# Can Higher-Order Monte Carlo Methods Help Reinforcement Learning?

**Kei Ishikawa**
kishikawa@student.ethz.ch

## Abstract

One of the biggest problems in reinforcement learning is variance reduction. Be it discounted cumulative rewards or its gradient, high variances in their estimation significantly slow down the learning. In this report, we propose the use of two different higher-order Monte Carlo methods—Russian roulette estimation and quasi Monte Carlo methods—for the variance reduction of policy gradient estimation. Theoretically, these estimators have better guarantees of sample efficiency than the vanilla policy gradient based on standard Monte Carlo methods. However, in our experiments, no reduction in the variance of policy gradient was attained for either by the higher-order Monte Carlo methods. Given this result, we discuss several possible hypotheses that explain their failure.

## 1 Introduction

Despite the recent breakthroughs [1, 2, 3, 4], reinforcement learning is notoriously sample inefficient [5]. Among various strategies for improving the sample efficiency [6, 7, 8], variance reduction of the Monte Carlo estimator of learning objective (discounted cumulative reward) and/or its gradient is one of the most effective approaches. Commonly-used variance reduction methods are baselines [9, 10] and predictive variance reduction [11, 12], and they can be understood as the control variate methods for Monte Carlo estimation [13]. However, even though there are other well-established variance reduction methods known by the research community of Monte Carlo methods, almost none of them has been applied to reinforcement learning[1]. Therefore, in this report, we study the applicability of such variance reduction techniques. Especially, we focus on two different types of higher-order Monte Carlo methods, which are known to achieve higher-order convergence of mean squared error (MSE), as the number of samples is increased. As a standard example of Monte Carlo estimation in reinforcement learning, we consider the policy gradient estimation problem, and we derive policy gradient estimator by these techniques and examine its empirical performance by numerical experiments.

The first higher-order method we use is Russian roulette (RR) estimation [14, 15]. Intuitively, the Russian roulette estimation reduces the complexity of generating excessively long trajectories by (random) truncation. Consider an infinite horizon Markov decision process (MDP) with discount factor $\gamma < 1$. The complexity of the vanilla policy gradient estimator (required for a constant variance) increases as we decrease the bias of estimation. For bias size $\varepsilon > 0$, we can truncate the MDP at $T \approx \mathcal{O}(-\log \varepsilon)$ to obtain policy gradient whose bias is controlled as $\|\nabla \sum_{t=T+1}^{\infty} \gamma^t r(s_t, a_t)\|_2 \approx \mathcal{O}(\gamma^T) < \varepsilon$. However, this can be prohibitive if we need an unbiased policy gradient estimator. To avoid the increase of per-trajectory complexity due to increased episode length, the Russian roulette policy gradient estimator truncates the MDP at a random time and estimates the policy gradient from the truncated trajectory in an unbiased manner, while theoretically keeping the expected complexity

[1] To the knowledge of the author, there is at least no work that applies the higher-order Monte Carlo methods we use in this report to reinforcement learning.

Foundations of Reinforcement Learning HS2021, Department of Computer Science, ETH Zürich.

and variance finite. However, in spite of its theoretically attractive properties, we were unable to observe any variance reduction by this estimator in our experiments,

The second higher-order method we use is quasi Monte Carlo (QMC) methods [16, 17, 18]. Simply put, the QMC methods replace the pseudo random sequence used in standard Monte Carlo methods with a low-discrepancy (or quasi Monte Carlo) sequence. Unlike a pseudo random sequence, the low-discrepancy sequence is not independent of one another but is more evenly spread in the sample space. This enables the quasi Monte Carlo average to have $\mathcal{O}(1/N^2)$ MSE instead of $\mathcal{O}(1/N)$ MSE of standard Monte Carlo methods, assuming sufficient smoothness of the Monte Carlo integrand. To leverage the QMC method, we replace the Monte Carlo samples (trajectories) in a mini-batch policy gradient estimator with QMC samples. By sampling trajectories more evenly from the possible state and action space, we aim to reduce the variance. However, similarly to the Russian roulette estimator, this estimator did reduce the variance of the estimation compared to the vanilla policy gradient.

The remaining sections of the report are organized as follows. In Section 2, we review related works on variance reduction and higher-order Monte Carlo methods. Section 3 describes the two proposed policy gradient estimators based on the higher-order Monte Carlo methods. In Section 4, we present the results of numerical experiments and discuss why the proposed methods did not reduce the variance. In Section 5 we conclude the work by discussing whether higher-order Monte Carlo methods can be useful for variance reduction in reinforcement learning.

Finally, we list our contributions of this report for clarity:

- We propose a new policy gradient estimator based on the Russian roulette estimation technique that is theoretically more sample efficient, and its capability in variance reduction is evaluated by numerical experiments.
- We also propose another new policy gradient estimator using quasi Monte Carlo methods and its empirical performance is evaluated similarly.

## 2 Related Works

In this section, we firstly review two of the commonly used variance reduction technique in reinforcement learning: baselines and predictive variance reduction. We then provide reviews on the higher-order Monte Carlo methods we use and their applications in machine learning.

### 2.1 Variance Reduction Methods in Reinforcement Learning

The Baselines are probably the most common approach for variance reduction in policy gradient estimation. They leverage the mean-zero property of score function and construct control variate by multiplying the policy's score function by a quantity that is conditionally independent of the currently sampled action given the current state. This quantity is often called a baseline, and ones dependent on the current state [19, 10, 20] are often used. Recently, baselines that depend not only on the current state but also on the other current actions are actively studied [21, 22, 23, 24, 25].

Recently, an approach called predictive variance reduction [26, 27] has gained popularity. The predictive variance reduction was introduced in the context of stochastic gradient descent, and it was introduced to policy gradient estimation [11, 12]. It reduces the variance of the stochastic gradient by leveraging the recent history of stochastic gradients. Though this is conceptually similar to momentum-based methods, predictive variance reduction methods provide unbiased stochastic gradients. From the low-variance but biased stochastic gradients estimated from the history, they remove the bias in a clever manner so that the variance of the resulting estimator has only a little more variance than the original low-variance biased estimator.

