Contemporary Mathematics
https://ojs.wiserpub.com/index.php/CM/

Research Article

On the Stochastic Power Algorithms for Estimating the Maximum Eigenvalue of Symmetric Matrices

Silvi-Maria Gurova[1,2], Todor Gurov[1,3*], Aneta Karaivanova[1]

[1] Institute of Information and Communication Technologies-Bulgarian Academy of Sciences (IICT-BAS), 1113, Sofia, Bulgaria
[2] Centre of Excellence in Informatics and Information and Communication Technologies, Sofia, Bulgaria
[3] Centre for Implementation of Scientific Research on Digitisation of the Economy in an Environment of Big Data, Sofia, Bulgaria
E-mail: todor.gurov@iict.bas.bg

Received: 9 July 2025; **Revised:** 21 August 2025; **Accepted:** 9 September 2025

Abstract: Estimating the largest eigenvalue of large symmetric matrices is a critical problem in numerical linear algebra, with wide-ranging applications in science and engineering. Classical iterative methods, such as the Power method, are often computationally expensive or impractical for high-dimensional problems. To address this, we investigate two stochastic methods, Power Monte Carlo (PMC) and Power Quasi-Monte Carlo (PQMC), which integrate Markov chain-based simulations with classical Power iterations to estimate the maximum eigenvalue. These methods are designed to manage both systematic error (from iteration truncation) and stochastic error (from probabilistic sampling), enabling efficient eigenvalue computation even for large matrices. A main contribution of this study is the precise description of the methodology for constructing *Almost Optimal* PMC and PQMC algorithms, which can cover a broad class of symmetric matrices, including both sparse and dense structures. The algorithms employ a special choice of transition density matrices in constructing the Markov chain, which leads to a significantly reduced variance. Numerical experiments demonstrate that these algorithms outperform *classical* PMC/PQMC approaches in accuracy and computational complexity, particularly when using robust random number generators and low-discrepancy sequences. They demonstrate that it is vital to find the right balance between the number of steps in the Markov chain and the number of its simulations. Our findings provide new insights into the design of efficient stochastic algorithms for high-dimensional matrix problems and establish a foundation for further optimization and scalability.

Keywords: *Almost Optimal* PMC and PQMC algorithms, low-discrepency sequences, Markov chain, maximum eigenvalue, stochastic and systematic error

MSC: 65C05, 65C10, 91G60, 65F15

1. Introduction

The eigenvalue problem, particularly the problem for finding the maximum eigenvalue of symmetric matrices, remains essential to many fields, such as quantum mechanics, financial mathematics, data science and machine learning, graph theory and network science, and structural engineering. For instance, in spectral graph theory, the maximum eigenvalue of the adjacency matrix reveals crucial properties of graphs such as expansion, connectivity, and robustness,

