\|^2 &\le \frac{2I(B<n)}{B}\,\mathbb{E}\,\frac{1}{n}\sum_{t=1}^{n}\|Q^{-1}(G_t - G)\Delta_m\|^2 + \frac{2I(B<n)\|Q^{-1}\|^2}{B}\underbrace{\frac{1}{n}\sum_{t=1}^{n}\|G_t z^\star - g_t\|^2}_{=\mathcal{H}}\\
&= \frac{2I(B<n)}{B}\,\mathbb{E}\,\frac{1}{n}\sum_{t=1}^{n}\|Q^{-1}(G_t - G)\Delta_m\|^2 + \frac{2I(B<n)\|Q^{-1}\|^2}{B}\mathcal{H}\\
&\le \frac{2I(B<n)}{B}\,\mathbb{E}\,\frac{1}{n}\sum_{t=1}^{n}\|Q^{-1}G_t\Delta_m\|^2 + \frac{2I(B<n)\|Q^{-1}\|^2}{B}\mathcal{H}\\
&\le \frac{2I(B<n)}{B}\kappa(Q)^2 L_G^2\,\mathbb{E}\,\|Q^{-1}\Delta_m\|^2 + \frac{2I(B<n)\|Q^{-1}\|^2}{B}\mathcal{H} \qquad (4.27)
\end{aligned}$$

We used in the third inequality the fact that variance of random variable is upper bounded by its second moment. We used the last inquality the fact that $\frac{1}{n}\sum_{t=1}^{n}\|Q^{-1}G_t\Delta_m\|^2 \leq \kappa(Q)^2 L_G^2\,\mathbb{E}\,\|Q^{-1}\Delta_m\|^2$ as shown in [15]. □

Based on the result in Theorem 3, we show the computational cost of Algorithm 3 in the following corollary.

Corollary 1. *Suppose Assumption 1 holds. Set* $\sigma_\theta = \min\left\{\frac{\lambda_{min}}{20\kappa(Q)^2 L_G^2}, \frac{5}{28B\lambda_{max}}\right\}$, $\sigma_\omega = \beta\sigma_\theta$ *and* $B = \min\left\{\max\left\{\frac{70\kappa(Q)^2L_G^2}{\lambda_{min}^2}, \frac{120\kappa(Q)^2\mathcal{H}}{\lambda_{min}^2\epsilon}\right\}, n\right\}$. *Let* $\epsilon > 0$, *the computational cost in expectations required to obtain* $\mathbb{E}\,\|\theta_{M-1} - \theta^\star\|^2 \leq \epsilon$ *is:*

$$O\left(\left(\min\left\{\frac{\kappa(Q)^2\mathcal{H}}{\lambda_{min}^2\epsilon}, n\right\} + \frac{\kappa(Q)^2L_G^2}{\lambda_{min}^2}\right)\log\left(\frac{\kappa(Q)^2\mathbb{E}\,\|\Delta_0\|^2}{\epsilon}\right)\right)$$

Proof. From Theorem 3, we know that:

$$\mathbb{E}\,\|\theta_{M-1} - \theta^\star\|^2 \leq \frac{(1+0.7\sigma_\theta B\lambda_{max})\kappa(Q)^2}{(1+0.8\sigma_\theta B\lambda_{min})^M}\mathbb{E}\,\|\Delta_0\|^2 + \frac{60\kappa(Q)^2\mathcal{H}I(B<n)}{B\lambda_{min}^2} \tag{4.28}$$

If $\max\left\{\frac{70\kappa(Q)^2L_G^2}{\lambda_{min}^2}, \frac{120\kappa(Q)^2\mathcal{H}}{\lambda_{min}^2\epsilon}\right\} < n$, then $B \geq \frac{120\kappa(Q)^2\mathcal{H}}{\lambda_{min}^2\epsilon}$, implying that $\frac{60\kappa(Q)^2\mathcal{H}}{B\lambda_{min}^2} \leq \frac{\epsilon}{2}$. We can thus write the above inequality equivalently as:

$$\mathbb{E}\,\|\theta_{M-1} - \theta^\star\|^2 \leq \frac{(1+0.7\sigma_\theta B\lambda_{max})\kappa(Q)^2}{(1+0.8\sigma_\theta B\lambda_{min})^M}\mathbb{E}\,\|\Delta_0\|^2 + \frac{\epsilon}{2} \tag{4.29}$$

In the case where $\max\left\{\frac{70\kappa(Q)^2L_G^2}{\lambda_{min}^2}, \frac{120\kappa(Q)^2\mathcal{H}}{\lambda_{min}^2\epsilon}\right\} \geq n$, $\frac{30\kappa(Q)^2\mathcal{H}I(B<n)}{B\lambda_{min}^2} = 0$, so the above inequality still holds.

By our choice of σ_θ, $\sigma_\theta \leq \frac{5}{28B\lambda_{max}}$, so $\sigma_\theta B\lambda_{max} \leq \frac{5}{28}$. Right hand side of (4.29) can be bounded as:

$$\mathbb{E}\,\|\theta_{M-1} - \theta^\star\|^2 \leq \frac{1.125\times\kappa(Q)^2}{(1+0.8\sigma_\theta B\lambda_{min})^M}\mathbb{E}\,\|\Delta_0\|^2 + \frac{\epsilon}{2} \tag{4.30}$$

$\mathbb{E}\left\|\theta_{M-1}-\theta^\star\right\|^2 \le \epsilon$ if $M \ge \frac{\log\left(\frac{2.25\kappa(Q)^2\mathbb{E}\|\Delta_0\|^2}{\epsilon}\right)}{\log(1+0.8\sigma_\theta B\lambda_{min})}$, so the number of outer loop iterations that Algorithm 3 needs to take in order to reach an ϵ-optimal solution is:

$$\begin{aligned}\frac{\log\left(\frac{2.25\kappa(Q)^2\mathbb{E}\|\Delta_0\|^2}{\epsilon}\right)}{\log\left(1+0.8\sigma_\theta B\lambda_{min}\right)} &= O\left(\left(1+\frac{1}{\sigma_\theta B\lambda_{min}}\right)\log\left(\frac{\kappa(Q)^2\mathbb{E}\left\|\Delta_0\right\|^2}{\epsilon}\right)\right)\\ &= O\left(\left(1+\frac{\kappa(Q)^2L_G^2}{B\lambda_{min}^2}\right)\log\left(\frac{\kappa(Q)^2\mathbb{E}\left\|\Delta_0\right\|^2}{\epsilon}\right)\right)\end{aligned}$$

The second equality follows from $\sigma_\theta B\lambda_{min} = \min\left\{\frac{\lambda_{min}}{20\kappa(Q)^2L_G^2}, \frac{5}{28B\lambda_{max}}\right\}B\lambda_{min}$. If $\frac{B\lambda_{min}^2}{20\kappa(Q)^2L_G^2} < \frac{5\lambda_{min}}{28\lambda_{max}} < 1$, $\frac{1}{\sigma_\theta B\lambda_{min}} = O\left(\frac{\kappa(Q)^2L_G^2}{B\lambda_{min}^2}\right)$. Otherwise, $\frac{1}{\sigma_\theta B\lambda_{min}} = O(1)$.

In each iteration of the outer loop in Algorithm 3, a batch of B gradients are computed before entering the inner loop, then, the inner loop executes for $O(B)$ iterations in expectations. In total, $O(B)$ gradient evaluations are required in expectations during each epoch of Algorithm 3. The total computation cost in expectation is:

$$\begin{aligned}&O\left(\left(B+\frac{\kappa(Q)^2L_G^2}{\lambda_{min}^2}\right)\log\left(\frac{\kappa(Q)^2\mathbb{E}\left\|\Delta_0\right\|^2}{\epsilon}\right)\right)\\ &= O\left(\left(\min\left\{\max\left\{\frac{70\kappa(Q)^2L_G^2}{\lambda_{min}^2}, \frac{120\kappa(Q)^2\mathcal{H}}{\lambda_{min}^2\epsilon}\right\}, n\right\}+\frac{\kappa(Q)^2L_G^2}{\lambda_{min}^2}\right)\log\left(\frac{\kappa(Q)^2\mathbb{E}\left\|\Delta_0\right\|^2}{\epsilon}\right)\right)\\ &= O\left(\left(\min\left\{\frac{\kappa(Q)^2\mathcal{H}}{\lambda_{min}^2\epsilon}, n\right\}+\frac{\kappa(Q)^2L_G^2}{\lambda_{min}^2}\right)\log\left(\frac{\kappa(Q)^2\mathbb{E}\left\|\Delta_0\right\|^2}{\epsilon}\right)\right)\end{aligned}$$

□

Table 4–1 contains the computational cost of Algorithm 2 and 3, along with costs of SVRG [15] and GTD2 [38]. All four algorithms are designed for policy evaluation with linear function approximations. We use quantities in our result to represent GTD2's computational costs. Since SVRG computes the full gradient in every epoch, its cost contains n, which could be computationally expensive when n is large. Costs of GTD2 and SCSG do not rely on n, however, GTD2 has a sublinear convergence rate and SCSG has a faster linear convergence rate. When n is large and the required accuracy ϵ is low, SCSG saves unnecessary computations and is able to achieve the target accuracy with potentially less than a single pass through the dataset.

