Proceedings of the 2025 Winter Simulation Conference
E. Azar, A. Djanatliev, A. Harper, C. Kogler, V. Ramamohan, A. Anagnostou, and S. J. E. Taylor, eds.

FAST MONTE CARLO

Irene Aldridge[1]

[1]Cornell University, ORIE, Financial Engineering, Ithaca, NY, USA

ABSTRACT

This paper proposes an eigenvalue-based small-sample approximation of the celebrated Markov Chain Monte Carlo that delivers an invariant steady-state distribution that is consistent with traditional Monte Carlo methods. The proposed eigenvalue-based methodology reduces the number of paths required for Monte Carlo from as many as 1,000,000 to as few as 10 (depending on the simulation time horizon T), and delivers comparable, distributionally robust results, as measured by the Wasserstein distance. The proposed methodology also produces a significant variance reduction in the steady-state distribution.

1 INTRODUCTION

Monte Carlo simulation is an intuitive computational methodology that estimates the distribution of a given variable. Monte Carlo is flexible and non-parametric, repeatedly sampling various possible evolution paths of the variable instead of imposing rigid modeling specifications. Monte Carlo is ubiquitous in applications, such as asset pricing: see, for example, (Boyle et al. 1997) and (Glasserman 2003), forecasting: (Leith 1974), and, most recently, machine learning optimization: (Painter et al. 2023) and (Henderson, S. G. and S. P. Meyn 2020), among many others. The application of Monte Carlo proved to be particularly effective in multi-agent problems, such as multi-period games: (Cheng 1994) and (Matthew V Macfarlane and Edan Toledo and Donal Byrne and Paul Duckworth and Alexandre Laterre 2024)). Most recently, (Agrawal et al. 2024) use Monte-Carlo to estimate Efficient Influence Functions (EIF) that determine causal relationships among different variables. (Abbassi et al. 2015) applied Monte Carlo to advertising optimization. Monte Carlo is even used to assess the structural stability of buildings (see (Song and Kawai 2023) for a comprehensive survey).

One of the drawbacks of Monte Carlo algorithms is that they take a long time to compute. For example, (Anderson et al. 2018) identify computational complexities for different versions of Monte Carlo implementations ranging from $\Theta(N^{2\alpha}+N)$ to $\Theta(N^{2\alpha-1}(logN)^2+N)$. Different mitigation strategies have been proposed to speed up the computation time, including using advanced hardware by (Rosenthal 2000) and machine-learning-aided estimation by (Lam and Zhang 2023) and (Li, F, H. Chen, J. Lin, A. Gupta, X. Tan, H. Zhao, G. Xu, Y. Nevmyvaka, A. Capponi and H. Lam. 2025).

In this paper, I propose to dramatically speed up the computation of Monte-Carlo using the Perron-Frobenius theorem. Extending the classic Markov Chain Monte Carlo (MCMC) due to (Metropolis et al. 1953a) and (Hastings 1970), the proposed algorithm:

1. incorporates intermediate simulation steps, discarded in classic Monte-Carlo approaches, into the Markov Chain. This part of the process can be calculated in $\mathcal{O}(N)$.
2. applies Perron-Frobenius theorem to find the steady-state distribution of the Markov Chain, that also happens to be the terminal Monte-Carlo distribution. This part of the process can be implemented in constant time $\mathcal{O}(1)$
3. the complete process runs in $\mathcal{O}(N)$, significantly faster than comparable alternatives.

In the classic Markov Chain Monte Carlo (MCMC) framework developed by (Von Neumann 1945), (Metropolis et al. 1953a) and (Hastings 1970), the Markov chain records transitions from the beginning

state to the terminal state of each simulation path: S_0 to S_T. The classic MCMC ignores intermediate steps or subpaths from S_t to S_{t+1} for $0 < t < T-1$. However, the intermediate subpaths are all drawn from the same distribution and, therefore, are valid paths under each Monte Carlo specification. Further, each subpath forms a path toward the terminal state destination. I propose to include the subpaths in the Markov matrix to dramatically increase the number of observations in the Markov Chain. The computational complexity of the resulting process is $\mathcal{O}(N)$, directly proportional to N, the number of paths chosen for the operation. This methodology further develops the Stein approach extended by (Lam and Zhang 2023).

Next, traditional MCMC approaches rely on computationally complex algorithms to find the steady-state terminal distribution of the simulation. I propose to use the Perron-Frobenius theorem to quickly estimate the steady-state distribution as the first eigenvector of the Markov Chain. With recent advances in matrix multiplication, the first eigenvector can be computed in $\mathcal{O}(1)$.

Taken together, the extended Markov chain and the Perron-Frobenius estimation of the steady state result in $\mathcal{O}(N)$ complexity of the Monte Carlo calculation. In addition to the theoretical derivation, I provide simulation results that vividly illustrate the process.

The proposed eigenvalue framework leverages low-rank approximations. Related approaches include (Kozyrskiy, N. and A.-H. Phan 2020) and (Hu, E. J., Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S., L. Wang, and C. Weizhu 2021), who use low-rank decomposition to increase the computational speed of Convolutional Neural Networks (CNNs) and Large Language Models (LLMs), respectively.

This work falls under the umbrella of distributionally robust optimization (DRO). The field of DRO was pioneered by (Scarf 1958). This paper further advances ideas proposed by (Calafiore and Ghaoui 2006), (Bertsimas et al. 2015) and (Bertsimas et al. 2018) who emphasized the importance of optimization over randomization. Specifically, (Bertsimas et al. 2018) showed that small sample inferences can be robustly estimated using *some* statistical test. We follow this approach to show that the proposed Monte Carlo methodology also delivers statistically robust inferences.