These variance reduction methods are quite orthogonal to our approaches using higher-order Monte Carlo methods. As our approaches do not use control variates, they can easily be combined with these commonly used variance reduction methods.

### 2.2 Higher-Order Monte Carlo Methods

The Russian roulette estimation [14, 15] is a variance reduction method in the Monte Carlo estimation of stochastic infinite series. It randomly truncates the series to keep the length of the series finite and

re-weight each term so that the resulting sum is unbiased in its expectations. Over the last decade, this technique has been actively applied to various tasks in statistics and machine learning to obtain unbiased estimators. Well-known applications include Markov chain Monte Carlo methods [28, 29], particle filtering [30], and variational autoencoders [31].

The quasi Monte Carlo method is another commonly used variance reduction technique [16, 17, 18]. It uses quasi-random samples instead of standard i.i.d. random samples in taking the Monte Carlo average. The quasi-random samples are a set of points in the unit hypercube $[0, 1]^d$ that covers the hypercube more uniformly than random points. This uniformity enables faster convergence in Monte Carlo integration over the hypercube. More details of its theoretical properties are provided in Appendix A. Traditionally, QMC has frequently been applied to the financial simulations of stochastic differential equations [32, 33, 34] and partial differential equations with random coefficients [35]. In statistics and machine learning, it has also found applications in sequential Monte Carlo [36], particle filtering [37], as variational inference [38, 39] and random Fourier features [40].

Though not experimented in this report [2] multi-level Monte Carlo (MLMC) methods are one of the most popular higher-order Monte Carlo methods. It reduces the variance of the Monte Carlo average by considering a hierarchy in the accuracy of simulations. Instead of drawing all samples by the most precise but very expensive random simulations, it combines the results of a large number of cheap but low-accuracy simulations with a very small number of highly accurate simulations and achieves higher-order convergence. Similarly to QMC, MLMC has applications in the simulations of stochastic differential equations [41, 42] and partial differential equations with random coefficients [43]. In statistics and machine learning, MLMC has been applied to Markov Chain Monte Carlo [44], sequential Monte Carlo [45], particle filtering [46], nested expectation [47, 48, 49].

# 3 Proposed Methods

In this section, we start by reviewing the vanilla policy gradient estimator. Then, we describe how to construct an estimator that leverages Russian roulette estimation and quasi Monte Carlo methods.

## 3.1 Vanilla Policy Gradient

Given a trajectory $(s_0, a_0, s_1, a_1, \ldots, s_T, a_T)$ of a MDP with terminal time $T$, the vanilla policy gradient estimator is defined as

$$\nabla_\theta \hat{J}^{\text{Vanilla}} := \sum_{t=1}^{T} \left\{ \left( \sum_{t'=t}^{T} \gamma^{t'} r(s_{t'}, a_{t'}) \right) \nabla_\theta \log \pi_\theta(a_t; s_t) \right\}. \tag{1}$$

Clearly, the complexity of this estimator is $O(T)$, and the variance of this estimator is finite and does not depend on $T$ (as discussed below). This implies that the complexity of policy gradient estimator grows linearly to the terminal time T. Now, let us prove that the variance is finite and does not depend on $T$. Assume a uniform upper bound on the second moment of each term of the policy gradient estimator: for any $t, t' (\geq t)$ and $\theta$, we assume $\mathbb{E} \left| r(s_{t'}, a_{t'}) \frac{\partial}{\partial \theta_i} \log \pi_\theta(a_t; s_t) \right|^2 \leq M_i$ holds, so that

$$\mathbb{E} \left\| r(s_{t'}, a_{t'}) \nabla_\theta \log \pi_\theta(a_t; s_t) \right\|_2^2 \leq M := \sum_i M_i. \tag{2}$$

[2] Though my original intention was to apply MLMC for the policy gradient estimation, we found that the Russian roulette method is much easier to implement in practice while having the same theoretical sample efficiency as MLMC. Therefore, we decided to implement the Russian roulette estimator instead. Though we intended to implement MLMC later, we decided not to implement it, as our experimental results suggested that MLMC and RR are not suited for the reinforcement learning task we used.

Then, we can use Cauchy-Schwarz inequality to obtain the upper bound of the variance of vanilla policy gradient as

$$
\begin{aligned}
&\mathrm{tr}\left[\mathbb{C}\mathrm{ov}[\nabla_\theta \hat{J}^{\text{Vanilla}}]\right] \\
&\quad\leq \mathbb{E}\|\nabla_\theta \hat{J}^{\text{Vanilla}}\|_2^2 \\
&\quad= \sum_{i=1}^{\dim(\theta)} \mathbb{E}\left|\sum_{t=1}^{T}\left\{\left(\sum_{t'=t}^{T}\gamma^{t'} r(s_{t'},a_{t'})\right)\frac{\partial}{\partial\theta_i}\log\pi_\theta(a_t;s_t)\right\}\right|^2 \\
&\quad= \sum_{i}\sum_{t'=1}^{T}\sum_{\tau'=1}^{T}\gamma^{t'+\tau'}\sum_{t=1}^{t'}\sum_{\tau=1}^{\tau'}\mathbb{E}\left[r(s_{t'},a_{t'})\frac{\partial}{\partial\theta_i}\log\pi_\theta(a_t;s_t)\cdot r(s_{\tau'},a_{\tau'})\frac{\partial}{\partial\theta_i}\log\pi_\theta(a_\tau;s_\tau)\right] \\
&\quad\leq \sum_{i}\sum_{t'=1}^{T}\sum_{\tau'=1}^{T}\gamma^{t'+\tau'}\sum_{t=1}^{t'}\sum_{\tau=1}^{\tau'}\sqrt{\mathbb{E}\left|r(s_{t'},a_{t'})\frac{\partial}{\partial\theta_i}\log\pi_\theta(a_t;s_t)\right|^2}\sqrt{\mathbb{E}\left|r(s_{\tau'},a_{\tau'})\frac{\partial}{\partial\theta_i}\log\pi_\theta(a_\tau;s_\tau)\right|^2} \\
&\quad\leq \sum_{i} M_i\left\{\sum_{t'=1}^{T} t'\gamma^{t'}\right\}^2 \leq M\cdot\frac{\gamma^2}{(1-\gamma)^4}. \qquad (3)
\end{aligned}
$$

In the last line, we used $\sum_{k=0}^{n} = r\frac{1-(n+1)r^n+nr^{n+1}}{(1-r)^2} = \mathcal{O}\left(\frac{1}{(1-r)^2}\right)$.