Algorithm	Computational Cost
GTD2	$O(\frac{\kappa(Q)^2\mathcal{H}}{\lambda_{\min}^2\epsilon})$
SVRG	$O\left(\left(n+\frac{\kappa(Q)^2L_G^2}{\lambda_{\min}^2}\right)\ln(1/\epsilon)\right)$
Batching SVRG	$O\left(\left(n'+\frac{\kappa(Q)^2L_G^2}{\lambda_{\min}^2}\right)\ln(1/\epsilon)\right)$
SCSG	$O\left(\left(\frac{\kappa(Q)^2\mathcal{H}}{\lambda_{\min}^2\epsilon}\wedge n+\frac{\kappa(Q)^2L_G^2}{\lambda_{\min}^2}\right)\ln(1/\epsilon)\right)$

Table 4–1: Computational cost of different policy evaluation algorithms.

Now we compare the computational costs of Batching SVRG and SVRG. In epoch m, Batching SVRG's computational cost is $O(B_m+K_m)$, where B_m is the batch size and K_m is the number of inner loop iteration. According to Theorem 1, K_m is set as $O(\frac{\kappa(Q)^2L_G^2}{\lambda_{\min}^2})$ for all m. If we use the exponentially increasing sequence of batch sizes that we considered in Theorem 2, B_m is strictly less than n, for all m. Since Batching SVRG converges linearly, it takes $O(\ln(1/\epsilon))$ epochs to reach an ϵ-optimal solution. The overall computational cost of Batching SVRG is $O\left(\left(n'+\frac{\kappa(Q)^2L_G^2}{\lambda_{\min}^2}\right)\ln(1/\epsilon)\right)$, where $n'<n$, so Batching SVRG is computationally more efficient than the vanilla SVRG.

Task	GTD2	SVRG	SAGA	Batching SVRG	SCSG	LSTD
Mountain Car	348 ± 181	186 ± 149	$\mathbf{166 \pm 125}$	170 ± 135	276 ± 178	292 ± 187
Cart Pole	-163 ± 95	$\mathbf{-280 \pm 88}$	-246 ± 75	$\mathbf{-283 \pm 82}$	-217 ± 100	-183 ± 80
Acrobot	431 ± 128	$\mathbf{96 \pm 6}$	101 ± 6	$\mathbf{97 \pm 6}$	126 ± 78	176 ± 157
Mountain Car (large data)	264 ± 178	500 ± 0	500 ± 0	349 ± 189	304 ± 182	$\mathbf{116 \pm 6}$
Cart Pole (large data)	-198 ± 98	-9 ± 0	-9 ± 0	-267 ± 93	-215 ± 91	$\mathbf{-290 \pm 26}$
Acrobot (large data)	416 ± 137	500 ± 0	500 ± 0	121 ± 77	$\mathbf{99 \pm 6}$	$\mathbf{91 \pm 4}$

Table 4–2: All methods' performances in control. First column shows names of all tasks. All values in the table are number of steps (or the negative number of steps) that each method takes to reach the terminal state. Small values mean good performances in control. Experimental details can be found in Section 4.3.3.

4.3 Experiments

4.3.1 Observations

To compare effectiveness of all methods, we show their performances in control tasks, listed in Table 4–2. We have the following observations:

1. LSTD is outperformed by gradient based methods in some tasks. All gradient based methods solve the equivalent objective function with LSTD, but LSTD computes the solution analytically, so we should expect LSTD to have better performances than gradient based methods in control tasks, given we only run gradient based methods for 100 passes of the data set. This is not the case as SAGA and Batching SVRG outperforms LSTD in Mountain Car. SVRG and Batching SVRG outperform LSTD in Cart Pole and Acrobot.

2. Well-conditioned empirical matrices $\hat{A}$ and $\hat{C}$ help gradient based methods in some tasks. $\hat{A}$ is singular and $\hat{C}$ is not positive definite in small-data settings. We add $10^{-5} \times I$ to $\hat{A}$ and $\hat{C}$ if they are not well-conditioned. From Table C–1, we observe that all gradient based methods have better control performances in Cart Pole after we make $\hat{A}$ and $\hat{C}$ well-conditioned.

3. GTD2, Batching SVRG and SCSG can handle large data settings. In Mountain Car (large data), Cart Pole (large data) and Acrobot (large data), we sample $n = 1,000,000$ data. Figure 4–2 shows learning curves (policy evaluation results) of all gradient based methods in large data settings. In large data settings, we allow all gradient based methods to use the data set only once. SVRG and SAGA need to use the entire data set at the beginning, so there is no learning and they have poor control performances. GTD2, Batching SVRG and SCSG do not rely on full gradients and are able make progress instantly. No methods except LSTD solve the large data setting of Mountain Car well. Batching SVRG reaches a same level of performances with LSTD in Cart Pole (large data). SCSG has a better control performance than LSTD in Acrobot (large data).

4. Policy evaluation performances are not always related with control performances. In all experiments, we generate data by performing random actions in the environment. We then run gradient based methods to optimize the saddle-point formulation of EM-MSPBE, given in (2.10). Finally, we take gradient based methods' results from policy evaluation and evaluate them in control tasks. Figure 4–1 and Figure 4–2 show policy evaluation results in small and large data settings respectively. Low MSPBE values indicate that the value function is approximated well, so we should expect good performances in control. This is not always true in our results. For example, GTD2 has the best policy evaluation performances in Acrobot (shown in Figure 4–1), but it has poor control performances in Acrobot. In large data settings, Figure 4–2 shows that GTD2 has the most stable policy evaluation performance in Cart Pole, but its control performance is not as good as Batching SVRG and SCSG in this task. Unlike LSTD, gradient based methods learn policies in an iterative way. This may result in learning a good value function with a large MSPBE, which may cause gradient based methods to outperform LSTD in certain control tasks.

5. No gradient based methods solve Mountain Car (large data). As Table 4–2 shows, no methods except LSTD solve Mountain Car (large data). As we identified in the previous observation, good policy evaluation performances (i.e. low MSPBE values) do not always correlate with good control performances. We choose step sizes that achieve the lowest MSPBE value in a validation data set for all gradient based methods, so we might have choose step sizes that are too small for gradient based methods to have good control performances in Mountain Car (large data).