DOI: https://doi.org/10.37256/cm.6520257781

influencing algorithms in machine learning and network analysis [1]. In the Machine Learning (ML)-driven link adaptation framework [2], maximum eigenvalues can serve as strong features for classification because they capture the dominant channel gain beyond raw Signal to Noise Ratio (SNR) or Bit Error Rate (BER). In quantum mechanics, for example, the Schrödinger equation is an eigenvalue problem where the Hamiltonian operator determines the energy spectrum of a system. The largest eigenvalue is associated with the highest energy state and carries physical significance in excited-state analysis [3]. A set of exact solutions for Schrödinger-type nonlinear evolution equations in [4] includes optical bright, dark, singular, and periodic solitons. A future spectral analysis (considering maximum eigenvalues of linearized operators around these solutions) would be crucial to evaluate their physical viability in nonlinear optical media. Higher maximum eigenvalues indicate instability and potential modulation growth, while lower or negative ones suggest stable propagation, essential for optical communication. In financial mathematics, particularly in risk modelling and portfolio optimization, the covariance matrix of asset returns is essential. The largest eigenvalue represents the most significant direction of variance in the portfolio-typically linked to systematic market risk and analysing it is crucial for spotting diversification opportunities [5–7]. Similarly, in tensor analysis and high-dimensional data processing, spectral norms (which are tied directly to the largest eigenvalue) govern the behaviour of iterative solvers and regularisation techniques used in signal processing, physics-based simulations, and AI models. Studying the maximum eigenvalue in tensor decompositions is meaningful for understanding energy concentration, robustness, and interpretability of extracted components [8–10]. To find the largest eigenvalue, deterministic methods have traditionally been used. Among these, the Power method is well known for its simplicity and efficiency when the spectral gap is sufficiently large [11, 12]. It involves repeatedly multiplying a random vector by the matrix and normalising it, eventually converging to the dominant eigenvector and its corresponding eigenvalue. However, for high-dimensional or sparse matrices, or when the spectral gap is small, the Power method may converge slowly or become computationally costly. More advanced methods such as QR decomposition (In linear algebra, a QR decomposition or a QR factorisation is a decomposition of a matrix A into a product A = QR of an orthonormal matrix Q and an upper triangular matrix R.) [11] or Arnoldi iteration [13] are computationally intensive and less suitable for parallelization in large-scale applications. Generally, as the matrix size increases, *classical* iterative methods often encounter significant challenges, such as higher computational costs (time, processing power, memory); numerical instabilities due to accumulated rounding errors; and sensitivity to noise in real-world problems.

Stochastic methods like Monte Carlo (MC) and Quasi-Monte Carlo (QMC) overcome most of the previously mentioned limitations. They incorporate randomness into numerical computations to tackle the challenges of large-scale problems. These methods often sacrifice deterministic precision for computational and memory efficiency and scalability, making them suitable for estimating the maximum eigenvalue for high-dimensional symmetric matrices.

The MC method is a computational technique that uses repeated random sampling to estimate numerical results. It is particularly useful for problems that are deterministic in principle but too complex to solve directly. In a typical case, the method constructs a random variable θ with a chosen probability distribution (discrete or continuous) such that its expectation equals a target functional I, i.e. $E(\theta) = I$. This functional I might represent, for example, the value of a definite integral or a physical quantity. Let $\theta_1, \ldots, \theta_N$ be realizations of the random variable θ. Then I can be approximated by the average value $\overline{\theta}_N = \sum_{i=1}^{N} \theta_i$. The error of this estimate depends on the variance σ^2 and the number of samples N. To generate samples, the method uses Pseudorandom Number Generators (PRNGs) [14, 15] and the probability distribution of θ. PRNGs produce independent and uniformly distributed numbers in the interval $(0,\ 1)$ based on a given seed and deterministic formulas. A good PRNG must pass statistical tests, have a long period, be efficient, and support parallel computation [16, 17]. In contrast, the QMC methods replace randomness with low-discrepancy sequences [18], such as Halton [19, 20] or Sobol [21–26] sequences. These sequences spread points more evenly across the sample space and often yield better performance in integration, optimisation, and high-dimensional problems [27, 28]. In practice, they are often scrambled to allow error estimation and to avoid certain artifacts [29–31]. The key difference between MC and QMC lies in their convergence rates. MC methods typically converge at $\mathcal{O}(N^{-\frac{1}{2}})$ [16, 32, 33]. QMC methods can achieve faster rates, around $\mathcal{O}((\log N)^k N^{-1})$ [28, 34–37], where k is the dimension of the problem (e.g., the number of variables in a multidimensional integral).

In linear algebra problems, random variables are often employed to estimate functions involving matrix–vector products, typically constructed via Markov chains, where k denotes the number of transitions (or steps) in the chain. In this context, MC and QMC methods are referred to as iterative MCs and QMCs since they usually require many iterations to achieve reasonable accuracy. These iterative methods introduce two types of errors: systematic error, which arises from truncated iterations and depends on k, and stochastic error, which results from the probabilistic nature of sampling and depends on N. MC/QMC methods for LA problems can be conditionally divided into three categories: those developed for solving Systems of Linear Equations (SLEs) [34, 38–40], those addressing inverse matrix problems [32, 41–43], and those designed for eigenvalue problems [44, 45]. Within the third category, the Power MC method [44, 46] provides an approximation of the largest eigenvalue of a given matrix, while the Resolvent MC method [44, 47] is particularly suitable for estimating the smallest eigenvalue. Achieving a balance between systematic and stochastic errors, together with variance reduction, is essential for the development of efficient PMC and PQMC algorithms.