### 3.2 Russian Roulette Policy Gradient

To define a Russian roulette policy gradient, we need to consider a truncated version of the original MDP with random terminal time $\mathcal{T}$. Assuming that $\mathcal{T}$ is independent of the trajectory, we can rewrite our reward as

$$
J = \mathbb{E}_{s,a}\left[\sum_{t=1}^{\infty}\gamma^t r(s_t,a_t)\right] = \mathbb{E}_{\mathcal{T},s,a}\left[\sum_{t=1}^{\infty}\left\{\frac{\mathbb{1}_{t\leq\mathcal{T}}(t,\mathcal{T})}{\mathbb{P}(t\leq\mathcal{T})}\cdot\gamma^t r(s_t,a_t)\right\}\right], \qquad (4)
$$

where we introduced indicator function $\mathbb{1}_{t\leq\mathcal{T}}(\cdot,\cdot)$ which equals one if $t\leq\mathcal{T}$ and zero otherwise. When we set the cumulative distribution of $\mathcal{T}$ as $\mathbb{P}(\mathcal{T}\leq t) := 1-\alpha^t$ with $\gamma<\alpha<1$, we have

$$
J = \mathbb{E}_{\mathcal{T},s,a}\left[\sum_{t=1}^{\mathcal{T}}\left(\frac{\gamma}{\alpha}\right)^t r(s_t,a_t)\right]. \qquad (5)
$$

Therefore, we can construct our unbiased policy gradient estimator from a truncated trajectory $(s_0,a_0,s_1,a_1,\ldots,s_{\mathcal{T}},a_{\mathcal{T}})$ as

$$
\nabla_\theta \hat{J}^{\text{RR}} := \sum_{t=1}^{\mathcal{T}}\left\{\left(\sum_{t'=t}^{\mathcal{T}}\left(\frac{\gamma}{\alpha}\right)^{t'} r(s_{t'},a_{t'})\right)\nabla_\theta\log\pi_\theta(a_t;s_t)\right\}. \qquad (6)
$$

Indeed, the expected complexity and the variance of this estimator are finite. Let us firstly look at the complexity. Conditionally on the sample $\mathcal{T}=T$, the complexity of this estimator is $\mathcal{O}(T)$. Therefore, the expected complexity is:

$$
\mathbb{C}\mathrm{omplexity}[\nabla_\theta\hat{J}^{\text{RR}}] = \mathbb{E}_{\mathcal{T}}[\mathcal{O}(\mathcal{T})] = \mathcal{O}\left(\sum_{T=1}^{\infty}\mathbb{P}(T\leq\mathcal{T})\right) = \mathcal{O}\left(\frac{\alpha}{1-\alpha}\right). \qquad (7)
$$

Similarly to the vanilla policy gradient, we can upper bound the variance of this estimator under assumption (2):

$$
\begin{aligned}
&\mathrm{tr}\left[\mathbb{C}\mathrm{ov}[\nabla_\theta \hat{J}^{\mathrm{RR}}]\right] \\
&\quad \leq \mathbb{E}\|\nabla_\theta \hat{J}^{\mathrm{RR}}\|_2^2 \\
&\quad = \sum_{i=1}^{\dim(\theta)} \mathbb{E}\left|\sum_{t=1}^{\mathcal{T}} \left\{ \left( \sum_{t'=t}^{\mathcal{T}} \left(\frac{\gamma}{\alpha}\right)^{t'} r(s_{t'}, a_{t'}) \right) \frac{\partial}{\partial \theta_i} \log \pi_\theta(a_t; s_t) \right\} \right|^2 \\
&\quad = \sum_i \mathbb{E}_{\mathcal{T}} \left[ \sum_{t'=1}^{\mathcal{T}} \sum_{\tau'=1}^{\mathcal{T}} \left(\frac{\gamma}{\alpha}\right)^{t'+\tau'} \sum_{t=1}^{t'} \sum_{\tau=1}^{\tau'} \mathbb{E}\left[ r(s_{t'}, a_{t'}) \frac{\partial}{\partial \theta_i} \log \pi_\theta(a_t; s_t) \cdot r(s_{\tau'}, a_{\tau'}) \frac{\partial}{\partial \theta_i} \log \pi_\theta(a_\tau; s_\tau) \right] \right] \\
&\quad \leq \sum_i \mathbb{E}_{\mathcal{T}} \left[ \sum_{t'=1}^{\mathcal{T}} \sum_{\tau'=1}^{\mathcal{T}} \left(\frac{\gamma}{\alpha}\right)^{t'+\tau'} \sum_{t=1}^{t'} \sum_{\tau=1}^{\tau'} \sqrt{\mathbb{E}\left| r(s_{t'}, a_{t'}) \frac{\partial}{\partial \theta_i} \log \pi_\theta(a_t; s_t) \right|^2} \sqrt{\mathbb{E}\left| r(s_{\tau'}, a_{\tau'}) \frac{\partial}{\partial \theta_i} \log \pi_\theta(a_\tau; s_\tau) \right|^2} \right] \\
&\quad \leq \sum_i M_i \mathbb{E}_{\mathcal{T}} \left[ \left\{ \sum_{t'=1}^{\mathcal{T}} t' \left(\frac{\gamma}{\alpha}\right)^{t'} \right\}^2 \right] \leq M \cdot \frac{(\gamma/\alpha)^2}{(1-\gamma/\alpha)^4}.
\end{aligned} \tag{8}
$$

Hence, the Russian roulette estimator $\nabla_\theta \hat{J}^{\mathrm{RR}}$ has finite expected complexity and finite variance while being unbiased (under reasonable assumptions). This is in contrast to the vanilla policy gradient estimator, which needs infinite complexity for unbiased estimation in infinite-horizon MDPs.