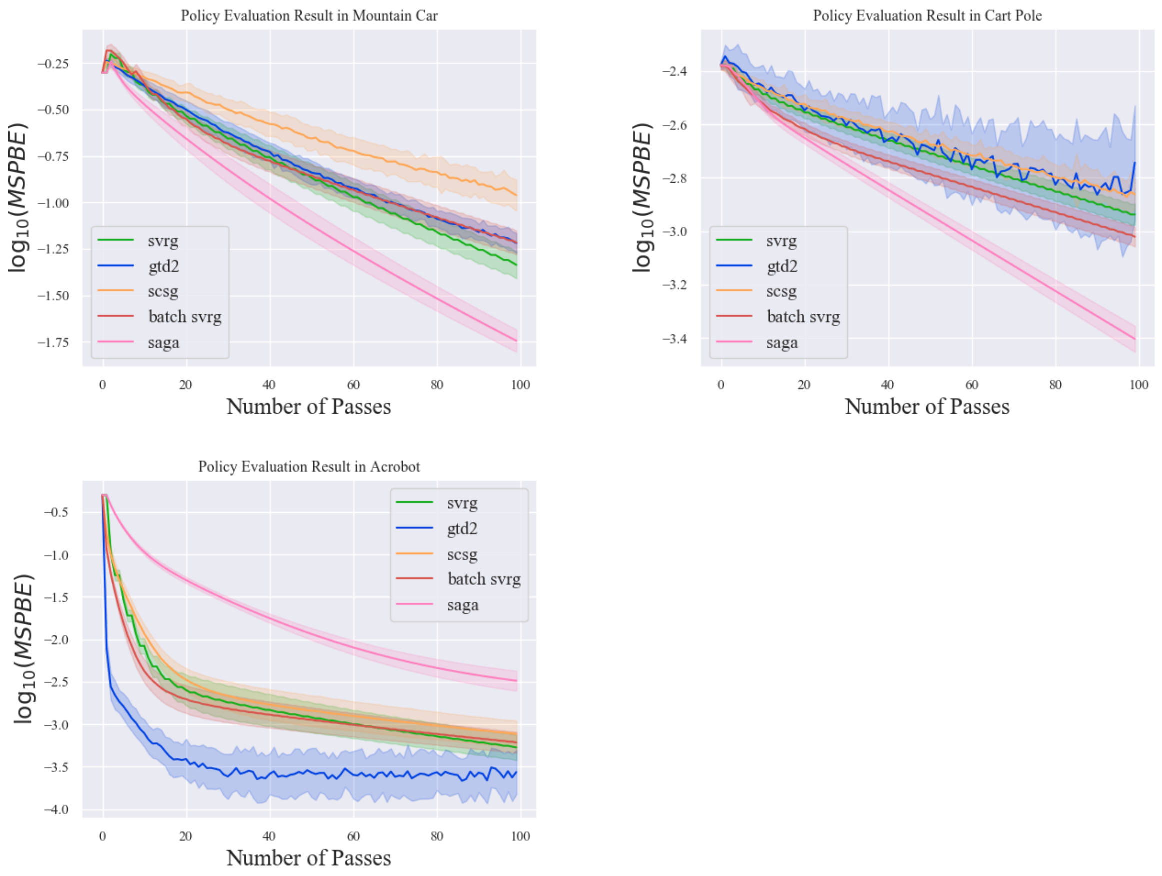

Figure 4–1: Policy evaluation results in Mountain Car, Cart Pole and Acrobot.

4.3.2 Discussion

An advantage of gradient based methods over LSTD is that gradient based methods are able to handle ill-conditioned control problems, as Table C–1 illustrates. LSTD is not able to solve ill-conditioned problems since it requires inverting $\hat{A}$, but gradient based methods are able to achieve passable performances in such control tasks.

Our proposed methods and LSTD both use the data set once and solve the control tasks in large data settings, while other gradient based methods need more than one pass of the data set. More importantly, unlike LSTD, our methods are first order methods and do not need to invert matrices, which makes our methods practical when both the size of the data set and the number of state's features are large.

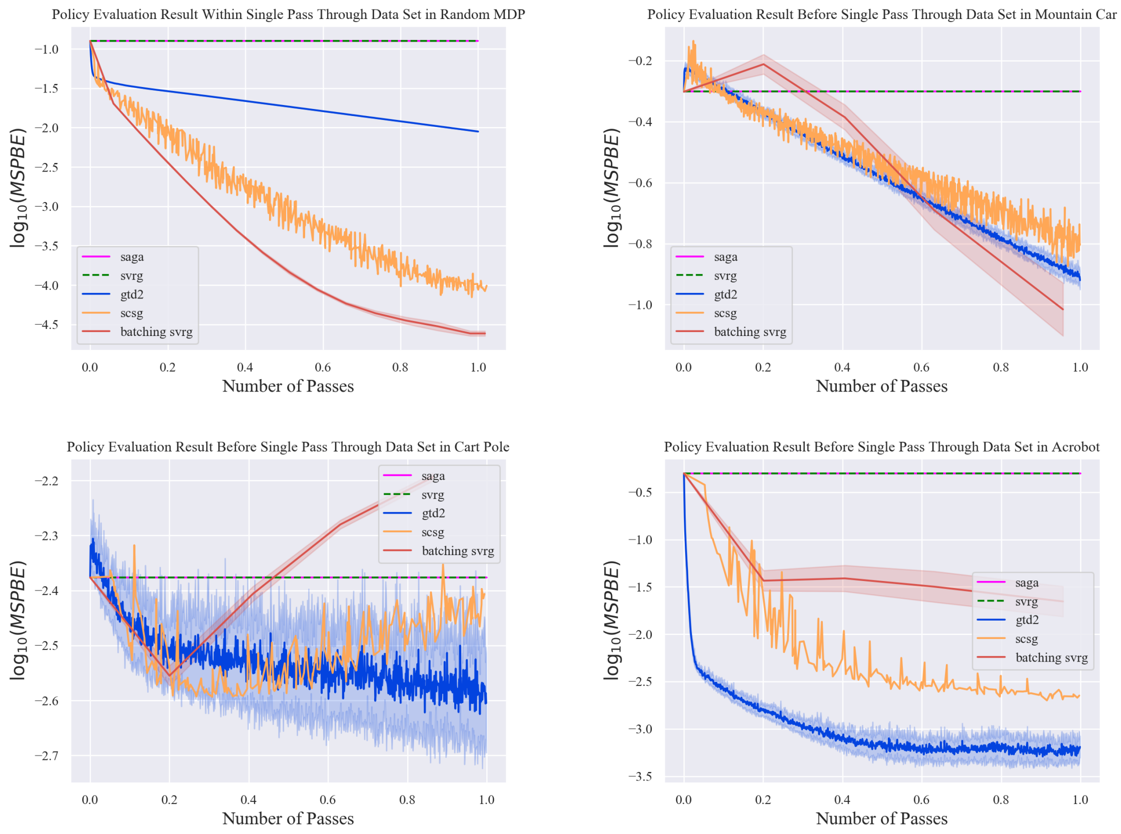

Figure 4–2: Policy evaluation results in single pass through the large data set. For Random MDP experiments, we sample 10 million data from the environment. For Mountain Car, Cart Pole and Acrobot experiments, we sample 1 million data.

Performances of gradient based methods are quite sensitive to feature values. We applied radial basis functions to raw observation values in our control tasks, then we normalize feature vectors so they sum to 1. Without normalization, gradient based methods have terrible policy evaluation performances. This is also observed by Dann et al in [13].

Besides comparing all methods on toy control tasks, we also run them in Random MDP. Figure 4–3 shows control performances. We run policy evaluation for all methods, then we run policy improvement. This process is repeated for 5 episodes. Figure 4–4 shows policy evaluation performances in each episode of policy iteration. We observe that all gradient based methods have similar performances in control with LSTD, despite SVRG and Batching SVRG have better policy evaluation performances.

In large data setting of Random MDP, we sample 10 million data from a policy which performs random actions in the Random MDP environment. Policy evaluation result is given in Figure 4–2. Each method is allowed to use the data set once. We observe that Batching SVRG and SCSG converge faster than GTD2.

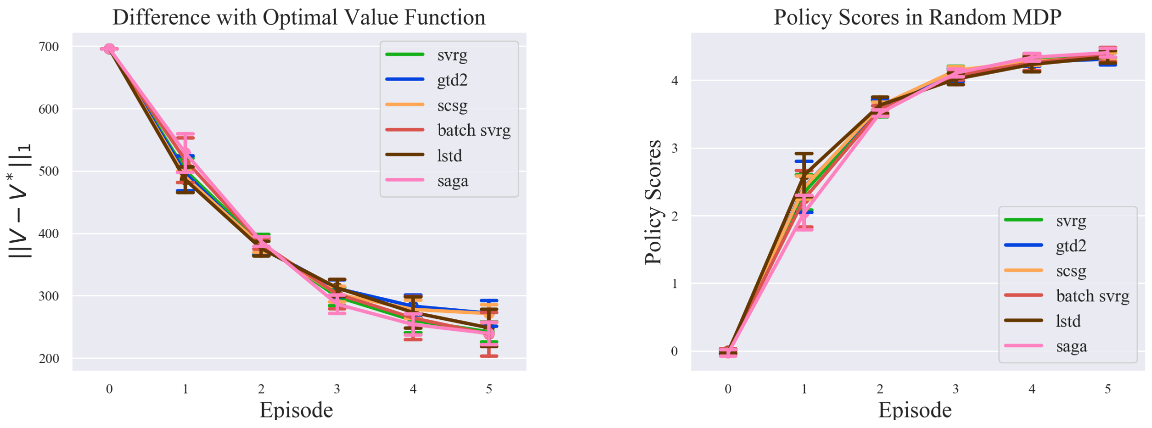

Figure 4–3: Performances in Random MDP control. The optimal value function in Random MDP is obtained from value iteration and we report the L1 difference between each method's value function and the optimal value function. Policy scores are computed by averaging rewards received after executing each method's learned policy for 200 time steps and 100 independent runs.