In this paper, we study the PMC and PQMC methods for estimating the largest eigenvalues of high-dimensional, non-singular, symmetric matrices. By introducing the *Almost Optimal* initial density vector and the *Almost Optimal* density matrix, we refine the construction of random variables using Markov chains and develop computational procedures referred to as the *Almost Optimal* PMC/PQMC algorithms. We analyze these algorithms in terms of computational complexity, accuracy, and the trade-off between systematic and stochastic errors as functions of k and N.

The algorithms are implemented in MATLAB version 9.10 (R2021a). For the *Almost Optimal* PMC algorithm, we employ the Mersenne Twister (MT) and Middle Square (MS) PRNGs [48, 49]. For the *Almost Optimal* PQMC algorithm, randomized Sobol [50, 51] and Halton [31, 52] sequences are used. Numerical experiments were carried out on dense symmetric matrices of dimensions up to several hundred. We provide explicit comparisons between the *Almost Optimal* PMC/PQMC algorithms and their *classical* counterparts, highlighting the advantages of our approach, particularly the reduction in variance and lower computational cost for a given error tolerance.

The paper is organized as follows: Section 2 formulates the problem and introduces the *classical* Power method as well as the Stochastic Power methods (PMC and PQMC) for estimating the largest eigenvalue of dense symmetric matrices. Section 3 presents the *Almost Optimal* PMC/PQMC methods and algorithms, along with proofs and constructions of the random variables for general and specific cases. Section 4 reports numerical results and discussion. Section 5 concludes the study with final remarks and potential directions for future research.

2. Problem formulation and employed methods

2.1 *Formulation of the problem*

Given a non-singular symmetric matrix, $A = \{a_{ij}\}_{i,\,j=1}^{n} \in \mathbb{R}^{n\times n}$, $a_{ij} = a_{ji}$, $i,\ j = 1,\ \ldots,\ n$. Consider the eigenvalue problem: find $\lambda(A)$ such that:

$$A\boldsymbol{x} = \lambda\boldsymbol{x},\ \ \boldsymbol{x} \in \mathbb{R}^n \text{ is arbitrary non-zero vector.} \tag{1}$$

For symmetric matrices, all eigenvalues are real. Suppose that

$$\lambda_{min} = |\lambda_n| \leq |\lambda_{n-1}| \leq \ldots \leq |\lambda_2| < |\lambda_1| = \lambda_{max}. \tag{2}$$

Our task in this study is to propose computational algorithms based on the PMC and PQMC methods for estimating the maximum eigenvalue of symmetric dense matrices. We aim to investigate how stochastic and systematic errors depend on the parameters k and N. Additionally, we intend to examine the computational complexity and accuracy of the suggested PMC and QMC algorithms. This analysis will focus on how the selection of probabilities in the transition density matrix affects the reduction of the variance of a random variable constructed using a Markov chain.

2.2 *Classical Power method*

The Power method is a *classical* iterative technique widely used for estimating the maximum eigenvalue λ_{max} of square matrices [11, 12]. The iterative process can be described in the following 3 steps using the Rayleigh quotient [12]:

$$r(\boldsymbol{x}) = \frac{\boldsymbol{x}^T A \boldsymbol{x}}{\boldsymbol{x}^T \boldsymbol{x}},$$

as follows:

1. **Initialization:** Choose an initial vector $\boldsymbol{x}_0$ such that $\|\boldsymbol{x}_0\| = 1$.
2. **Iteration:** For $k = 0,\ 1,\ \ldots$ **Compute:** $\boldsymbol{x}_{k+1} = r(\boldsymbol{x}_k)$.
3. **Eigenvalue Approximation:** The corresponding maximum eigenvalue for problem (1), (2) after k iterations is estimated as:

$$\lambda_{max} \approx \lambda_{max}^{(k)} = r(\boldsymbol{x}_k).$$

Systematic error: The truncation error of the Power method in case of a symmetric matrix is

$$|\lambda_{max}^{(k)} - \lambda_{max}| = \mathscr{O}\Big(\Big|\frac{\lambda_2}{\lambda_1}\Big|^{2k}\Big). \tag{3}$$

Convergence: The rate of convergence of the iterative processor for symmetric matrices is characterised by the convergence factor $\left|\frac{\lambda_2}{\lambda_1}\right|^2$ (see [12]). Difficulties in convergence may be anticipated if the first two eigenvalues (in magnitude) are "close". In this instance, preconditioning techniques are utilised to distinguish between the two largest eigenvalues.

Computational complexity: The number of arithmetic operations is $O(kn^2)$ for the Power method, where k is several iterations in the above process, and n is the dimension of the matrix A.

2.3 *Stochastic Power method*

Consider again the eigenvalue problem (1), (2). The maximum eigenvalue can be written as a limit of the Rayleigh quotient [12, 45]:

$$\lambda_{max} = \lim_{k\to\infty} \frac{(\boldsymbol{h},\ A^k \boldsymbol{f})}{(\boldsymbol{h},\ A^{k-1}\boldsymbol{f})}, \tag{4}$$

where $(\boldsymbol{h},\ A^k\boldsymbol{f})$ is a scalar product, $\boldsymbol{f} = \{f_i\}_{i=1}^n$ and, $\boldsymbol{h} = \{h_i\}_{i=1}^n$ are arbitrary vectors. If k is an arbitrarily large natural number, then

$$\lambda_{max}^{(k)} = \frac{(\boldsymbol{h},\ A^k \boldsymbol{f})}{(\boldsymbol{h},\ A^{k-1}\boldsymbol{f})}, \tag{5}$$

is an approximation of λ_{max} with a certain systematic error.

The stochastic power method is based on constructing random variables whose mathematical expectations estimate the scalar products in (5). Consider the r.v. $\theta^{(k)}$ for estimating the scalar product $(\boldsymbol{h},\ A^k\boldsymbol{f})$ [32, 45] which employs a discrete Markov chain [16, 33]. Its construction can be described in the following steps:

1. **Constructing** of a discrete finite Markov chain with states l_t:

$$l_0 \to \cdots \to l_t \to \cdots \to l_k\ (1 \le l_t \le n,\ 0 \le t \le k) \tag{6}$$

with initial density vector $\boldsymbol{p} = \{p_i\}_{i=1}^n \in \mathbb{R}^n,\ \sum_{i=1}^n p_i = 1$ and probabilities $Pr(l_0 = i) = p_i \ge 0,\ i = 1,\ \ldots,\ n$ and the transition density matrix $P = \{p_{ij}\}_{i,\ j=1}^n \in \mathbb{R}^{n\times n},\ \sum_{j=1}^n p_{ij} = 1$ and $Pr(l_t = j | l_{t-1} = i) = p_{ij} \ge 0,\ (i,\ j = 1,\ \ldots,\ n)$ and $(t = 1,\ \ldots,\ k)$.

2. **Computing** the weight W_k iteratively:

$$W_0 = 1,\ W_t = W_{t-1}\frac{a_{l_{t-1}l_t}}{p_{l_{t-1}l_t}},\ t = 1,\ \ldots,\ k. \tag{7}$$

3. **Determining** the r.v. $\theta^{(k)}$ using the formula:

$$\theta^{(k)} = \frac{h_{l_0}}{p_{l_0}} W_k f_{l_k}. \tag{8}$$

To construct the weight W using the Markov chain (6), we define a set of *permissible* densities $\mathfrak{P}_h$ and $\mathfrak{P}_A$ depending on h and A, (see [45]).