### 3.3 Quasi Monte Carlo Policy Gradient

The quasi Monte Carlo policy gradient estimator builds on top of the mini-batch version of the vanilla policy gradient estimator. For mini-batch vanilla policy gradient estimator

$$
\nabla_\theta \hat{J}^{\text{mini-batch}} := \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T} \left\{ \left( \sum_{t'=t}^{T} \gamma^{t'} r(s_{t'}^{(n)} a_{t'}^{(n)}) \right) \nabla_\theta \log \pi_\theta(a_t^{(n)}; s_t^{(n)}) \right\}. \tag{9}
$$

we sample $N$ trajectories $\left\{ \left( s_0^{(n)}, a_0^{(n)}, s_1^{(n)}, a_1^{(n)}, \ldots, s_T^{(n)}, a_T^{(n)} \right) \right\}_{n=1}^{N}$ randomly. In quasi-Monte Carlo policy gradient estimation, we sample these $N$ trajectories from a so-called randomized quasi Monte Carlo sequence.

As discussed in the Appendix A, the QMC policy gradient estimator has $O(1/N^2)$ variance for batch size $N$ when the integrand is sufficiently regular. Indeed, this higher convergence rate implies faster convergence of mini-batch stochastic gradient descent. In the optimization of typical smooth non-convex objective, to obtain $\varepsilon$-stationary solution, we need $\mathcal{O}(\varepsilon^{-3/2})$ sample complexity, instead of $\mathcal{O}(\varepsilon^{-2})$ for classical (mini-batch) stochastic gradient descent [3]

One of the biggest challenges of quasi Monte Carlo methods is that it practically achieves higher-order convergence only for Monte Carlo integration with low effective dimension. We say an integrand has effective dimension $d$ if its variation is mostly dependent on the first $d$ dimensions of its domain. In many applications of QMC, domain-specific changes of variables are often used to convert the Monte Carlo integration over the original space to that over another space with a lower effective dimension, in order to obtain the higher-order convergence.

For example, in the case of simulations of stochastic differential equations, discrete-time Brownian motion is sampled using QMC. However, when we naively sample multivariate standard normal distributions by a QMC sequence and generate a Brownian motion as a random walk, earlier dimensions of the QMC sample represent the variations of early steps of the random walks (i.e. Brownian motion). For such a sampling scheme, only integrands whose variations mostly depend on the early steps of the Brownian motion can enjoy the higher-order convergence. Fortunately, this

[3] Here, we need to pick the batch size to be $N = \mathcal{O}(\varepsilon^{-1/2})$ and iterate $\mathcal{O}(\varepsilon^{-1})$ updates of stochastic mini-batch gradient descent. Then by Theorem 1 in [38], the solution is $\varepsilon$-stationary.

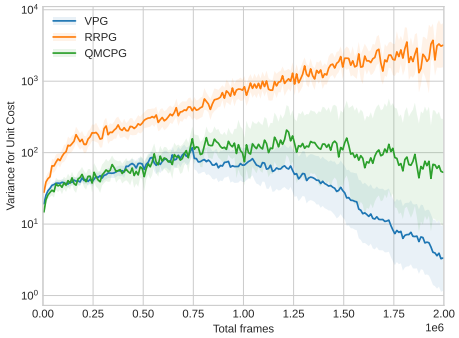



Figure 1: Variance attainable by a unit complexity during the training. Other statistics of variance, are provided in Appendix B

problem can be practically resolved by using so-called PCA construction [50] of a Brownian motion. Let $\Sigma$ be the covariance matrix of the Brownian motion $W = (W_1, \ldots, W_T)$, and $\Sigma = VSV^T$ be its eigenvalue decomposition. By sampling $U$ from a multivariate standard normal distribution using QMC and converting it as $W = VS^{1/2}U$, we can obtain a Brownian Motion most of whose variations depend on the earlier dimensions of QMC sample $U$.

In our experiment, we considered an MDP with ordered discrete action space ($\{-1, 0, +1\}$) with deterministic state transition. To capture the variations of trajectories with the earlier dimensions of the QMC samples, we applied inverse of cumulative distribution function (CDF) of action distribution to transformed increments of the Brownian motion sampled by QMC. More specifically, we first sample discrete-time Brownian motion sampled from QMC using PCA construction. Then we apply the inverse transformation of CDF to the increments (i.e. difference) of the Brownian motion to obtain uniform samples over $[0, 1]$. Then, the sequence of uniform distributions is converted to actions by applying the inverse of action distributions' CDFs. However, there may exist better ways to sample actions (and states) than ours. In most cases, the only way to confirm such a sampling scheme for QMC works is to observe its empirical performance.

## 4 Experiments

In this section, we experimentally compare the performance of the Russian roulette policy gradient (RRPG) and quasi Monte Carlo policy gradient (QMCPG) against that of the vanilla policy gradient (VPG).

### 4.1 Experimental Settings

In our experiments, we used the CartPole-v1 task from the OpenAI Gym environment [51]. This task involves a pole on a cart in a two-dimensional space and the agent must keep the pole upright as long as possible by moving the cart in left/right (or keeping it still). A reward of $+1$ is given for every time step as long as the pole remains no more than $15$ degrees from vertical. Once the angle exceeds $15$ degrees, the episode terminates and the reward stays $0$ afterward. Additionally, we terminate the episode whenever the time step reaches $500$. Discount factor $\gamma$ was set as $\gamma = 0.98$. Hence, the maximum possible undiscounted/discounted cumulative reward in our experiment is $500$ and $50 (= \frac{1}{1-\gamma})$, respectively. The mini-batch size for the VPG and the QMCPG was fixed to 16, while for the QMCPG we picked 48 as the mini-batch size to balance the average complexity per mini-batch during the training. Our implementation is based on autonomous learning library [52] and is available at `https://github.com/kstoneriv3/autonomous-learning-library-with-rrpg`.