I measure convergence using Wasserstein distance, which is identical to the MC convergence metric of (Bellin et al. 1994). Furthermore, traditional Monte Carlo methods converge in $\mathcal{O}(\sqrt{N})$. We show that our methodology converges in $\mathcal{O}(1)$.

Last but not least, the proposed approach significantly reduces the variance of Monte-Carlo result. (Li, F, H. Chen, J. Lin, A. Gupta, X. Tan, H. Zhao, G. Xu, Y. Nevmyvaka, A. Capponi and H. Lam. 2025) document the importance of variance reduction in real-life applications. The proposed methodology delivers both accurate estimates and a considerably smaller variance of the result.

2 RELATED LITERATURE

2.1 Monte-Carlo

The original idea of the Monte-Carlo framework was developed by (Von Neumann 1945), (Metropolis and Ulam 1949) and (Metropolis et al. 1953a) as part of the Manhattan project. Monte-Carlo referred to the mathematical modeling of random movements of particles, and was named after the popular gambling destination of that era. Subsequently, Monte-Carlo became a go-to tool for integration and other computational problems popularized by (Bauer 1958), (Hammersley and Morton 1954). Manufacturing, production and scheduling applications soon followed in (Youle et al. 1959), (Jessop 1956), (Crane et al. 1955), (Miller 1961), (Blumstein 1957), (Rich 1955) and surveyed by (Shubik 1960). (Johnson 2013) notes that Monte-Carlo rapidly expanded to social sciences as well, including (McPhee and Smith 1962) and (Barnett 1962). Since then, Monte-Carlo has been widely used in finding distributionally-stable statistical estimators, like in (D'Agostino and Rosman 1974) and many others. Monte-Carlo found a particularly enthusiastic following in Finance: (Palmer et al. 1994), (Johnson 2002) and (Linn and Tay 2007).

The most recent applications of Monte Carlo fall into the following three, equally important, classes:

- Diffusion
- Agent-based modeling

- Reinforcement Learning

2.1.1 Monte-Carlo in Diffusion

Monte Carlo diffusion extends the standard Brownian motion to find the distribution of future values. For example, in Finance, Monte Carlo diffusion is often used to price options by first simulating the distribution of the underlying variable (e.g., a stock) and then estimating the option payout for each outcome and taking the present value of the payouts to find the fair option price. A sample stock price diffusion model and the option payout calculation shown in equations

$$dS_t = \mu S_t dt + \sigma S_t dW_t \tag{1}$$

$$P_C(S_t) = max(S_t - K, 0) \tag{2}$$

$$C_t = E_\theta[P_c(S_t)] \tag{3}$$

where E_θ is the expected value under the chosen probability measure.

(Giles 2008) proposed Euler discretization which he called MultiLevel Monte Carlo (MLMC). The idea of MLMC is to cut down on computation by generating a few high-accuracy approximations and many low-accuracy approximations and sampling repeatedly from the combined pool of paths. The fastest variation of MLMC is the unbiased MLMC tau-leaping algorithm developed by (Anderson and Higham 2012).

2.1.2 Monte-Carlo in Agent-based Models

Monte Carlo has been used in agent-based models, where simulation is used to determine the likely outcomes of the actions of loosely connected autonomous agents. This methodology estimates the joint likelihood function of many potentially heterogeneous agents with possibly unobserved distributions.

Sequential Monte Carlo (SMC) has been deployed to estimate the densities $p(x_t|y_t,\theta)$ of unknown variables x_t, $t = 1,...,T$ given the observed variables y_t, $t = 1,...,T$, and parameters θ. SMC provides a point-wise approximation of density $p(x_t|y_t,\theta)$ where the closed-form solution may not be available. Due to their point-wise nature, SMC is often referred to as a particle filter method and is popular in the estimation of the joint likelihood function. The seminal works in this area include (Gordon et al. 1993) and (Kitagawa 1996). Since estimation can be based on previously-observed variables, the SMC methodology is also considered to be Bayesian.

In a system with B agents or particles, a random collection of $j = 1,...,B$ particles is sampled from the stationary distribution of the hidden variable x_t. The densities $P(y_t|x_t^{(j)})$ are computed. Next. particles are resampled using normalized weights $P(y_t|x_t^{(j)})/\sum_{j=1}^{B} P(y_t|x_t^{(j)})$. A one-period simulation is conducted on each particle to bring it to $t+1$ realization. The process is repeated for $t = 1,...,T$. Recent applications of SMC include (Amisano and Tristani 2010), (Fernández-Villaverde and Rubio-Ramírez 2007), (Bao et al. 2012), (Pitt et al. 2014), (Blevins 2016) and (Gallant et al. 2018).

The SMC process is computationally-intensive, requiring $\mathcal{O}(B \times T \times (\text{rounds of optimization}))$. The methodology proposed in this paper significantly reduces the computational complexity of the SMC method by reducing the number of optimization rounds required.

2.1.3 Monte-Carlo in Reinforcement Learning

A reinforcement learning algorithm can be represented as a sequential decision process. To do so, (Matthew V Macfarlane and Edan Toledo and Donal Byrne and Paul Duckworth and Alexandre Laterre 2024) use the Markov Decision Process (MDP) framework, a standard in Q-learning. MDP can be described as a tuple $(\mathcal{S},\mathcal{A},\mathcal{T},r,\gamma,\mu)$, where, according to (Matthew V Macfarlane and Edan Toledo and Donal Byrne and Paul Duckworth and Alexandre Laterre 2024):

- $\mathscr{S}$ is the set of all possible states
- $\mathscr{A}$ is the set of actions an agent can take
- $\mathscr{T}$ is the state transition probability function, $\mathscr{T} : \mathscr{S} \times \mathscr{A} \to \mathscr{P}(\mathscr{S})$
- r is the reward function: $r : \mathscr{S} \times \mathscr{A} \to \mathscr{R}$
- γ is the discount factor, $\gamma \in [0,1]$
- μ is the initial state distribution.