Definition 1 The initial density vector $\boldsymbol{p} = \{p_i\}_{i=1}^n \in \mathbb{R}^n$ is called *permissible* for a vector $\boldsymbol{h} = \{h_i\}_{i=1}^n \in \mathbb{R}^n$, i.e. $\boldsymbol{p} \in \mathfrak{P}_{\boldsymbol{h}}$ if $p_i > 0$ when $h_i \ne 0$ and $p_i = 0$ when $h_i = 0$ for $i = 1,\ \ldots,\ n$.

Definition 2 The transition density matrix $P = \{p_{ij}\}_{i,\ j=1}^n \in \mathbb{R}^{n\times n}$ is called *permissible* for a symmetric matrix $A = \{a_{ij}\}_{i,\ j=1}^n \in \mathbb{R}^{n\times n}$, i.e. $P \in \mathfrak{P}_A$ if $p_{ij} > 0$ when $a_{ij} \ne 0$ and $p_{ij} = 0$ when $a_{ij} = 0$ for $i,\ j = 1,\ \ldots,\ n$.

Let us consider the N realization of the Markov chain (6) using Pseudorandom Number Generators (PRNGs) [16, 33]. Given a density vector $\boldsymbol{p}$ and a transition density matrix P. Define the mean value

$$\overline{\theta}_N^{(k)} = \frac{1}{N}\sum_{s=1}^N (\theta^{(k)})_s, \tag{9}$$

where the sth realization of the r.v. $\theta^{(k)}$ is denoted by $(\theta^{(k)})_s$, where $1 \le s \le N$.

The following theorem is valid:

Theorem 1 Let the r.v. $\theta^{(k)}$ is defined by formula (8). Then the mathematical expectation of the r.v. $\theta^{(k)}$ is equal to the scalar product $(\boldsymbol{h},\ A^k\boldsymbol{f})$, i.e.

$$E[\theta^{(k)}] = (\boldsymbol{h},\ A^k\boldsymbol{f}). \tag{10}$$

The proof of the Theorem 1 can be found in [45].

The mean value (9) is considered as an MC approximation of the scalar product $(\boldsymbol{h},\ A^k\boldsymbol{f})$ [45] with a probability error

$$R_N^{(k)} = |(\boldsymbol{h},\, A^k\boldsymbol{f}) - \overline{\theta}_N^{(k)}| < c_1\sigma(\theta^{(k)})N^{-1/2}, \tag{11}$$

where c_1 is a constant and $\sigma(\theta^{(k)})$ is a standard deviation.

When we use low-discrepancy sequences, like Sobol and Halton ones, to realise N times the Markov chain, then the mean value (9) is defined as a QMC approximation of the scalar product $(\boldsymbol{h},\, A^k\boldsymbol{f})$ [45]. The error of the QMC estimator is:

$$QR_N^{(k)} = |(\boldsymbol{h},\, A^k\boldsymbol{f}) - \overline{\theta}_N^{(k)}| < c_2\frac{(\log N)^{k+1}}{N}. \tag{12}$$

Here, $(k+1)$ is the dimension of each element in the low-discrepancy sequence, which is used to compute one value of r.v. $\theta^{(k)}$. The definition of the low-discrepancy sequences and Koksma-Hlawka inequality, which lead to the estimate (12) can be found in [5, 16].

The following theorem holds:

Theorem 2 Let us consider $\theta^{(k)}$ and $\theta^{(k-1)}$ for the fixed steps k and $k-1$ of the Markov chain (6). Then

$$\frac{E[\theta^{(k)}]}{E[\theta^{(k-1)}]} = \lambda_{max}^{(k)}. \tag{13}$$

Proof. Considering the construction of the random variables $\theta^{(k)}$ and $\theta^{(k-1)}$ for the fixed steps k and $k-1$ of the Markov chain, we have:

$$\theta^{(k)} = \frac{h_{l_0}}{p_{l_0}}W_k f_{l_k} \text{ and } \theta^{(k-1)} = \frac{h_{l_0}}{p_{l_0}}W_{k-1} f_{l_{k-1}}.$$