### 4.2 Reduction in Variance

Figure 1 shows the variance ($\mathrm{tr}[\mathbb{C}\mathrm{ov}[\nabla \hat{J}]]$) attainable by a unit complexity of each policy gradient algorithms during the training. As the variance can be reduced by a factor of $1/n$ when we can afford

$n$ times more complexity, we define the attainable variance as the product of variance and complexity. The total number of frames sampled during the training is used as the horizontal axis instead of the number of episodes, so that it represents the total complexity consumed in the optimization regardless of the types of algorithms used.

Clearly, the RRPG and the QMCPG did not reduce the variance; rather, they increased the variance compared to the VPG. Unfortunately, this implies that the answer to the question in the title—"Can higher-order Monte Carlo methods help reinforcement learning?"—is likely to be negative. In the following, by taking a closer look at the situation, we conjecture potential reasons for this.

In the experiment, the average of the random truncation times of the RRPG was around 100–150. Therefore, as the complexity of the RRPG estimator of one trajectory is about 3–4 times less than that of the VPG. Therefore, if we increase the terminal time of the MDP by more than 10–100 times, the attainable variance of the VPG (and the QMCPG) is likely to surpass that of the RRPG as the complexity of the RRPG is asymptotically constant with respect to the length of the MDP.

Furthermore, when we consider conditional decomposition of the variance:

$$\mathbb{V}\mathrm{ar}[\nabla \hat{J}^{\mathrm{RR}}] = \mathbb{E}[\mathbb{V}\mathrm{ar}[\nabla \hat{J}^{\mathrm{RR}}|\mathcal{T}]] + \mathbb{V}\mathrm{ar}[\mathbb{E}[\nabla \hat{J}^{\mathrm{RR}}|\mathcal{T}]],$$

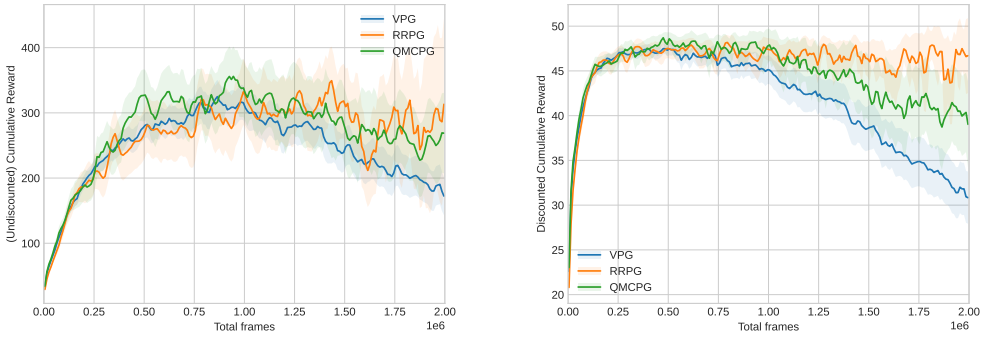
we can explain why the RRPG might have higher variance than the VPG. The first term in the right hand side is close to $\mathbb{V}\mathrm{ar}[\nabla \hat{J}^{\mathrm{Vanilla}}]$ because $\mathbb{V}\mathrm{ar}[\nabla \hat{J}^{\mathrm{RR}}|T] \approx \mathbb{V}\mathrm{ar}[\nabla \hat{J}^{\mathrm{Vanilla}}]$ for sufficiently large $T$, and $\mathcal{T}$ is usually large enough for this approximation. Then, the second term almost equals the excessive variance of the RRPG compared to VPG. As the conditional expectation has order $\mathcal{O}\left(\sum_{k=0}^{\infty} \gamma^k\right)$, the conditional expectation varies by a constant scale as we change $T$. Hence, the second term always incurs additional variance.

Another rationale for the high variance in the RRPG is that the RRPG prefers MDPs where rewards at earlier steps contribute to the majority of the variance of the learning objective (discounted cumulative reward). When later steps make a significant influence on the objective, truncating the trajectory earlier is equivalent to ignoring the most important part of the trajectory. In practical reinforcement learning, it is often the case that later steps tend to become more important as the training progresses. For instance, the agent cannot usually learn the desired behavior in the later steps before learning prerequisite actions in the earlier steps. Speaking of which, the same argument also applies to the MLMC based approach, which we planned to implement at first. As the MLMC samples the later steps less often to gain computational savings, it is likely to suffer from the same issue.

When we look at QMCPG, it also has worse variance than VPG. This is simply because our transformation of the QMC samples from a unit hypercube to the trajectory space did not sufficiently reduce the effective dimension of the problem. In our transformation, we transformed the Brownian increments to a sequence of uniform distribution and used them to sample discrete actions, where the Brownian motion is sampled by QMC using the PCA construction. This transformation might be inefficient at capturing the variations of the distribution of the trajectory (sampled by the policy), because most of the discrete actions can be sampled almost deterministically regardless of the value of uniform samples when the policy is near-optimal. (This is because the optimal action is usually deterministic!) In such cases, the variation in trajectory distribution should come from parts where there exists high uncertainty in the distribution and most of the uniform samples do not contribute at all to the variation of the trajectory. One possible way to capture the uncertain parts of the trajectory distribution would be to entropy as a metric. Nevertheless, at the moment, we do not have any concrete idea about how we can implement such a sampling scheme.

### 4.3 Comparison of Convergence

Figure 2 shows the cumulative reward during the training. Interestingly, the RRPG constantly has high rewards while other methods seem to suffer from some decay in learning objectives. One possible reason for this is that more noise in the gradient regularizes parameters during the optimization. In the context of stochastic gradient descent, it is known that a relatively smaller batch size gives better test performance than the large batch size because it chooses more "flat" minima, which tends to generalize more [53].