The sequential decision process also incorporates a set of agent actions following a policy $\pi : \mathscr{S} \to \mathscr{P}(\mathscr{A})$. At each step t, an agent:

1. chooses action $a_t \in \mathscr{A}$
2. ends up in state s_t
3. receives a reward $r_t = r(s_t, a_t)$

At time $t = 0$, the present value of the agent's reward from step t is $\gamma^t r_t$. Over the lifetime of the process, the present value of the cumulative rewards at time $t = 0$ is then $\sum_{t=0}^{\infty} \gamma^t r_t$. The agent seeks an optimal policy π^* to maximize this amount:

$$\pi^* \in arg\,max_{\pi \in \Pi} E_\pi \left[\sum_{t=0}^{\infty} \gamma^t r_t \right] \tag{4}$$

where Π is a set of all realizable policies.

We also define the value function:

$$V^\pi(s_t) = E_\pi \left[\sum_{t=t}^{\infty} \gamma^t r_t | s_t \right] \tag{5}$$

which is the present value of future rewards when the agent starts out from the state s_t and follows the policy π.

The state-action value function maps a state s_t to the present value of future rewards when first choosing action a_t and then following policy π:

$$Q^\pi(s_t, a_t) = E_\pi \left[\sum_{t=t}^{\infty} \gamma^t r_t | s_t, a_t \right] \tag{6}$$

(Toussaint and Storkey 2006), (Kappen et al. 2012) and others formulate the distribution over possible paths $\tau = (s_0, a_0, s_1, a_1, ..., s_T, a_T)$ for horizon $T < \infty$:

$$p(\tau) = \mu(s_0) \Pi_{t=0}^{T} p(a_t) \mathscr{T}(s_{t+1} | s_t, a_t) \tag{7}$$

where $\mathscr{T}$ is the transition probability matrix defined above and

- μ_0 is the initial state distribution
- a_t is the immediately preceding action

The Monte Carlo process is used to estimate the optimal policy π^*. This is accomplished by:

1. Sampling possible paths from τ
2. Simulating terminal outcomes
3. Computing present values of the outcomes under different specifications (such as probabilities of individual states)
4. Finally, choosing the path with the highest present value.

(Agrawal et al. 2024) use Monte-Carlo reinforcement learning to estimate Efficient Influence Functions (EIF) that determine causal relationships among different variables.

3 PROPOSED METHODOLOGY

Following (Toussaint and Storkey 2006) and (Kappen et al. 2012), we define a path τ as a sequence of state transitions under the Monte-Carlo simulation from the beginning state to the terminal state:

$$\tau = (s_0, s_1, , ..., s_T) \tag{8}$$

In the methodology discussed in this section, we explicitly omit the action space required in reinforcement learning, however, it can be inserted without sacrificing the results.

3.1 Markov Chain Construction

The eigenvalue-decomposition approach for Monte Carlo presented in this paper further optimizes any Monte Carlo methodology to reduce the required number of paths. Our approach is related to the Metropolis-Hastings formulation ((Metropolis et al. 1953b), (Hastings 1970)), but also relies on the Perron-Frobenius theorem ((Frobenius 1912)).

As with (Toussaint and Storkey 2006) and others, each simulation randomly samples a path from a set of all possible paths. Then, following Metropolis-Hastings Markov Chain Monte Carlo (MCMC), using the sample paths, we construct a transition probability matrix.

The proposed Markov Chain construction is different from the Metropolis-Hastings MCMC. The Metropolis-Hastings approach only considers transitions from the first step to the respective terminal step, $s_0 \rightarrow s_T$, effectively discarding all the simulation steps in between. The approach proposed in this paper records every transition step in the Markov Matrix: $s_0 \rightarrow s_1, s_1 \rightarrow s_2, \ldots, s_{t-1} \rightarrow s_t \; \forall t \in \{1, \ldots, T\}$. Although the recorded transitions are much smaller than the start-to-end transitions under Metropolis-Hastings MCMC, when considered in the steady-state framework, both the proposed transitions and Metropolis-Hastings MCMC converge to the same terminal distribution. This result follows directly from the principles of Markov Chain construction. By construction, the proposed Eigen Monte Carlo approach produces a much larger sample of transitions than does Metropolis-Hastings MCMC, requiring a lot fewer paths to draw sound steady-state conclusions.

Our proposed framework, therefore, increases the number of observations in the Markov Chain without increasing the number of simulation paths.

3.2 Computational Improvement of Steady-State Calculations

Next, we further improve the computation speed of the steady-state by deploying the Perron-Frobenius theorem: the first eigenvector of a Markov Chain transition probability matrix serves as the vector of the steady-state probabilities of the Markov Chain matrix. With Perron-Frobenius, we are able to deduce a steady-state distribution of the full Monte Carlo simulation with fewer steps.

To compute the steady-state vector for the transition probability matrix $\mathscr{T}$:

1. Find the largest eigenvalue λ_{max} and the corresponding singular vector v_0 by solving $(A - I_n)v_0 = 0$
2. Divide v_0 by the sum of entries $\sum_i v_0$ to obtain a normalized vector w that sums up to 1.
3. The vector w will automatically have all positive values. This is the steady-state vector for the transition probability matrix $\mathscr{T}$.