Taking into account equalities (5) and theorem 1 we obtain:

$$\frac{E[\theta^{(k)}]}{E[\theta^{(k-1)}]} = \frac{(\boldsymbol{h},\, A^k\boldsymbol{f})}{(\boldsymbol{h},\, A^{k-1}\boldsymbol{f})} = \lambda_{max}^{(k)}.$$

□

The stochastic Power method to estimate $\lambda_{max}^{(k)}$ is defined in the following way:

$$\lambda_{max}^{(k)} \approx \frac{\sum\limits_{s=1}^{N}(\theta^{(k)})_s}{\sum\limits_{s=1}^{N}(\theta^{(k-1)})_s}. \tag{14}$$

Here, the symbol "≈" indicates that the expression on the right side in (14) is an approximate solution for the $\lambda_{max}^{(k)}$ depending asymptotically on k and N. The meaning of this symbol is the same in all corollaries below.

Remark 1 The stochastic power method is referred to as the Power Monte Carlo (PMC) method when the samples for both random variables $\theta^{(k)}$ and $\theta^{(k-1)}$ are generated using random number generators to construct the Markov chain (6).

Thus, the stochastic error of the PMC method is a function of the probable error defined by (11).

Remark 2 When low-discrepancy sequences, such as Sobol sequences [24], are used to construct the Markov chain for sampling both random variables $\theta^{(k)}$ and $\theta^{(k-1)}$, the stochastic power method is referred to as the Power Quasi-Monte Carlo (PQMC) method.

Similarly, the stochastic error of the PMC method is a function of the probable error defined by (12). We note that the theoretical estimates for the stochastic errors of the PMC and PQMC methods are not found. We note that theoretical estimates for the stochastic errors of the PMC and PQMC methods are unavailable. This is an open problem for future work.

3. *Almost Optimal* PMC and PQMC methods and algorithms

3.1 *Estimates with Almost Optimal probabilities for the Markov chain: errors balance*

Let us present the symmetric matrix A in the following way

$$A = \{a_{ij}\}_{i,\,j=1}^{n} = (\boldsymbol{a}_1,\,\ldots,\,\boldsymbol{a}_i,\,\ldots,\,\boldsymbol{a}_n)^T,\ \text{where}\ \boldsymbol{a}_i = (a_{i1},\,\ldots,\,a_{in}),\ i = 1,\,\ldots,\,n.$$

The symbol T denotes transposition. Further, for our study, we consider the following vector and matrix norms:

$$\|\boldsymbol{h}\| = \|\boldsymbol{h}\|_1 = \sum_{i=1}^{n} |h_i|,\quad \|\boldsymbol{a}_i\| = \|\boldsymbol{a}_i\|_1 = \sum_{j=1}^{n} |a_{ij}|,\ \|A\| = \|A\|_1 = \max_j \sum_{i=1}^{n} |a_{ij}|,\quad j = 1,\,\ldots,\,n.$$

The main issue in the stochastic simulation is that the estimates obtained from random sampling can lead to high variance. Different simulations can yield significantly different results, particularly when the sample size N is small. According to an inequality (11), the probability error depends on the standard deviation σ (or the variance σ^2) and on the sample size N, which is the number of realisations of the Markov chain for our problem. In general, various variance reduction techniques (for example: antithetic variates, stratified sampling, importance sampling, QMC simulations using low-discrepancy sequences, etc.), (see [16, 33]) have been developed for the MC methods to improve their accuracy, to reduce the number of samples needed, and/or to increase the convergence of the method. Applying the variance reduction techniques is especially important when dealing with complex simulations or large-scale problems.

All permissible density vectors $\boldsymbol{p} \in \mathfrak{P}_{\boldsymbol{h}}$ and transition density matrices $P \in \mathfrak{P}_A$ can be employed to construct $\theta^{(k)}$ and $\theta^{(k-1)}$ to estimate $\lambda_{max}^{(k)}$ according to the stochastic Power method (14).