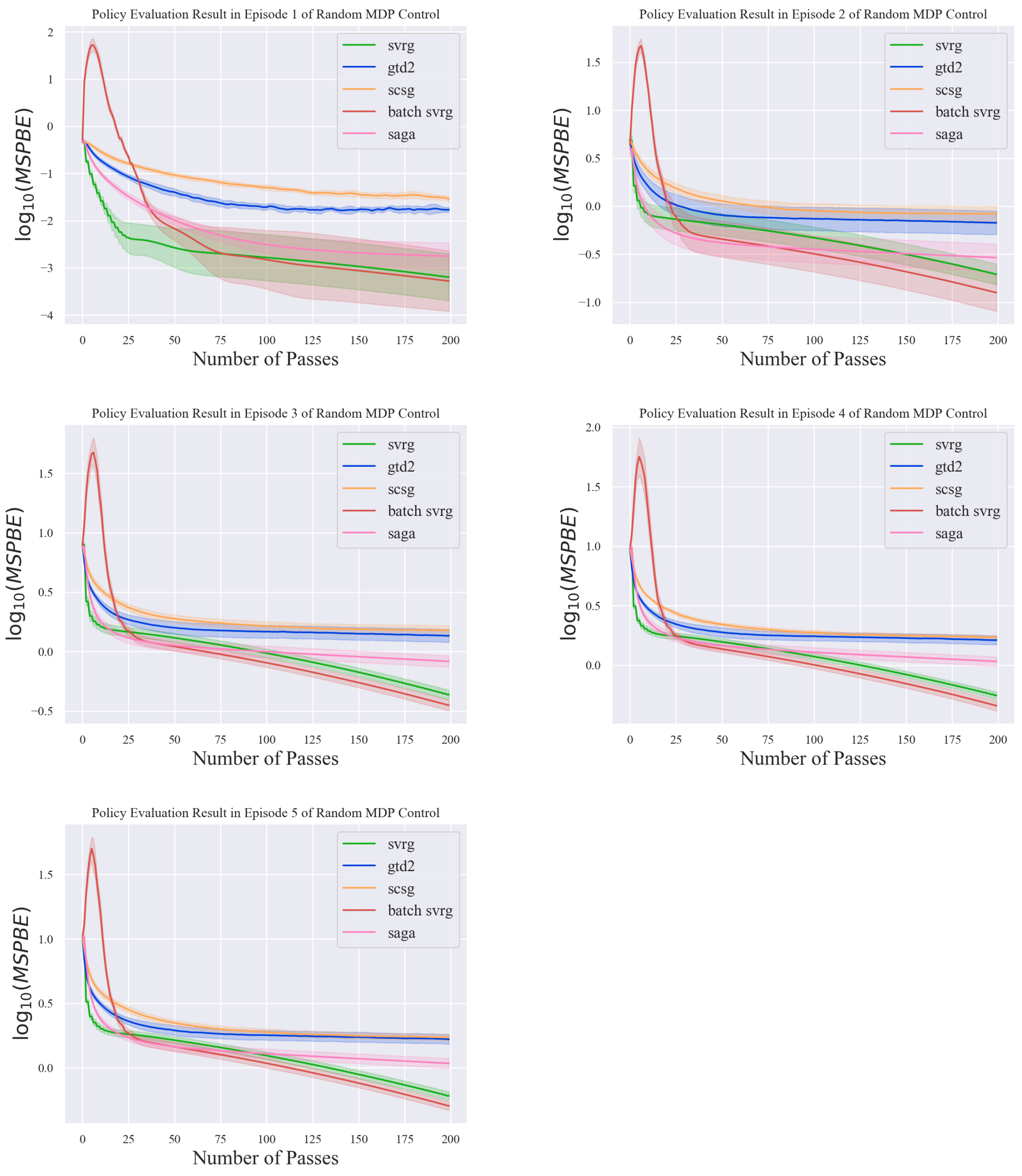

Figure 4–4: Policy evaluation results in Random MDP control. We run 5 episodes of policy iteration and the above figures show policy evaluation results in each episode of policy iteration.

4.3.3 Experiment Details

1. **Environments**:

(a) Random MDP environment [13]. A randomly generated MDP with 400 states, 10 actions. Each state has a 201 dimensional feature vector where each entry except the last one is uniformly sampled from $[0, 1]$. The last entry of every state's feature vector is set to 1. Transition probabilities and rewards are uniformly sampled from $[0, 1]$. Discount factor is 0.95.

(b) We used MountainCar-v0, CartPole-v1 and Acrobot-v1 from Open AI Gym [9] to simulate Mountain Car, Cart Pole and Acrobot environments. Discount factor is set to 0.99 in Mountain Car and Cart Pole experiments. In Acrobot experiments, discount factor is 0.9. We changed the reward function in Cart Pole experiments. The agent receives a reward of -1 when it fails and 0 otherwise.

2. **Data collection process.** In all experiments, we evaluate the data generated by a policy which performs random actions. We collect $n = 20,000$ data samples in small-data experiments, and we collect $n = 1,000,000$ data samples in large-data experiments.

3. **Feature engineering process.** We do not modify state's feature vector in Random MDP environment. In Mountain Car and Acrobot experiments, we scale state's features to $[0, 1]$. This process is not included in Cart Pole experiments. Then, we apply state's feature to 10 (in Mountain Car environment) and 3 (in Cart Pole and Acrobot environment) evenly spaced RBF kernels, which is defined as $f(x) = \exp(-r/\sigma)$. $\sigma = 0.01$ in Mountain Car environment. $\sigma = 0.5$ in Cart Pole environment. $\sigma = 0.1$ in Acrobot environment. We finally normalize state's feature values so they sum to 1.

4. **Parameter selection** Parameters are set after grid searching. Grid searching results can be found in Appendix A. Results are aggregated from 10 independent runs. In each run, each method is started from a randomly initialized vector. Detailed parameter settings can be found in Appendix B.

5. **Initialization:** In all experiments, we initialize θ and ω to zero vectors.

6. **Description of how experiments were run.** We generate data and set parameters by following procedures that we described above. In Mountain Car, Cart Pole and Acrobot experiments, we run each method for 50 times in small-data experiments and we run each method for 20 times in large-data experiments. In Random MDP control experiments, we alternate between policy evaluation and policy improvement. This process is repeated for 5 episodes and we run each method for 10 times. In Random MDP large-data experiments, we run each method for 10 times.

CHAPTER 5
Conclusion and Future Work

In this thesis, we show that Batching SVRG and SCSG converge linearly when solving the saddle-point formulation of MSPBE. This problem is convex-concave and is not strongly convex in the primal variable, so it is very different with the original objective function that Batching SVRG and SCSG attempt to solve. We showed that our proposed methods could be more efficient than existing gradient based methods in policy evaluation and control tasks, especially in large data settings. In general, there is a lot of room for applying more efficient optimization algorithms to problems in reinforcement learning, in order to obtain better theoretical guarantees and to improve sample and computational efficiency.

References

[1] Zeyuan Allen-Zhu. Katyusha: The first direct acceleration of stochastic gradient methods. *arXiv:1603.05953*, 2016. 7

[2] Cauchy Augustin. M´ethode g´en´erale pour la r´esolution des syst‘emes d’´equations simultan´ees. *Comptes rendus des s´eances de l’Acad´emie des sciences de Paris*, 1847. 7

[3] P. Balamurugan and Francis Bach. Stochastic variance reduction methods for saddle-point problems. In *Advances in Neural Information Processing Systems*, 2016. 9

[4] Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 1983. 22

[5] Jalaj Bhandari, Daniel Russo, and Raghav Singal. A finite time analysis of temporal difference learning with linear function approximation. In *Conference on Learning Theory*, 2018. 3

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

[7] Steven J. Bradtke and Andrew G. Barto. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22(1-3):33–57, 1996. 2

[8] Steven J. Bradtke and Andrew G. Barto. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22(1-3):33–57, 1996. 2

[9] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016. 22, 44

[10] Lucas Cassano, Kun Yuan, and Ali H. Sayed. Multi-agent fully decentralized value function learning with linear convergence rates. *arXiv:1810.07792*, 2018. 3

[11] Bo Dai, Albert Shaw, Lihong Li, Lin Xiao, Niao He, Zhen Liu, Jianshu Chen, and Le Song. Sbeed: Convergent reinforcement learning with nonlinear function approximation. *International Conference on Machine Learning*, 2018. 3

[12] Gal Dalal, Balazs Szorenyi, and Gugan Thoppe. A tale of two-timescale reinforcement learning with the tightest finite-time bound. In *arxiv:1911.09157*, 2019. 3