It is easy to prove that the largest eigenvalue of a Markov matrix is always 1 ((Seabrook and Wiskott 2023)).

Our key result is to show that the Perron-Frobenius eigenvector v_0 corresponding to $\lambda = 1$ in any Markov matrix is always invariant. That is, regardless of how many observations we have, the vector v_0 retains its distributional properties and represents steady-state distributions of the Markov matrix.

Theorem 1 The steady-state distribution of a Markov Chain Monte Carlo is invariant for any number of paths k which satisfies a normal distribution.

Proof. This result follows directly from the eigenvalue definition applied to the transition probability matrix $\mathscr{T}$:

$$\mathscr{T}v_0 = \lambda_{max}v_0 \tag{9}$$

Since $\lambda_{max} = 1$ for any Markov matrix,

$$\mathscr{T}v_0 = v_0 \tag{10}$$

Suppose now that the Markov transition probability matrix is perturbed:

$$\mathscr{T}' = \mathscr{T} + \varepsilon \tag{11}$$

Then, v_0' is the first eigenvector of $\mathscr{T}'$. Furthermore, since equation (10) holds for all Markov matrices:

$$(\mathscr{T} + \varepsilon)v_0' = v_0' \tag{12}$$

Equation (12) can be further represented as:

$$\mathscr{T}v_0' + \varepsilon v_0' = v_0' \tag{13}$$

Since, equation (10), $\mathscr{T}v_0 = v_0$, holds for any v_0, including v_0', $\varepsilon v_0' \to 0$.

The v_0', therefore, remains invariant from the very few original paths taken by the Monte Carlo simulation. □

Empirically, as discussed in the next section, I find that just a handful of the paths are enough to produce inferences comparable with full-fledged Monte Carlo.

The proposed algorithm works as follows:

Algorithm 1: Fast Monte Carlo

FastMonteCarlo

output: Output of a Monte Carlo Simulation
initialize a Markov chain $M(s_i, s_j) = 0$;
initialize a Monte-Carlo process;
while $i \geq N$ **do**
  foreach *path* $\tau_i = (s_{i0}, s_{i1}, , ..., s_{iT})$ **do**
    Record each intertemporal transition in the Markov chain (e.g., record transition from each state s_t to state s_{t+1} for $t = 1, 2, ...T$: $M(s_t, s_{t+1}) \leftarrow +1 \; \forall \, t$
Run eigenvalue decomposition on the Markov Chain M;
By Perron-Frobenius theorem, the first eigenvector V_0, normalized, represents steady-state probabilities associated with matrix M;
return V_0;

The resulting distribution of the states and their probabilities match those of full Monte Carlo at a fraction of the computational cost. Note that since the Markov Chain incorporates intra-path transitions, the proposed process can run with a fraction of paths planned: $n << N$.

4 COMPUTATIONAL COMPLEXITY

(Andersen et al. 2006) and (Sun et al. 2020) showed that the steady state of a Markov transition probability matrix can be computed in nearly constant time. The result is at least in part due to fast matrix multiplication algorithms, which are themselves an active area of research in reinforcement learning (see (Fawzi et al. 2022)).

As a result, the Monte-Carlo simulation can be distilled in as few as $\mathscr{O}(N)$ steps. The resulting convergence rate of Monte-Carlo is $\mathscr{O}(1)$, independent of the number of steps due to the invariance of the first eigenvalue and the related eigenvector.

5 EXPERIMENTS

5.1 Standard Monte-Carlo

In this section, we reproduce a standard Monte Carlo simulation and compare the speed of the proposed approach. Specifically, we represent the simulated data in a Markov matrix and then find the steady-state distribution using the Perron-Frobenius matrix. Finally, we compare the resulting distribution with a base Monte Carlo distribution obtained by simulating 1,000,000 paths.

In a traditional Monte Carlo simulation, we seek to record the terminal result of each path and create as many paths as possible. In contrast, the proposed method studies the intermediate steps of the path itself. In the proposed model, we examine the entire data path and treat each step as an independent transition that we can subsequently use in the analysis.

In the simulation, we construct a basic binomial tree and compare the performance of the traditional 1,000,000 path Monte Carlo model with the proposed eigenvalue-based model.

Each of the 1,000,000 paths was constructed as follows:

- Each path started with the base value of variable $S_0 = 1$
- A 21×21 Markov Transition Probability matrix $\mathscr{T}$ was initialized. Each row corresponded to 0.1 increment in value. The row indexed by $i = 11$ corresponded to $S = 1$, $i = 12$: $S = 1.1$, etc.
- In each subsequent step $t \in \{1, ...9\}$, a random number $r_t \in [0,1]$ was drawn. Whenever $r_t = 1$, the value of the variable S increased by 10% ($\Delta s_u = 1.1$):

$$S_t|_{r_t=1} = S_{t-1}\Delta s_u \tag{14}$$

  Similarly, whenever $r_t = 0$, the value of the variable S decreased by 10% ($\Delta s_d = 0.9$):

$$S_t|_{r_t=0} = S_{t-1}\Delta s_d \tag{15}$$

- Along each path, all transitions on the path were added to the Markov Transition Probability matrix $\mathscr{T}$. For example, if $S_{t-1} = 1$ and $S_t = 1.1$, then $\mathscr{T}_{11,12}$ was incremented by 1.
- At the end of each path, a copy of the matrix $\mathscr{T}$ was created and normalized. The steady-state probabilities were estimated as the first eigenvector, normalized. The resulting steady-state distribution corresponding to the path $n \in 1,000,000$ was recorded.
- When $n = 1,000,000$ paths were simulated, the average steady-state "true Monte Carlo" distribution was calculated.
- For each value of $n \in 1,000,000$, the Wasserstein Distance was computed between the eigen-distribution (d_e) corresponding to the first n paths and the true Monte-Carlo distribution (d_{MC}). The Wasserstein distance was calculated as:

$$W_p = E[(d_e - d_{MC})^2]^{1/2} \tag{16}$$

Figures 1 and 2 summarize the results. As Figure 1 shows, the Wasserstein distance between the Eigen Monte Carlo and the true Monte-Carlo is minimal even when the Eigen Monte Carlo is estimated only on the first 10 paths! This confirms our theoretical results.