The most common approach for faster Markov chain construction is when the coordinates of the permissible density vector $\boldsymbol{p}$ and the elements of the transition density matrix P are set to $1/n$, i.e:

$$\boldsymbol{p} = \left\{p_i = \frac{1}{n}\right\}_{i=1}^{n} \quad \text{and}\ P = \left\{p_{ij} = \frac{1}{n}\right\}_{i,\,j=1}^{n},\ i,\ j = 1,\,\ldots,\,n. \tag{15}$$

We note that PMC/PQMC methods are referred to as *classical* when the choice (15) is used for the initial density vector and the transition density matrix in constructing the Markov chain.

Typically, MC solutions derived from alternative choices of the permissible density vector $\boldsymbol{p}$ and the transition density matrix P are compared to the *classical* approach to assess their accuracy and computational efficiency.

Among all permissible density vectors $\boldsymbol{p} \in \mathfrak{P}_{\boldsymbol{h}}$ and transition density matrices $P \in \mathfrak{P}_A$ there exists a choice that leads to a significant reduction in the variance [32, 45]:

$$\boldsymbol{p} = \{p_i\}_{i=1}^{n},\ p_i = \frac{|h_i|}{\|\boldsymbol{h}\|} \text{ and } P = \{p_{ij}\}_{i,\ j=1}^{n}\, p_{ij} = \frac{|a_{ij}|}{\|\boldsymbol{a}_i\|},\ i,\ j = 1\ldots.,\ n. \tag{16}$$

We observe (see construction of the Stochastic Power method below) that only for this choice is the random variable $\theta^{(k)}$ constructed fundamentally differently from that in formulas (7) and (8).

The initial density vector p is referred to as *Almost Optimal* initial density vector, and the transition density matrix P is referred to as *Almost Optimal* transition density matrix. The elements of the P are called *Almost Optimal* probabilities. Taking into account the choice of a vector $\boldsymbol{p}$ and a transition density matrix P according to formulas (16), the following theorem holds:

Theorem 3 Let $\boldsymbol{p}$ and P be the *Almost Optimal* initial density vector and *Almost Optimal* density matrix. Then the mean value (9) can be expressed as follows:

$$\overline{\theta}_N^{(k)} = \frac{1}{N}\sum_{s=1}^{N} sign\left(h_{l_0^{(s)}}\right)\|\boldsymbol{h}\|\left\{\prod_{t=1}^{t=k} sign\left(a_{l_{t-1}^{(s)}l_t^{(s)}}\right)\left\|\boldsymbol{a}_{l_{t-1}^{(s)}}\right\|\right\} f_{l_k^{(s)}}; \tag{17}$$

Proof. Taking into account formulas (8, 9) we obtain:

$$\overline{\theta}_N^{(k)} = \frac{1}{N}\sum_{s=1}^{N}(\theta^{(k)})_s = \frac{1}{N}\sum_{s=1}^{N}\left\{\frac{h_{l_0}}{p_{l_0}} W_k f_{l_k}\right\}_s = \frac{1}{N}\sum_{s=1}^{N}\frac{h_{l_0^{(s)}}}{p_{l_0^{(s)}}}\left\{\prod_{t=1}^{t=k}\frac{a_{l_{t-1}^{(s)}l_t^{(s)}}}{p_{l_{t-1}^{(s)}l_t^{(s)}}}\right\} f_{l_k^{(s)}}.$$

The proof of the theorem follows after applying formulas (16), i.e.

$$\overline{\theta}_N^{(k)} = \frac{1}{N}\sum_{s=1}^{N}\frac{h_{l_0^{(s)}}}{\left|h_{l_0^{(s)}}\right|}\|\boldsymbol{h}\|\left\{\prod_{t=1}^{t=k}\frac{a_{l_{t-1}^{(s)}l_t^{(s)}}}{\left|a_{l_{t-1}^{(s)}l_t^{(s)}}\right|}\left\|\boldsymbol{a}_{l_{t-1}^{(s)}}\right\|\right\} f_{l_k^{(s)}}$$

$$= \frac{1}{N}\sum_{s=1}^{N} sign\left(h_{l_0^{(s)}}\right)\|\boldsymbol{h}\|\left\{\prod_{t=1}^{t=k} sign\left(a_{l_{t-1}^{(s)}l_t^{(s)}}\right)\left\|\boldsymbol{a}_{l_{t-1}^{(s)}}\right\|\right\} f_{l_k^{(s)}}.$$