[13] Christoph Dann, Gerhard Neumann, and Jan Peters. Policy evaluation with temporal differences: a survey and comparison. *Journal of Machine Learning Research*, 15(1):809–883, 2014. 22, 41, 44

[14] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, 2014. 3, 7

[15] Simon S. Du, Jianshu Chen, Lihong Li, Lin Xiao, and Dengyong Zhou. Stochastic variance reduction methods for policy evaluation. In *International Conference on Machine Learning*, 2017. 1, 3, 9, 10, 13, 14, 16, 17, 35, 36

[16] Sina Ghiassian, Andrew Patterson, Martha White, Richard S. Sutton, and Adam White. Online off-policy prediction. *arXiv:1811.02597*, 2018. 3

[17] Reza Harikandeh, Mohamed Osama Ahmed, Alim Virani, Mark Schmidt, Jakub Konečný, and Scott Sallinen. Stop wasting my gradients: Practical svrg. In *Advances in Neural Information Processing Systems*, 2015. 2, 11, 18

[18] Howard Huang and Doina Precup. An empirical study of least-squares algorithms in reinforcement learning. *European Workshop on Reinrocement Learning*, 2018. 23

[19] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, 2013. 3, 7, 8

[20] Anatoli Juditsky, Arkadi Nemirovski, and Claire Tauvel. Solving variational inequalities with stochastic mirror-prox. *Stochastic Systems*, 1(1):17–58, 2011. 3

[21] Nathaniel Korda and L.A Prashanth. On td(0) with function approximation: Concentration bounds and a centered variant with exponential convergence. In *International Conference on Machine Learning*, 2015. 3

[22] Lihua Lei and Michael I. Jordan. Less than a single pass: Stochastically controlled stochastic gradient method. In *International Conference on Artificial Intelligence and Statistics*, 2017. 2, 20, 26, 27

[23] Hongzhou Lin, Julien Mairal, and Zaid Harchaoui. A universal catalyst for first-order optimization. In *Advances in Neural Information Processing Systems*, 2015. 7

[24] Bo Liu, Ji Liu, Mohammad Ghavamzadeh, Sridhar Mahadevan, and Marek Petrik. Finite-sample analysis of proximal gradient td algorithms. In *Conference on Uncertainty in Artificial Intelligence*, 2015. 3

[25] Andrew Moore. Efficient memory-based learning for robot control. *Technical report*, 1990. 22

[26] Matteo Papini, Damiano Binaghi, Giuseppe Canonaco, Matteo Pirotta, and Marcello Restelli. Stochastic variance-reduced policy gradient. *International Conference on Machine Learning*, 2018. 3

[27] Zilun Peng, Ahmed Touati, Pascal Vincent, and Doina Precup. Svrg for policy evaluation with fewer gradient evaluations. *arXiv:1906.03704*, 2019. 4, 34

[28] Doina Precup. Eligibility traces for off-policy policy evaluation. . *Computer Science Department Faculty Publication Series*, page 80, 2000. 3

[29] Herbert Robbins and Sutton Monro. A stochastic approximation method. In *Annals of Mathematical Statistics*, pages 400–407, 1951. 7

[30] Nicolas Le Roux, Mark Schmidt, and Francis Bach. Minimizing finite sums with the stochastic average gradient. In *Advances in Neural Information Processing Systems*, 2012. 7

[31] John Schulman, Philipp Moritz Sergey Levine, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. In *International Conference on Machine Learning*, 2015. 3

[32] Richard Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. *Advances in Neural Information Processing Systems*, 1996. 22

[33] Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, 1988. 2

[34] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA, 2018. 6, 22

[35] Richard S. Sutton, Hamid R. Maei, and Csaba Szepesvári. A convergent o(n) algorithm for off-policy temporal-difference learning with linear function approximation. In *Advances in Neural Information Processing Systems*, 2008. 2

[36] 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 *International Conference on Machine Learning*. ACM Press, 2009. 2, 3, 7

[37] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, 2000. 3

[38] Ahmed Touati, Pierre-Luc Bacon, Doina Precup, and Pascal Vincent. Convergent tree backup and retrace with function approximation. In *International Conference on Machine Learning*, pages 4962–4971, 2018. 3, 36

[39] Hoi-To Wai, Zhuoran Yang, Zhaoran Wan, and Mingyi Hong. Multi-agent reinforcement learning via double averaging primal-dual optimization. In *Advances in Neural Information Processing Systems*, 2018. 3

[40] Yue Wang, Wei Chen, Yuting Liu, Zhi-Ming Ma, and Tie-Yan Liu. Finite sample analysis of the gtd policy evaluation algorithms in markov setting. In *Advances in Neural Information Processing Systems*, 2017. 3

[41] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992. 4

[42] Tengyang Xie, Philip S. Thomas, and Gerome Miklau. Privacy preserving off-policy evaluation. *arXiv:1902.00174*, 2019. 3

[43] Tianbing Xu, Qiang Liu, , and Jian Peng. Stochastic variance reduction for policy gradient estimation. *arXiv:1710.06034*, 2017. 3

[44] Yuchen Zhang and Lin Xiao. Stochastic primal-dual coordinate method for regularized empirical risk minimization. In *International Conference on Machine Learning*, 2015. 7

Appendices

APPENDIX A
Parameter Sensitivity Results

A.1 Parameter sensitivity results on Random MDP environment

We run GTD2 for 200 passes of the data set. Other methods were run 50 passes of the data set. $SVRG(1*n)$ means that the number of inner loop iterations is set to n.

Table A–1: Average of last 5 percent of MSPBE in Random MDP.

σ_θ σ_ω	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}
GTD2					
10^{-1}	NaN	NaN	NaN	NaN	NaN
10^{-2}	4.052 ± 0.315	0.223 ± 0.016	0.022 ± 0.002	0.075 ± 0.011	0.333 ± 0.032
10^{-3}	2.162 ± 0.133	0.157 ± 0.007	0.017 ± 0.001	0.075 ± 0.012	0.334 ± 0.032
10^{-4}	2.084 ± 0.137	0.155 ± 0.011	0.018 ± 0.003	0.084 ± 0.013	0.339 ± 0.033
10^{-5}	1.744 ± 0.159	0.204 ± 0.022	0.079 ± 0.008	0.153 ± 0.023	0.378 ± 0.035
SVRG $(1*n)$					
10^{-1}	NaN	NaN	NaN	NaN	NaN
10^{-2}	6249.5	0.001 ± 0.001	0.024 ± 0.005	0.228 ± 0.025	0.425 ± 0.038
10^{-3}	0.075 ± 0.079	0.002 ± 0.001	0.025 ± 0.005	0.231 ± 0.025	0.426 ± 0.038
10^{-4}	0.074 ± 0.015	0.026 ± 0.004	0.060 ± 0.010	0.260 ± 0.029	0.433 ± 0.038
10^{-5}	0.159 ± 0.011	0.150 ± 0.019	0.198 ± 0.028	0.377 ± 0.035	0.454 ± 0.040
10^{-6}	0.208 ± 0.027	0.239 ± 0.031	0.361 ± 0.034	0.450 ± 0.039	0.463 ± 0.040
SVRG $(2*n)$					
10^{-1}	NaN	NaN	NaN	NaN	NaN
10^{-2}	82.8	0.001 ± 0.001	0.015 ± 0.003	0.190 ± 0.022	0.413 ± 0.037
10^{-3}	0.094 ± 0.072	0.001 ± 0.001	0.015 ± 0.003	0.193 ± 0.023	0.414 ± 0.037
10^{-4}	0.159 ± 0.062	0.020 ± 0.005	0.039 ± 0.007	0.217 ± 0.026	0.421 ± 0.038
10^{-5}	0.174 ± 0.015	0.130 ± 0.011	0.168 ± 0.023	0.334 ± 0.033	0.447 ± 0.039
SCSG 0.1					
10^{-1}	NaN	NaN	NaN	NaN	NaN
10^{-2}	5.6e+09	0.267 ± 0.051	0.052 ± 0.008	0.227 ± 0.029	0.425 ± 0.037
10^{-3}	6.6	0.231 ± 0.062	0.056 ± 0.009	0.231 ± 0.026	0.427 ± 0.038
10^{-4}	4.690 ± 0.572	0.271 ± 0.040	0.083 ± 0.012	0.260 ± 0.029	0.433 ± 0.038
10^{-5}	1.946 ± 0.363	0.273 ± 0.028	0.209 ± 0.026	0.373 ± 0.036	0.454 ± 0.039