The simulation performed generalizes to Brownian motion and other Monte Carlo-based techniques. Given recent advances in calculating the first singular vector (see, for example, (Andersen et al. 2006) and (Sun et al. 2020)), the proposed Eigen Monte Carlo approach can run in as little as $O(1+\varepsilon)$ computational time. This is a significant improvement over existing methods.

5.2 Diffusion, Step=by-Step

This section compares the traditional and proposed Monte Carlo approaches in options pricing. We seek to estimate a European call option on a hypothetical stock with a mean drift $\mu = 0.002$, volatility $\sigma = 0.01$,

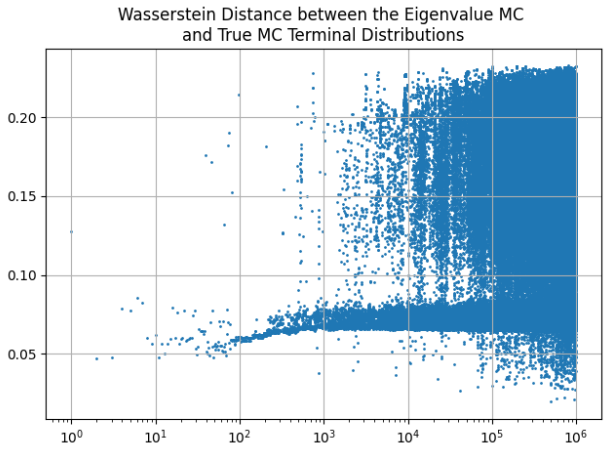

Figure 1: Raw Wasserstein Distance Between the 1,000,000 path Monte-Carlo and the Eigen Monte Carlo determined on the first x paths.

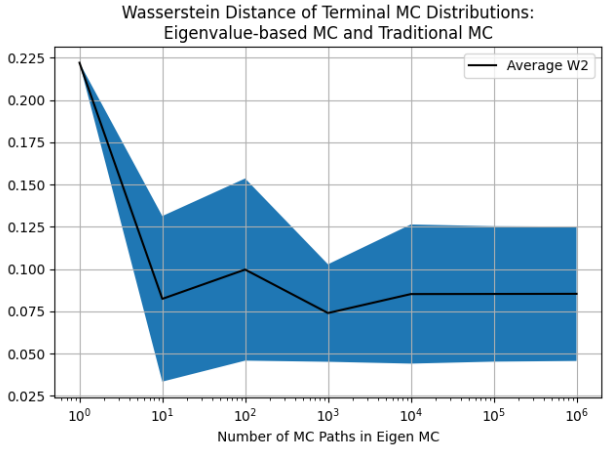

Figure 2: Mean Wasserstein Distance +/- 1 Standard Deviation between the 1,000,000 path Monte-Carlo and the Eigen Monte Carlo determined on the first x paths.

starting price $S_0 = 100$ and option strike price of 110. To keep the simulation tractable and easy to replicate, we simulate $N = 300$ paths over a $T = 30$ time horizon.

We simulate the diffusion process $dS/S = \mu dt + \sigma dW_t$, where W_t is a standard Brownian motion. The resulting lognormal process translates to

$$S_t = S_{t-1} exp\left((\mu + \sigma^2/2)dt + \sigma dW\right) \tag{17}$$

The continuous-value setting of traditional diffusion is a challenge to the MCMC approach, because MCMC enforces discretization of the states. In our experiment, to maintain tractability, we split all possible diffusion values into just 40 states, confining the possible price levels to whole values only, in the range $S = [80, 120]$. If the simulation breached either of the boundary values, the algorithm bucketed the values into the boundary state, another limitation of the experiment. Thus, a simulated price of 80 would be recorded as 80, but so would the price of 79 and any lower price.

To compare the evolution of S under the traditional and proposed MCMC approaches, we construct two distribution:

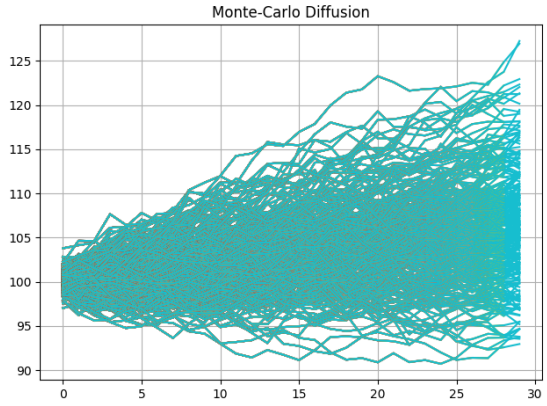

Figure 3: Diffusion

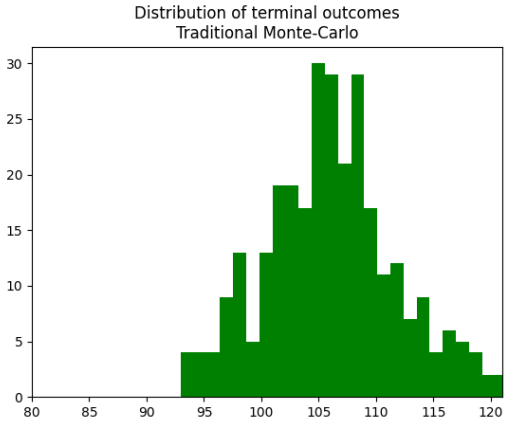

Figure 4: Traditional Diffusion: Terminal State.