(a) (Undiscounted) Cumulative Reward

(b) Discounted Cummulative reward

Figure 2: Cumulative reward during the training.

## 5 Discussion and Conclusion

In this report, we presented the application of two different higher-order Monte Carlo methods for the variance reduction in policy gradient estimation. The first method, the Russian roulette policy gradient reduces the theoretical sample efficiency, by exploiting the property of MDPs that reward at a further time makes an exponentially small contribution to the learning objective (discounted cumulative reward). Though this approach did not improve the sample efficiency in our experiment, it might still be effective in settings where the rewards at a closer time have more influence on the learning objective than the further ones, such as portfolio optimization. In the second approach, quasi Monte Carlo policy gradient, we attempted to reduce the variance of mini-batch by using QMC sequences to sample more evenly from the space of trajectories. Though this method did not provide any performance gain experimentally, this does not completely deny its applicability to reinforcement learning. The QMC can be a great variance reduction method, if we can find a method to capture the variations of learning objective in a low-dimensional space, like in simulations of stochastic differential equations.

To conclude, we found that higher-order Monte Carlo methods do not improve the sample efficiency of relatively simple reinforcement learning tasks. Though these methods might still be useful under some limited conditions, whether such conditions can be met in any practical reinforcement learning settings remains an open question.

## References

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

[2] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

[3] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3389–3396. IEEE, 2017.

[4] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293*, 2018.

[5] Yang Yu. Towards sample efficient reinforcement learning. In *IJCAI*, pages 5739–5743, 2018.

[6] Arthur Guez, David Silver, and Peter Dayan. Efficient bayes-adaptive reinforcement learning using sample-based search. *arXiv preprint arXiv:1205.3109*, 2012.

[7] Ofir Nachum, Shixiang Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. *arXiv preprint arXiv:1805.08296*, 2018.

[8] Jacob Buckman, Danijar Hafner, George Tucker, Eugene Brevdo, and Honglak Lee. Sample-efficient reinforcement learning with stochastic ensemble value expansion. *arXiv preprint arXiv:1807.01675*, 2018.

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

[10] Lex Weaver and Nigel Tao. The optimal reward baseline for gradient-based reinforcement learning. *arXiv preprint arXiv:1301.2315*, 2013.

[11] Matteo Papini, Damiano Binaghi, Giuseppe Canonaco, Matteo Pirotta, and Marcello Restelli. Stochastic variance-reduced policy gradient. In *International conference on machine learning*, pages 4026–4035. PMLR, 2018.

[12] Pan Xu, Felicia Gao, and Quanquan Gu. Sample efficient policy gradient methods with recursive variance reduction. *arXiv preprint arXiv:1909.08610*, 2019.

[13] Christian P Robert, George Casella, and George Casella. *Monte Carlo statistical methods*, volume 2. Springer, 2004.

[14] Herman Kahn. *Use of different Monte Carlo sampling techniques*. Rand Corporation, 1955.

[15] Don McLeish. A general method for debiasing a monte carlo estimator. *Monte Carlo methods and applications*, 17(4):301–315, 2011.

[16] Harald Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*. CBMS-NSF Regional Conference Series in Applied Mathematics, Series Number 63, SIAM, 1992.

[17] Russel E Caflisch. Monte carlo and quasi-monte carlo methods. *Acta numerica*, 7:1–49, 1998.

[18] Josef Dick, Frances Y Kuo, and Ian H Sloan. High-dimensional integration: The quasi-Monte Carlo way. *Acta Numerica*, 22:133–288, 2013.

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

[20] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[21] Shixiang Gu, Timothy Lillicrap, Zoubin Ghahramani, Richard E Turner, and Sergey Levine. Q-prop: Sample-efficient policy gradient with an off-policy critic. *arXiv preprint arXiv:1611.02247*, 2016.

[22] Hao Liu, Yihao Feng, Yi Mao, Dengyong Zhou, Jian Peng, and Qiang Liu. Action-depedent control variates for policy optimization via stein's identity. *arXiv preprint arXiv:1710.11198*, 2017.

[23] Will Grathwohl, Dami Choi, Yuhuai Wu, Geoffrey Roeder, and David Duvenaud. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. *arXiv preprint arXiv:1711.00123*, 2017.

[24] Cathy Wu, Aravind Rajeswaran, Yan Duan, Vikash Kumar, Alexandre M Bayen, Sham Kakade, Igor Mordatch, and Pieter Abbeel. Variance reduction for policy gradient with action-dependent factorized baselines. *arXiv preprint arXiv:1803.07246*, 2018.

[25] George Tucker, Surya Bhupatiraju, Shixiang Gu, Richard Turner, Zoubin Ghahramani, and Sergey Levine. The mirage of action-dependent baselines in reinforcement learning. In *International conference on machine learning*, pages 5015–5024. PMLR, 2018.

[26] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26:315–323, 2013.

[27] Sashank J Reddi, Ahmed Hefny, Suvrit Sra, Barnabas Poczos, and Alex Smola. Stochastic variance reduction for nonconvex optimization. In *International conference on machine learning*, pages 314–323. PMLR, 2016.

[28] Peter W Glynn and Chang-han Rhee. Exact estimation for markov chain equilibrium expectations. *Journal of Applied Probability*, 51(A):377–389, 2014.

[29] Pierre E Jacob, John O'Leary, and Yves F Atchadé. Unbiased markov chain monte carlo methods with couplings. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 82 (3):543–600, 2020.

[30] Pierre E Jacob, Fredrik Lindsten, and Thomas B Schön. Smoothing with couplings of conditional particle filters. *Journal of the American Statistical Association*, 2019.

[31] Yucen Luo, Alex Beatson, Mohammad Norouzi, Jun Zhu, David Duvenaud, Ryan P Adams, and Ricky TQ Chen. Sumo: Unbiased estimation of log marginal probability for latent variable models. *arXiv preprint arXiv:2004.00353*, 2020.

[32] Spassimir Paskov and Joseph F Traub. Faster valuation of financial derivatives. *Journal of Portfolio Management*, 22:113–123, 1995.