Table A–2: Area under curve (AUC) of MSPBE in Random MDP

σ_ω \ σ_θ	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}
			GTD2		
10^{-1}	NaN	NaN	NaN	NaN	NaN
10^{-2}	4.047 ± 0.086	0.224 ± 0.005	0.05 ± 0.004	0.175 ± 0.02	0.392 ± 0.035
10^{-3}	2.15 ± 0.032	0.161 ± 0.002	0.047 ± 0.004	0.177 ± 0.02	0.392 ± 0.035
10^{-4}	1.948 ± 0.06	0.17 ± 0.006	0.068 ± 0.007	0.196 ± 0.022	0.398 ± 0.036
10^{-5}	1.297 ± 0.105	0.218 ± 0.014	0.17 ± 0.018	0.282 ± 0.03	0.427 ± 0.038
			SVRG $(1 * n)$		
10^{-1}	NaN	NaN	NaN	NaN	NaN
10^{-2}	866.9	0.026 ± 0.003	0.103 ± 0.012	0.323 ± 0.031	0.444 ± 0.039
10^{-3}	0.141 ± 0.046	0.033 ± 0.004	0.112 ± 0.014	0.328 ± 0.031	0.445 ± 0.039
10^{-4}	0.152 ± 0.026	0.095 ± 0.011	0.175 ± 0.021	0.359 ± 0.034	0.450 ± 0.039
10^{-5}	0.203 ± 0.019	0.226 ± 0.022	0.309 ± 0.032	0.430 ± 0.038	0.460 ± 0.040
10^{-6}	0.257 ± 0.024	0.315 ± 0.032	0.424 ± 0.038	0.459 ± 0.040	0.464 ± 0.040
			SVRG $(2 * n)$		
10^{-1}	NaN	NaN	NaN	NaN	NaN
10^{-2}	18.4	0.024 ± 0.003	0.084 ± 0.010	0.295 ± 0.029	0.438 ± 0.038
10^{-3}	0.193 ± 0.075	0.029 ± 0.003	0.091 ± 0.011	0.299 ± 0.029	0.439 ± 0.039
10^{-4}	0.209 ± 0.048	0.080 ± 0.009	0.145 ± 0.018	0.329 ± 0.032	0.444 ± 0.039
10^{-5}	0.207 ± 0.016	0.206 ± 0.020	0.278 ± 0.030	0.410 ± 0.037	0.458 ± 0.040
			SCSG 0.1		
10^{-1}	NaN	NaN	NaN	NaN	NaN
10^{-2}	8.9e+08	0.294 ± 0.021	0.123 ± 0.014	0.322 ± 0.035	0.443 ± 0.039
10^{-3}	6.3	0.285 ± 0.023	0.130 ± 0.017	0.326 ± 0.032	0.445 ± 0.039
10^{-4}	3.683 ± 0.411	0.292 ± 0.032	0.186 ± 0.021	0.359 ± 0.034	0.450 ± 0.039
10^{-5}	1.179 ± 0.127	0.296 ± 0.022	0.312 ± 0.030	0.428 ± 0.039	0.460 ± 0.040

Table A–3: Parameter sensitivity results of SAGA in Random MDP.

σ_ω \ σ_θ	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
Average of last 5 percent of MSPBE					
1	NaN	NaN	NaN	NaN	NaN
10^{-1}	2.9e+12	0.007 ± 0.002	0.392 ± 0.036	0.463 ± 0.040	0.464 ± 0.040
10^{-2}	0.039 ± 0.008	0.019 ± 0.004	0.399 ± 0.036	0.463 ± 0.040	0.464 ± 0.040
10^{-3}	0.203 ± 0.022	0.246 ± 0.030	0.459 ± 0.040	0.464 ± 0.040	0.464 ± 0.040
10^{-4}	0.248 ± 0.030	0.457 ± 0.040	0.464 ± 0.040	0.464 ± 0.040	0.464 ± 0.040
10^{-5}	0.458 ± 0.040	0.464 ± 0.040	0.464 ± 0.040	0.464 ± 0.040	0.464 ± 0.040
AUC of MSPBE					
1	NaN	NaN	NaN	NaN	NaN
10^{-1}	2.0e+11	0.067 ± 0.008	0.427 ± 0.038	0.464 ± 0.040	0.464 ± 0.040
10^{-2}	0.118 ± 0.016	0.112 ± 0.014	0.433 ± 0.038	0.464 ± 0.040	0.464 ± 0.040
10^{-3}	0.237 ± 0.021	0.360 ± 0.034	0.462 ± 0.040	0.464 ± 0.040	0.464 ± 0.040
10^{-4}	0.360 ± 0.034	0.461 ± 0.040	0.464 ± 0.040	0.464 ± 0.040	0.464 ± 0.040
10^{-5}	0.461 ± 0.040	0.464 ± 0.040	0.464 ± 0.040	0.464 ± 0.040	0.464 ± 0.040

A.2 Parameter sensitivity results on Mountain Car environment

All methods were run 100 passes of the data set. $SVRG(1*n)$ means that the number of inner loop iterations is set to n.

Table A–4: Average of last 5 percent of MSPBE in Mountain Car.

σ_ω \ σ_θ	10	1	0.1	0.01
GTD2				
10	NaN	NaN	NaN	6.8e+292
1	1.013 ± 0.151	0.122 ± 0.006	0.343 ± 0.001	0.483 ± 0.000
0.1	0.529 ± 0.057	0.173 ± 0.011	0.347 ± 0.001	0.483 ± 0.000
0.01	2.618 ± 0.204	0.738 ± 0.069	0.388 ± 0.008	0.484 ± 0.000
SVRG $(1 * n)$				
10	NaN	NaN	4.7e+211	4.4e+232
1	NaN	0.117 ± 0.000	0.408 ± 0.000	0.493 ± 0.000
0.1	NaN	0.117 ± 0.000	0.408 ± 0.000	0.493 ± 0.000
0.01	3.9e+81	0.128 ± 0.002	0.410 ± 0.000	0.493 ± 0.000
SVRG $(2 * n)$				
10	NaN	NaN	1.7e+218	2.2e+242
1	NaN	0.089 ± 0.000	0.381 ± 0.000	0.490 ± 0.000
0.1	NaN	0.089 ± 0.000	0.381 ± 0.000	0.490 ± 0.000
0.01	6.3e+98	0.102 ± 0.002	0.383 ± 0.000	0.490 ± 0.000
SAGA				
10	NaN	NaN	NaN	NaN
1	NaN	0.058 ± 0.000	0.483 ± 0.000	0.506 ± 0.001
0.1	NaN	0.064 ± 0.001	0.483 ± 0.000	0.506 ± 0.001
0.01	4.8e+33	15.0	0.622 ± 0.023	0.508 ± 0.001
SCSG 0.1				
10	NaN	NaN	NaN	NaN
1	NaN	0.175 ± 0.013	0.415 ± 0.002	0.494 ± 0.001
0.1	NaN	0.400 ± 0.028	0.421 ± 0.003	0.494 ± 0.001
0.01	3.6e+107	4.619 ± 1.086	0.497 ± 0.014	0.496 ± 0.001

Table A–5: Area under curve (AUC) of MSPBE in Mountain Car.

σ_ω \ σ_θ	10	1	0.1	0.01
		GTD2		
10	NaN	NaN	NaN	3.4e+291
1	1.659 ± 0.081	0.240 ± 0.002	0.418 ± 0.000	0.494 ± 0.000
0.1	1.473 ± 0.045	0.372 ± 0.003	0.423 ± 0.001	0.494 ± 0.000
0.01	9.8	1.636 ± 0.046	0.468 ± 0.001	0.495 ± 0.000
		SVRG $(1 * n)$		
10	NaN	NaN	2.3e+210	2.2e+231
1	NaN	0.254 ± 0.004	0.453 ± 0.000	0.499 ± 0.000
0.1	NaN	0.308 ± 0.018	0.454 ± 0.000	0.499 ± 0.000
0.01	1.9e+80	0.531 ± 0.076	0.458 ± 0.001	0.499 ± 0.000
		SVRG $(2 * n)$		
10	NaN	NaN	8.4e+216	1.1e+241
1	NaN	0.228 ± 0.005	0.439 ± 0.000	0.497 ± 0.000
0.1	NaN	0.293 ± 0.020	0.439 ± 0.000	0.497 ± 0.000
0.01	3.1e+97	0.911 ± 0.158	0.444 ± 0.001	0.497 ± 0.000
		SAGA		
10	NaN	NaN	NaN	NaN
1	NaN	0.169 ± 0.001	0.493 ± 0.000	0.507 ± 0.001
0.1	NaN	0.358 ± 0.032	0.494 ± 0.000	0.507 ± 0.001
0.01	2.5e+32	17.7	0.657 ± 0.031	0.509 ± 0.001
		SCSG 0.1		
10	NaN	NaN	NaN	NaN
1	NaN	0.304 ± 0.003	0.460 ± 0.001	0.500 ± 0.000
0.1	NaN	0.626 ± 0.021	0.467 ± 0.001	0.500 ± 0.000
0.01	1.8e+106	5.9	0.547 ± 0.006	0.501 ± 0.001

A.3 Parameter sensitivity results on Cart Pole environment

All methods were run 100 passes of the data set. $SVRG(1*n)$ means that the number of inner loop iterations is set to n.