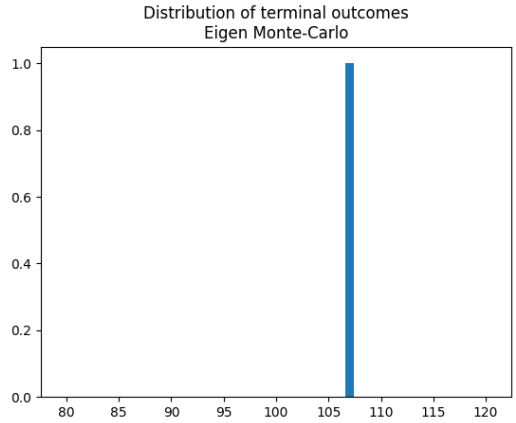

Figure 5: Proposed Eigen-MC: Terminal State.

1. A classic MC distribution of N full path transitions from state S_0 to S_T.
2. The "Eigen Monte Carlo" distribution constructed using the approach proposed in this paper. To develop this distribution, we first construct a Markov Chain that incorporates all intermediate transitions from each state S_i to S_{i+1}, where $i < T$. In this example, the Markov Chain contained just 40 rows and 40 columns. Next, we normalize each row and discard the rows where all the elements were zeros to simplify the estimation process. Finally, we performed the SVD, obtained the first singular vector, and normalized it to find the steady-state distribution.

Figure 3 shows the $N = 300$ simulated diffusion paths of a toy example with mean drift $\mu = 0.002$, volatility $\sigma = 0.01$, starting price $S_0 = 100$ and option strike price 110. Figures 4 and 5 show the terminal distributions of the simulated stock prices for the traditional diffusion Monte-Carlo and the proposed approach. The proposed Eigen Monte Carlo delivers a small-variance estimate.

However, the proposed approach did not always converge in the diffusion setting. The lack of convergence may be driven by excessive discretization. Section 6 considers some aspects of the approach performance driven by the model parameterization.

6 DISCUSSION OF RESULTS

The proposed methodology increases the underlying sample space of the Monte-Carlo simulation, without increasing the computational complexity of the process. Markov Chains effectively document the flow of data among states. The steady-state analysis captures the end points of the information flow, whether the data is flowing through the intermediate states or going directly to a terminal state. By recording all intermediate steps in the simulation, we increase the sample size and deliver more robust inferences with smaller variance.

Much further work is needed to determine the conditions and parameters that determine the simulation outcome:

- The impact of the number of paths N and the number of time steps T required to derive stable inferences using the Eigen Monte Carlo methodology proposed in this paper.
- The effect of the discretization of the Markov matrix: how granular should the states be and what is the impact of approximating the state values in the Markov chain?
- The parameters of the simulation itself: for example, the relationship between the drift and the variance.

7 CONCLUSION

This paper develops and tests a new approach to the classic Monte Carlo simulation. Recording all intra-path transitions in a Markov Transition matrix and then using Eigenvalue decomposition and related Perron-Frobenius theorem, we are able to compute steady-state Monte Carlo inferences with 1) a much smaller variance and 2) in just a few sample paths, down from as many as 1,000,000 iterations. The Eigenvalue-based Monte Carlo can be computed in $O(1)$ computational time.

REFERENCES

Abbassi, Z., A. Bhaskara, and V. Misra. 2015. "Optimizing Display Advertising in Online Social Networks". In *Proceedings of the 24th International Conference on World Wide Web*, 1–11.

Agrawal, R., S. Witty, A. Zane, and E. Bingham. 2024. "Automated Efficient Estimation using Monte Carlo Efficient Influence Functions". In *Advances in Neural Information Processing Systems*, edited by A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, Volume 37, 16102–16132: Curran Associates, Inc.

Amisano, G., and O. Tristani. 2010. "Euro Area Inflation Persistence in an Estimated Nonlinear DSGE Model". *Journal of Economic Dynamics and Control* 34(10):1837–1858.

Andersen, R., F. Chung, and K. Lang. 2006. "Local Graph Partitioning using PageRank Vectors". In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '06, 475–486. USA: IEEE Computer Society https://doi.org/10.1109/FOCS.2006.44.

Anderson, D., and D. Higham. 2012. "Multilevel Monte Carlo for Continuous Markov Chains, with Applications in Biochemical Kinetics.". *SIAM: Multiscale Modeling and Simulation* 10:146–179.

Anderson, D. F., D. J. Higham, and Y. Sun. 2018. "Computational Complexity Analysis for Monte Carlo Approximations of Classically Scaled Population Processes.". *Working paper*.

Bao, Y., C. Chiarella, and B. Kang. 2012. "Particle Filters for Markov Switching Stochastic Volatility Models". *Research Paper* (299).

Barnett, V. D. 1962. "The Monte Carlo Solution of a Competing Species Problem". *Biometrics* 18(1):76–103.