[33] Corwin Joy, Phelim P Boyle, and Ken Seng Tan. Quasi-Monte Carlo methods in numerical finance. *Management Science*, 42(6):926–938, 1996.

[34] Russel E Caflisch, William J Morokoff, and Art B Owen. Valuation of mortgage backed securities using Brownian bridges to reduce effective dimension. *Journal of Computational Finance*, 1:27–46, 1997.

[35] Frances Y Kuo, Christoph Schwab, and Ian H Sloan. Quasi-Monte Carlo finite element methods for a class of elliptic partial differential equations with random coefficients. *SIAM Journal on Numerical Analysis*, 50(6):3351–3374, 2012.

[36] Mathieu Gerber and Nicolas Chopin. Sequential quasi Monte Carlo. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 77(3):509–579, 2015.

[37] Dong Guo and Xiaodong Wang. Quasi-Monte Carlo filtering in nonlinear dynamic systems. *IEEE transactions on signal processing*, 54(6):2087–2098, 2006.

[38] Alexander Buchholz, Florian Wenzel, and Stephan Mandt. Quasi-monte carlo variational inference. In *International Conference on Machine Learning*, pages 668–677. PMLR, 2018.

[39] Sifan Liu and Art B Owen. Quasi-newton quasi-monte carlo for variational bayes. *arXiv preprint arXiv:2104.02865*, 2021.

[40] Jiyan Yang, Vikas Sindhwani, Haim Avron, and Michael Mahoney. Quasi-monte carlo feature maps for shift-invariant kernels. In *International Conference on Machine Learning*, pages 485–493. PMLR, 2014.

[41] Michael B Giles. Multilevel Monte Carlo path simulation. *Operations research*, 56(3):607–617, 2008.

[42] Mike Giles. Improved multilevel Monte Carlo convergence using the Milstein scheme. In *Monte Carlo and Quasi-Monte Carlo Methods 2006*, pages 343–358. Springer, 2008.

[43] K Andrew Cliffe, Mike B Giles, Robert Scheichl, and Aretha L Teckentrup. Multilevel Monte Carlo methods and applications to elliptic PDEs with random coefficients. *Computing and Visualization in Science*, 14(1):3, 2011.

[44] Tim J Dodwell, Christian Ketelsen, Robert Scheichl, and Aretha L Teckentrup. A hierarchical multilevel Markov Chain Monte Carlo algorithm with applications to uncertainty quantification in subsurface flow. *SIAM/ASA Journal on Uncertainty Quantification*, 3(1):1075–1108, 2015.

[45] Alexandros Beskos, Ajay Jasra, Kody Law, Raul Tempone, and Yan Zhou. Multilevel sequential Monte Carlo samplers. *Stochastic Processes and their Applications*, 127(5):1417–1440, 2017.

[46] Ajay Jasra, Kengo Kamatani, Kody JH Law, and Yan Zhou. Multilevel particle filters. *SIAM Journal on Numerical Analysis*, 55(6):3068–3096, 2017.

[47] Micheal B Giles and Takashi Goda. Decision-making under uncertainty: using MLMC for efficient estimation of EVPPI. *Statistics and Computing*, 29(4):739–751, 2019.

[48] Kei Ishikawa and Takashi Goda. Efficient debiased evidence estimation by multilevel monte carlo sampling. In *Uncertainty in Artificial Intelligence*, pages 34–43. PMLR, 2021.

[49] Yifan Hu, Xin Chen, and Niao He. On the bias-variance-cost tradeoff of stochastic optimization. *Advances in Neural Information Processing Systems*, 34, 2021.

[50] Peter A Acworth, Mark Broadie, and Paul Glasserman. A comparison of some monte carlo and quasi monte carlo techniques for option pricing. In *Monte Carlo and Quasi-Monte Carlo Methods 1996*, pages 1–18. Springer, 1998.

[51] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[52] Chris Nota. The autonomous learning library. `https://github.com/cpnota/autonomous-learning-library`, 2020.

[53] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[54] Christiane Lemieux. *Monte Carlo and quasi-Monte Carlo sampling*. Springer Science & Business Media, 2009.

[55] Gunther Leobacher and Friedrich Pillichshammer. *Introduction to quasi-Monte Carlo integration and applications*. Springer, 2014.

[56] Art B Owen. Scrambled net variance for integrals of smooth functions. *Annals of statistics*, 25 (4):1541–1562, 1997.

[57] Murray Rosenblatt. Remarks on a multivariate transformation. *The annals of mathematical statistics*, 23(3):470–472, 1952.

# A A Review on Quasi Monte Carlo Methods

Here, we review the theoretical properties of quasi Monte Carlo methods. The explanation here is heavily based on Lemieux [54] and Buchholz et al. [38].

## A.1 Koksma-Hlawka inequality

The QMC methods are a class of numerical integration methods that use low-discrepancy sequences over the hypercube $[0,1]^d$. The following inequality, called the Koksma-Hlawka inequality, represents the essential ideas of QMC:

$$\left| \int_{[0,1]^d} \psi(u) \mathrm{d}u - \frac{1}{|P|} \sum_{u \in P} \psi(u) \right| \leq V_{HK}(\psi) D^*(P). \tag{10}$$

This inequality provides upper bound on the errors in approximation of integration $\int_{[0,1]^d} \psi(u) \mathrm{d}u$ by the average of $\psi$ over point set $P \subset [0,1]^d$. The quantities on the right, $V_{HK}(\psi)$ and $D^*(P)$ are called the Hardy-Krause total variation and the star discrepancy, respectively. As the Hardy-Kruase total variation does not depend on $P$, by choosing point set $P$ so that $D^*(P) = \mathcal{O}(1/|P|^\beta)$ where $\beta > 1$, we can obtain higher-order convergence than the standard Monte Carlo methods.

The definition of the Hardy-Krause total variation for general class of function is a bit involved, but when $\psi$ has continuous partial derivatives up to order $d$, it equals to $\int_{[0,1]^d} \left| \frac{\partial^d \psi}{\partial u_1 \cdots \partial u_d} \right| \mathrm{d}u$.