Table A–6: Average of last 5 percent of MSPBE in Cart Pole.

σ_ω \ σ_θ	10	1	0.1	0.01
		GTD2		
50	NaN	NaN	NaN	NaN
10	0.028 ± 0.013	0.002 ± 0.000	0.004 ± 0.000	0.004 ± 0.000
1	0.017 ± 0.005	0.002 ± 0.000	0.004 ± 0.000	0.004 ± 0.000
0.1	0.036 ± 0.017	0.002 ± 0.000	0.003 ± 0.000	0.004 ± 0.000
		$SVRG\ (1*n)$		
50	NaN	NaN	NaN	NaN
10	1.9e+59	0.002 ± 0.000	0.004 ± 0.000	0.004 ± 0.000
1	2.6e+15	0.002 ± 0.000	0.004 ± 0.000	0.004 ± 0.000
0.1	1.9e+11	0.002 ± 0.000	0.004 ± 0.000	0.004 ± 0.000
		$SVRG\ (2*n)$		
50	NaN	NaN	NaN	NaN
10	3.8e+39	0.002 ± 0.000	0.004 ± 0.000	0.004 ± 0.000
1	9.9e+11	0.002 ± 0.000	0.004 ± 0.000	0.004 ± 0.000
0.1	1.1e+09	0.002 ± 0.000	0.004 ± 0.000	0.004 ± 0.000
		SAGA		
10	NaN	NaN	NaN	NaN
1	NaN	0.002 ± 0.000	0.004 ± 0.000	0.004 ± 0.000
0.1	5.7e+302	0.002 ± 0.000	0.004 ± 0.000	0.004 ± 0.000
0.01	1.3e+08	10.1	0.106 ± 0.032	0.005 ± 0.000
		SCSG 0.1		
50	NaN	NaN	NaN	NaN
10	NaN	0.002 ± 0.000	0.004 ± 0.000	0.004 ± 0.000
1	3.8e+101	0.002 ± 0.000	0.004 ± 0.000	0.004 ± 0.000
0.1	1.2e+15	0.002 ± 0.000	0.004 ± 0.000	0.004 ± 0.000

Table A–7: Area under curve (AUC) of MSPBE in Cart Pole.

σ_ω \ σ_θ	10	1	0.1	0.01
		GTD2		
50	NaN	NaN	NaN	NaN
10	0.021 ± 0.005	0.003 ± 0.000	0.004 ± 0.000	0.004 ± 0.000
1	0.014 ± 0.001	0.003 ± 0.000	0.004 ± 0.000	0.004 ± 0.000
0.1	0.023 ± 0.003	0.003 ± 0.000	0.004 ± 0.000	0.004 ± 0.000
		SVRG $(1 * n)$		
50	NaN	NaN	NaN	NaN
10	9.5e+57	0.003 ± 0.000	0.004 ± 0.000	0.004 ± 0.000
1	1.7e+14	0.003 ± 0.000	0.004 ± 0.000	0.004 ± 0.000
0.1	1.4e+10	0.004 ± 0.001	0.004 ± 0.000	0.004 ± 0.000
		SVRG $(2 * n)$		
50	NaN	NaN	NaN	NaN
10	2.1e+38	0.003 ± 0.000	0.004 ± 0.000	0.004 ± 0.000
1	5.4e+10	0.003 ± 0.000	0.004 ± 0.000	0.004 ± 0.000
0.1	8.3e+07	0.004 ± 0.001	0.004 ± 0.000	0.004 ± 0.000
		SAGA		
10	NaN	NaN	NaN	NaN
1	NaN	0.002 ± 0.000	0.004 ± 0.000	0.005 ± 0.000
0.1	2.8e+301	0.009 ± 0.003	0.004 ± 0.000	0.005 ± 0.000
0.01	1.0e+07	15.9	0.062 ± 0.022	0.005 ± 0.001
		SCSG 0.1		
50	NaN	NaN	NaN	NaN
10	NaN	0.003 ± 0.000	0.004 ± 0.000	0.004 ± 0.000
1	1.9e+100	0.003 ± 0.000	0.004 ± 0.000	0.004 ± 0.000
0.1	6.9e+13	0.003 ± 0.000	0.004 ± 0.000	0.004 ± 0.000

A.4 Parameter sensitivity results on Acrobot environment

All methods were run for 100 passes through the data set. $SVRG(1 * n)$ means that the number of inner loop iterations is set to n.

Table A–8: Average of last 5 percent of MSPBE in Acrobot.

σ_ω \ σ_θ	10	1	0.1	0.01
		GTD2		
10	NaN	NaN	NaN	NaN
1	NaN	0.00035 ± 0.00006	0.00104 ± 0.00002	0.00396 ± 0.00002
0.1	0.00455 ± 0.00204	0.00071 ± 0.00012	0.00117 ± 0.00002	0.00412 ± 0.00004
0.01	0.06489 ± 0.01786	0.00596 ± 0.00154	0.00236 ± 0.00005	0.00456 ± 0.00007
		SVRG $(1 * n)$		
10	NaN	NaN	NaN	NaN
1	NaN	0.21671 ± 0.67233	0.00159 ± 0.00002	0.01391 ± 0.00011
0.1	NaN	0.00049 ± 0.00004	0.00185 ± 0.00002	0.01530 ± 0.00014
0.01	NaN	0.00403 ± 0.00110	0.00429 ± 0.00011	0.01806 ± 0.00020
		SVRG $(2 * n)$		
10	NaN	NaN	NaN	NaN
1	NaN	0.00314 ± 0.00521	0.00132 ± 0.00001	0.00792 ± 0.00006
0.1	NaN	0.00036 ± 0.00003	0.00153 ± 0.00002	0.00861 ± 0.00008
0.01	NaN	0.01683 ± 0.03243	0.00312 ± 0.00008	0.01021 ± 0.00014
		SAGA		
10	NaN	NaN	NaN	NaN
1	NaN	1.4e+10	0.00398 ± 0.00003	0.41448 ± 0.00601
0.1	NaN	0.00278 ± 0.00072	0.00462 ± 0.00006	0.41424 ± 0.00609
0.01	NaN	34.3	0.13465 ± 0.01384	0.41846 ± 0.00641
		SCSG 0.1		
10	NaN	NaN	NaN	NaN
1	NaN	0.00092 ± 0.00034	0.00163 ± 0.00005	0.01351 ± 0.00051
0.1	NaN	0.00134 ± 0.00029	0.00192 ± 0.00009	0.01533 ± 0.00042
0.01	NaN	0.00870 ± 0.00286	0.00454 ± 0.00023	0.01851 ± 0.00099

Table A–9: Area under curve (AUC) of MSPBE in Acrobot.