Bauer, W. F. 1958. "The Monte Carlo Method". *Journal of the Society for Industrial and Applied Mathematics* 6(4):438–451 https://doi.org/10.1137/0106028.

Bellin, A., Y. Rubin, and A. Rinaldo. 1994. "Eulerian-Lagrangian Approach for Modeling of Flow and Transport in Heterogeneous Geological Formations". *Water Resources Research* 30(11):2913–2924.

Bertsimas, D., V. Gupta, and N. Kallus. 2018. "Robust sample average approximation". *Mathematical Programming*.

Bertsimas, D., M. Johnson, and N. Kallus. 2015. "The Power of Optimization Over Randomization in Designing Experiments Involving Small Samples". *Operations Research* 63(4):868–876.

Blevins, J. R. 2016. "Sequential Monte Carlo Methods for Estimating Dynamic Microeconomic Models". *Journal of Applied Econometrics* 31(5):773–804.

Blumstein, A. 1957. "A Monte Carlo Analysis of the Ground Controlled Approach System". *Operations Research* 5(3):397–408 https://doi.org/10.1287/opre.5.3.397.

Boyle, P., M. Broadie, and P. Glasserman. 1997. "Monte Carlo methods for security pricing". *Journal of Economic Dynamics and Control* 21(8):1267–1321 https://doi.org/https://doi.org/10.1016/S0165-1889(97)00028-6.

Calafiore, G., and L. Ghaoui. 2006. "On Distributionally Robust Chance-Constrained Linear Programs". *Journal of Optimization Theory and Applications* 130:1–22 https://doi.org/https://doi.org/10.1007/s10957-006-9084-x.

Cheng, R. C. H. 1994. "Selecting Input Models". In *1994 Winter Simulation Conference (WSC)*, 184–191 https://doi.org/10.1109/WSC.1994.717117.

Crane, R. R., F. B. Brown, and R. O. Blanchard. 1955. "An Analysis of a Railroad Classification Yard". *Operations Research* 3(3):262–271 https://doi.org/10.1287/opre.3.3.262.

D'Agostino, R. B., and B. Rosman. 1974. "The Power of Geary's Test of Normality". *Biometrika* 61(1):181–184.

Fawzi, A., M. Balog, A. Huang, T. Hubert, B. Romera-Paredes, M. Barekatain, *et al*. 2022. "Discovering Faster Matrix Multiplication Algorithms with Reinforcement Learning". *Nature* 610:47–53.

Fernández-Villaverde, J., and J. F. Rubio-Ramírez. 2007. "Estimating Macroeconomic Models: A Likelihood Approach". *The Review of Economic Studies* 74(4):1059–1087.

Frobenius, G. 1912. *Über Matrizen aus nicht negativen Elementen*.

Gallant, A. R., H. Hong, and A. Khwaja. 2018. "A Bayesian Approach to Estimation of Dynamic Models with Small and Large Number of Heterogeneous Players and Latent Serially Correlated States". *Journal of Econometrics* 203(1):19–32.

Giles, M. 2008. "Multilevel Monte Carlo Path Simulation". *Operations Research* 56:607–617.

Glasserman, P. 2003. *Monte Carlo Methods in Financial Engineering*. Springer Science + Business Media.

Gordon, N., D. Salmond, and A. Smith. 1993. "Novel Approach to Nonlinear/Non-Gaussian Bayesian State Estimation". *IEE Proceedings F (Radar and Signal Processing)* 140:107–113 https://doi.org/10.1049/ip-f-2.1993.0015.

Hammersley, J. M., and K. W. Morton. 1954. "Poor Man's Monte Carlo". *Journal of the Royal Statistical Society: Series B (Methodological)* 16(1):23–38 https://doi.org/10.1111/j.2517-6161.1954.tb00145.x.

Hastings, W. K. 1970. "Monte Carlo Sampling Methods Using Markov Chains and Their Applications". *Biometrika*.

Henderson, S. G. and S. P. Meyn 2020. "Variance Reduction in Simulation of Multiclass Processing Networks" https://doi.org/10.48550/arXiv.2005.14179.

Hu, E. J., Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S., L. Wang, and C. Weizhu 2021. "LoRA: Low-Rank Adaptation of Large Language Models" https://doi.org/10.48550/arXiv.2106.09685.

Jessop, W. N. 1956. "Monte Carlo Methods and Industrial Problems". *Journal of the Royal Statistical Society. Series C* 5(3):158–165 https://doi.org/10.2307/2985417.

Johnson, P. E. 2002. "Agent-Based Modeling: What I Learned From the Artificial Stock Market". *Social Science Computer Review* 20(2):174–186 https://doi.org/10.1177/089443930202000207.

Johnson, P. E. 2013. "454Monte Carlo Analysis in Academic Research". In *The Oxford Handbook of Quantitative Methods in Psychology, Vol. 1*. Oxford University Press https://doi.org/10.1093/oxfordhb/9780199934874.013.0022.

Kappen, H., V. Gómez, and M. Opper. 2012. "Optimal control as a graphical model inference problem". *Machine Learning* 87:159–182.

Kitagawa, G. 1996. "Monte Carlo Filter and Smoother for Non-Gaussian Nonlinear State Space Models". *Journal of computational and graphical statistics* 5(1):1–25.

Kozyrskiy, N. and A.-H. Phan 2020. "CNN Acceleration by Low-rank Approximation with Quantized Factors" https://doi.org/10.48550/arXiv.2006.08878.

Lam, H., and H. Zhang. 2023. "Doubly Robust Stein-Kernelized Monte Carlo Estimator: Simultaneous Bias-Variance Reduction and Supercanonical Convergence". *J. Mach. Learn. Res.* 24(1).