The star discrepancy is defined as follows: First, for a point $y = (y_1, \ldots, y_d) \in [0,1]^d$, we define

$$D(P, y) := \frac{1}{|P|} \sum_{u \in P} \mathbb{1}_{u \in [0, y_1] \times \cdots \times [0, y_d]} - \prod_{j=1}^{d} y_j,$$

where $\mathbb{1}_\bullet$ denotes the indicator function of $\bullet$. The first term denotes the relative number of points that fall in the rectangular $[0, y_1] \times \cdots \times [0, y_d]$, while the second term denotes the volume of the rectangular. Hence $D(P, y)$ denotes a *local discrepancy* between the empirical distribution determined by $P$ and the (ideal) uniform distribution. By taking a norm (typically, $L_p$-norm with some $p \in [1, \infty]$) of $D(P, y)$ with respect to $y$, we can measure a *global discrepancy* of a point set $P$ from the uniform distribution. Especially, when $p = \infty$, the discrepancy $D^*(P) := \sup_{y \in [0,1]^d} D(P, y)$ is called star discrepancy of points $P$.

The low discrepancy of the QMC sequence can be intuitively understood when we visualize Monte Carlo samples and QMC samples in a two-dimensional unit square. Figure 3 shows the random, QMC, and RQMC sequences. Here, we can observe that QMC and RQMC are more evenly distributed on the unit squares.

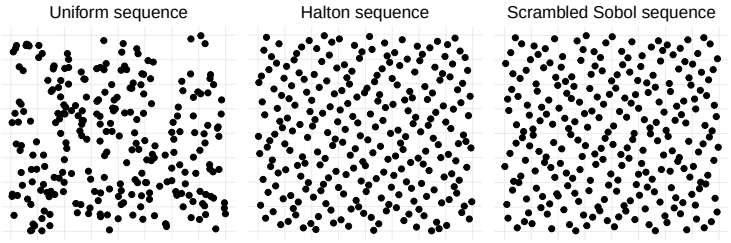



Figure 3: The first 256 points from random (left), QMC (center) and RQMC (right) sequences on unit squares $[0,1]^2$. The latter two tend to cover the target space more evenly. The figures are taken from [38]

.

## A.2 Rates of QMC and RQMC

Indeed, there exists multiple (deterministic) QMC sequence with $D^*(P) = \mathcal{O}((\log |P|)^{2d-2}/|P|^2)$ [55], such as Sobol and Halton sequence. Additionally, a randomization method called scrambling [56] of often applied to the QMC sequence to obtain randomized

quasi Monte Carlo (RQMC) sequences that produces unbiased estimate. With the scrambling, it is possible to take a RQMC sequence satisfying $\mathbb{V}\mathrm{ar}\left(\frac{1}{|P|}\sum_{u\in P}\psi(u)\right) = \mathcal{O}(1/|P|^2)$ when $\psi$'s partial derivatives up to order $d$ exists and is continuous [36]. Hence, achievable rates for QMC methods can be summarized as in Table 1.

| MC | QMC | RQMC |
|---|---|---|
| $N^{-1}$ | $N^{-2}(\log N)^{2d-2}$ | $N^{-2}$ |

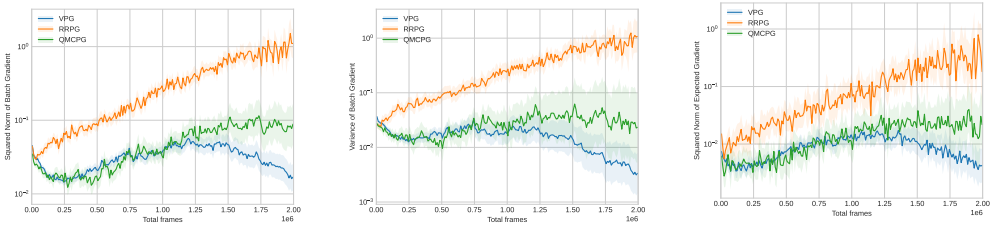
Table 1: Best achievable rates for MC, QMC and RQMC in terms of the MSE of the approximation.

### A.3 Transforming QMC and RQMC sequences

In a general class of Monte Carlo integration, we usually need to draw samples from some target distributions but not from a unit hypercube. In such cases, one can apply the inverse Rosenblatt transformation $\Gamma : u \in [0,1]^d \mapsto z \in \mathbb{R}^d$ such that $\int \psi(\Gamma(u))\mathrm{d}u = \int \psi(z)p(z)\mathrm{d}z$, where $p(z)$ is the respective measure of integration [57, 36]. The inverse Rosenblatt transformation can be understood as the multivariate extension of the inverse CDF transform. For the QMC and RQMC to achieve higher-order convergence, we have to make sure that $\psi \circ \Gamma$ is sufficiently regular.

## B More Statistics of Variance

Here we provide plots of batch variance during the optimization. Clearly, we can observe the same trend as in Figure 1; the RRPG and the QMCPG have more variance than the VPG, and the variance of the RRPG tends to diverge. One interesting observation of the RRPG is that even the gradient norm tends to diverge, which implies that RRPG is not converging to the stationary point while its objective function stays the highest in the later stage of optimization as in Figure 2.




(a) $\mathbb{E}[\|\nabla\hat{J}\|^2]$ (b) $\mathbb{V}\mathrm{ar}[\nabla\hat{J}]$ (c) $\|\nabla J\|^2 \left(= \|\mathbb{E}[\nabla\hat{J}]\|^2\right)$

Figure 4: The squared norm of batch variance (left), variance of batch gradient (center), and the squared norm of the (non-stochastic) gradient (right). Note that they satisfy a relation: $\mathbb{E}[\|\nabla\hat{J}\|^2] = \mathbb{V}\mathrm{ar}[\nabla\hat{J}] + \|\mathbb{E}[\nabla\hat{J}]\|^2$.