σ_ω \ σ_θ	10	1	0.1	0.01
		GTD2		
10	NaN	NaN	NaN	NaN
1	NaN	0.00656 ± 0.00013	0.00986 ± 0.00014	0.04394 ± 0.00048
0.1	0.01847 ± 0.00185	0.00757 ± 0.00019	0.01038 ± 0.00017	0.04514 ± 0.00052
0.01	0.22420 ± 0.04291	0.02091 ± 0.00112	0.01512 ± 0.00027	0.04799 ± 0.00055
		SVRG $(1 * n)$		
10	NaN	NaN	NaN	NaN
1	NaN	0.95628 ± 2.24547	0.01827 ± 0.00037	0.08068 ± 0.00093
0.1	NaN	0.07017 ± 0.05761	0.01912 ± 0.00048	0.08261 ± 0.00092
0.01	NaN	0.33818 ± 0.21250	0.02757 ± 0.00095	0.08690 ± 0.00093
		SVRG $(2 * n)$		
10	NaN	NaN	NaN	NaN
1	NaN	0.15919 ± 0.22831	0.01690 ± 0.00076	0.06467 ± 0.00072
0.1	NaN	0.08069 ± 0.03854	0.01742 ± 0.00033	0.06636 ± 0.00073
0.01	NaN	1.11509 ± 1.24275	0.02416 ± 0.00099	0.07010 ± 0.00077
		SAGA		
10	NaN	NaN	NaN	NaN
1	NaN	9.1e+08	0.04931 ± 0.00058	0.47627 ± 0.00758
0.1	NaN	0.50892 ± 0.24586	0.05331 ± 0.00058	0.47644 ± 0.00768
0.01	NaN	133.8	0.34266 ± 0.02498	0.48384 ± 0.00793
		SCSG 0.1		
10	NaN	NaN	NaN	NaN
1	NaN	0.02636 ± 0.01780	0.01391 ± 0.00096	0.07688 ± 0.00245
0.1	NaN	0.02028 ± 0.01265	0.01433 ± 0.00097	0.08049 ± 0.00209
0.01	NaN	0.16170 ± 0.15557	0.02408 ± 0.00125	0.08649 ± 0.00319

APPENDIX B
Parameter Settings

Table B–1: Parameter settings in small-data experiments. This table shows parameter settings in experiments where size of the data set is 20000. For SVRG and Batching SVRG, *inner loop multiplier* is a number that we multiply with n, the size of the data set and it gives the number of inner loop iterations. For SCSG, *batch ratio* is a number that we multiply with n which gives the size of the data set that we use to compute the batch gradient of SCSG. For Batching SVRG, *initial batch ratio* is a number that we multiply with n which gives the initial size of the data set that we use to compute the batch gradient of Batching SVRG. We increase the size of the batch data set by multiplying it with *batch increment ratio*.

Method	*Random MDP*	*Mountain Car*	*Cart Pole*	*Acrobot*
GTD2	$\sigma_\theta = 10^{-4}, \sigma_\omega = 10^{-4}$	$\sigma_\theta = 1, \sigma_\omega = 1$	$\sigma_\theta = 1, \sigma_\omega = 10$	$\sigma_\theta = 1, \sigma_\omega = 1$
SAGA	$\sigma_\theta = 10^{-2}, \sigma_\omega = 10^{-1}$	$\sigma_\theta = 1, \sigma_\omega = 1$	$\sigma_\theta = 1, \sigma_\omega = 1$	$\sigma_\theta = 0.1, \sigma_\omega = 1$
SVRG	$\sigma_\theta = 10^{-3}, \sigma_\omega = 10^{-3}$ inner loop multiplier = 1	$\sigma_\theta = 1, \sigma_\omega = 1$ inner loop multiplier = 2	$\sigma_\theta = 1, \sigma_\omega = 10$ inner loop multiplier = 1	$\sigma_\theta = 0.1, \sigma_\omega = 1$ inner loop multiplier = 2
SCSG	$\sigma_\theta = 10^{-4}, \sigma_\omega = 10^{-4}$ batch ratio = 0.1	$\sigma_\theta = 1, \sigma_\omega = 1$ batch ratio = 0.1	$\sigma_\theta = 1, \sigma_\omega = 10$ batch ratio = 0.1	$\sigma_\theta = 0.1, \sigma_\omega = 1$ batch ratio = 0.1
Batching SVRG	$\sigma_\theta = 10^{-3}, \sigma_\omega = 10^{-3}$ initial batch ratio =0.01 batch increment ratio =1.2 inner loop multiplier = 1	$\sigma_\theta = 1, \sigma_\omega = 1$ initial batch ratio =0.1 batch increment ratio =1.1 inner loop multiplier = 1	$\sigma_\theta = 1, \sigma_\omega = 10$ initial batch ratio =0.1 batch increment ratio =1.1 inner loop multiplier = 1	$\sigma_\theta = 0.1, \sigma_\omega = 1$ initial batch ratio =0.1 batch increment ratio =1.1 inner loop multiplier = 1

Table B–2: Parameter settings in large-data experiments. This table shows parameter settings in experiments where size of the data set is 1,000,000. For SVRG and Batching SVRG, *inner loop multiplier* is a number that we multiply with n, the size of the data set and it gives the number of inner loop iterations. For SCSG, *batch ratio* is a number that we multiply with n which gives the size of the data set that we use to compute the batch gradient of SCSG. For Batching SVRG, *initial batch ratio* is a number that we multiply with n which gives the initial size of the data set that we use to compute the batch gradient of Batching SVRG. We increase the size of the batch data set by multiplying it with *batch increment ratio*.

Method	*Random MDP*	*Mountain Car*	*Cart Pole*	*Acrobot*
GTD2	$\sigma_\theta = 10^{-4}, \sigma_\omega = 10^{-3}$	$\sigma_\theta = 1, \sigma_\omega = 1$	$\sigma_\theta = 1, \sigma_\omega = 10$	$\sigma_\theta = 1, \sigma_\omega = 1$
SCSG	$\sigma_\theta = 10^{-3}, \sigma_\omega = 10^{-2}$ batch ratio = 0.01	$\sigma_\theta = 1, \sigma_\omega = 1$ batch ratio = 0.01	$\sigma_\theta = 1, \sigma_\omega = 10$ batch ratio = 0.05	$\sigma_\theta = 0.1, \sigma_\omega = 1$ batch ratio = 0.05
Batching SVRG	$\sigma_\theta = 10^{-3}, \sigma_\omega = 10^{-2}$ initial batch ratio =0.01 batch increment ratio =1.1 inner loop multiplier = 0.05	$\sigma_\theta = 1, \sigma_\omega = 1$ initial batch ratio =0.001 batch increment ratio =5 inner loop multiplier = 0.2	$\sigma_\theta = 1, \sigma_\omega = 10$ initial batch ratio =0.1 batch increment ratio =1.1 inner loop multiplier = 0.1	$\sigma_\theta = 0.1, \sigma_\omega = 1$ initial batch ratio =0.001 batch increment ratio =5 inner loop multiplier = 0.2

APPENDIX C
Experiment Results in Poor Conditioned Problem Settings

Task	GTD2	SVRG	SAGA	Batching SVRG	SCSG	LSTD
Mountain Car (poor conditioned)	331 ± 185	201 ± 161	$\mathbf{172 \pm 134}$	$\mathbf{171 \pm 134}$	337 ± 180	N/A
Mountain Car	348 ± 181	186 ± 149	$\mathbf{166 \pm 125}$	170 ± 135	276 ± 178	292 ± 187
Cart Pole (poor conditioned)	-25 ± 10	-25 ± 11	-24 ± 10	-28 ± 10	-26 ± 12	N/A
Cart Pole	-163 ± 95	$\mathbf{-280 \pm 88}$	-246 ± 75	$\mathbf{-283 \pm 82}$	-217 ± 100	-183 ± 80
Acrobot (poor conditioned)	388 ± 151	$\mathbf{97 \pm 7}$	101 ± 7	$\mathbf{97 \pm 7}$	126 ± 78	N/A
Acrobot	431 ± 128	$\mathbf{96 \pm 6}$	101 ± 6	$\mathbf{97 \pm 6}$	126 ± 78	176 ± 157
Mountain Car (poor conditioned, large data)	316 ± 189	500 ± 0	500 ± 0	246 ± 171	357 ± 180	N/A
Mountain Car (large data)	264 ± 178	500 ± 0	500 ± 0	349 ± 189	304 ± 182	$\mathbf{116 \pm 6}$
Cart Pole (poor conditioned, large data)	-34 ± 5	-9 ± 0	-9 ± 0	-36 ± 3	-36 ± 5	N/A
Cart Pole (large data)	-198 ± 98	-9 ± 0	-9 ± 0	-267 ± 93	-215 ± 91	$\mathbf{-290 \pm 26}$
Acrobot (poor conditioned, large data)	427 ± 151	500 ± 0	500 ± 0	139 ± 78	$\mathbf{98 \pm 6}$	N/A
Acrobot (large data)	416 ± 137	500 ± 0	500 ± 0	121 ± 77	$\mathbf{99 \pm 6}$	$\mathbf{91 \pm 4}$

Table C–1: All methods' performances in control. First column shows names of all tasks. All values in the table are number of steps (or the negative number of steps) each method takes to reach the terminal state. Small values mean good performances in control. In poor conditioned problem settings, empirical matrix $\hat{A}$ is singular and $\hat{C}$ is not positive definite (i.e. Assumption 1 is not satisfied). LSTD does not have results in poor conditioned problem settings because it requires $\hat{A}$ to be nonsingular. In well conditioned problem settings, we add small identity matrices ($10^{-5} \times I$) to $\hat{A}$ and $\hat{C}$ so that Assumption 1 is satisfied.