Leith, C. E. 1974. "Theoretical Skill of Monte Carlo Forecasts". *Monthly Weather Review* 102(6):409 – 418 https://doi.org/10.1175/1520-0493(1974)102<0409:TSOMCF>2.0.CO;2.

Li, F, H. Chen, J. Lin, A. Gupta, X. Tan, H. Zhao, G. Xu, Y. Nevmyvaka, A. Capponi and H. Lam. 2025. "Prediction-Enhanced Monte Carlo: A Machine Learning View on Control Variate" https://doi.org/10.48550/arXiv.2412.11257.

Linn, S. C., and N. S. P. Tay. 2007. "Complexity and the Character of Stock Returns: Empirical Evidence and a Model of Asset Prices Based on Complex Investor Learning". *Management Science* 53(7):1165–1180 https://doi.org/10.1287/mnsc.1060.0622.

Matthew V Macfarlane and Edan Toledo and Donal Byrne and Paul Duckworth and Alexandre Laterre 2024. "SPO: Sequential Monte Carlo Policy Optimisation".

McPhee, W. N., and R. B. Smith. 1962. "A Model for Analyzing Voting Systems". In *Public Opinion and Congressional Elections*, edited by W. N. McPhee and W. A. Glaser, 123–154. New York: Free Press.

Metropolis, N., A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. 1953a. "Equation of State Calculations by Fast Computing Machines". *The Journal of Chemical Physics* 21(6):1087–1092 https://doi.org/10.1063/1.1699114.

Metropolis, N., A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. 1953b. "Equation of State Calculations by Fast Computing Machines.". *The Journal of Chemical Physics*.

Metropolis, N., and S. Ulam. 1949. "The Monte Carlo Method". *Journal of the American Statistical Association* 44:335–341.

Miller, A. J. 1961. "A Queueing Model for Road Traffic Flow.". *Journal of the Royal Statistical Society. Series B,* 23:64–90.

Painter, M., M. Baioumy, N. Hawes, and B. Lacerda. 2023. "Monte Carlo Tree Search with Boltzmann Exploration". In *Advances in Neural Information Processing Systems*, edited by A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Volume 36, 78181–78192: Curran Associates, Inc.

Palmer, R., W. Brian Arthur, J. H. Holland, B. LeBaron, and P. Tayler. 1994. "Artificial Economic Life: a Simple Model of a Stockmarket". *Physica D: Nonlinear Phenomena* 75(1):264–274 https://doi.org/https://doi.org/10.1016/0167-2789(94)90287-9.

Pitt, M. K., S. Malik, and A. Doucet. 2014. "Simulated Likelihood Inference for Stochastic Volatility Models using Continuous Particle Filtering". *Annals of the Institute of Statistical Mathematics* 66:527–552.

Rich, R. P. 1955. "Simulation as an Aid in Model Building". *Journal of the Operations Research Society of America* 3(1):15–19 https://doi.org/10.1287/opre.3.1.15.

Rosenthal, J. S. 2000. "Parallel Computing and Monte Carlo Algorithms.". *Far East Journal of Theoretical Statistics* 4:207–236.

Scarf, H. 1958. *Studies in the Mathematical Theory of Inventory and Production*, Chapter A Min-Max Solution of an Inventory Problem. Stanford University Press.

Seabrook, E., and L. Wiskott. 2023. "A Tutorial on the Spectral Theory of Markov Chains". *Neural Computation* 35(11):1713–1796 https://doi.org/10.1162/neco_a_01611.

Shubik, M. 1960. "Bibliography on Simulation, Gaming, Artificial Intelligence and Allied Topics". *Journal of the American Statistical Association* 55(292):736–751.

Song, C., and R. Kawai. 2023. "Monte Carlo and Variance Reduction Methods for Structural Reliability Analysis: A Comprehensive Review". *Probabilistic Engineering Mechanics* 73:103479 https://doi.org/https://doi.org/10.1016/j.probengmech.2023.103479.

Sun, Z., G. Pedretti, E. Ambrosi, A. Bricalli, and D. Ielmini. 2020. "In-memory Eigenvector Computation in Time O(1)". *Advanced Intelligent Systems*.

Toussaint, M., and A. Storkey. 2006. "Probabilistic inference for solving discrete and continuous state Markov Decision Processes". In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, 945–952. New York, NY, USA: Association for Computing Machinery https://doi.org/10.1145/1143844.1143963.

Von Neumann, J. 1945. "First Draft of a Report on the EDVAC". Technical report, Moore School of Electrical Engineering, University of Pennsylvania.

Youle, P. V., K. D. Tocher, W. N. Jessop, and F. I. Musk. 1959. "Simulation Studies of Industrial Operations". *Royal Statistical Society. Journal. Series A: General* 122(4):484–510 https://doi.org/10.2307/2343076.

AUTHOR BIOGRAPHIES

IRENE ALDRIDGE is a Visiting Professor at Cornell University, ORIE, Financial Engineering. She is a recognized researcher in the applications of Data Science to Finance. Her research interests span advances in financial technology and machine learning/AI, including blockchain-related optimization. Irene is a founder of several Fintech companies, most recently AbleMarkets and AbleBlox, where she served as CEO. In addition to startups, Irene also held several executive roles in banking. She is the author of several books on financial technology, including "High-Frequency Trading" (Wiley, 2013), "Real-Time Risk" (with Steven Krawciw, Wiley, 2017) and "Big Data Science in Finance" (with Marco Avellaneda, 2021). Her email address is irene.aldridge@gmail.com and her website is https://irenealdridge.com/.