□

Based on Theorem 2 and Theorem 3, the following corollaries are true:

Corollary 1 Let $\boldsymbol{p}$ and P be the *Almost Optimal* initial density vector and *Almost Optimal* density matrix. Then the stochastic Power method for estimating $\lambda_{max}^{(k)}$ is defined as the ratio of the mean values $\overline{\theta}_N^{(k)}$ and $\overline{\theta}_N^{(k-1)}$ by formula (17), i.e.

$$\lambda_{max}^{(k)} \approx \frac{\sum_{s=1}^{N} sign\left(h_{l_0^{(s)}}\right)\left\{\prod_{t=1}^{t=k} sign\left(a_{l_{t-1}^{(s)}l_t^{(s)}}\right)\left\|\boldsymbol{a}_{l_{t-1}^{(s)}}\right\|\right\} f_{l_k^{(s)}}}{\sum_{s=1}^{N} sign\left(h_{l_0^{(s)}}\right)\left\{\prod_{t=1}^{t=k-1} sign\left(a_{l_{t-1}^{(s)}l_t^{(s)}}\right)\left\|\boldsymbol{a}_{l_{t-1}^{(s)}}\right\|\right\} f_{l_{k-1}^{(s)}}}. \tag{18}$$

Corollary 2 Suppose the L_1 norm of all row-vectors of the symmetric matrix A is equal to the constant a, i.e.

$$\|\boldsymbol{a}_i\|_1 = \|\boldsymbol{a}_i\| = \sum_{j=1}^{n} |a_{ij}| = a,\ i = 1,\ \ldots,\ n. \tag{19}$$

Then the estimation for $\lambda_{max}^{(k)}$ by formula (18) can be rewritten in a simpler way:

$$\lambda_{max}^{(k)} \approx \frac{\sum\limits_{s=1}^{N} sign\left(h_{l_0^{(s)}}\right)\left\{\prod\limits_{t=1}^{t=k} sign\left(a_{l_{t-1}^{(s)} l_t^{(s)}}\right)\right\} a f_{l_k^{(s)}}}{\sum\limits_{s=1}^{N} sign\left(h_{l_0^{(s)}}\right)\left\{\prod\limits_{t=1}^{t=k-1} sign\left(a_{l_{t-1}^{(s)} l_t^{(s)}}\right)\right\} f_{l_{k-1}^{(s)}}}. \tag{20}$$

Corollary 3 Let the L_1 norm of all row-vectors of the symmetric matrix A satisfy condition (19). Suppose also that the following condition holds for all elements of the A:

$$sign\left(h_{l_0^{(s)}}\right) \prod_{t=1}^{t=k-1} sign\left(a_{l_{t-1}^{(s)} l_t^{(s)}}\right) = 1,\ s = 1,\ \ldots,\ N. \tag{21}$$

Then $\lambda_{max}^{(k)}$ can be estimated by a simpler formula as follow:

$$\lambda_{max}^{(k)} \approx \frac{\sum\limits_{s=1}^{N} sign\left(a_{l_{k-1}^{(s)} l_k^{(s)}}\right) a f_{l_k^{(s)}}}{\sum\limits_{s=1}^{N} f_{l_{k-1}^{(s)}}}. \tag{22}$$

Remark 3 The set of symmetric matrices satisfying the conditions of Corollary 3 is nonempty. For instance, consider a symmetric matrix A with positive elements, where all main diagonal elements are equal, i.e., $a_{ii} = b > 0$, $i = 1, \ldots, n$, and all off-diagonal elements are also equal, i.e., $a_{ij} = d > 0$, $i \neq j$, $i, j = 1, \ldots, n$.

Taking into account that the systematic error ε_1 depends on the number of transitions in the Markov chain k and the stochastic error ε_2 depends on the number of realisations N, we obtain the following estimates for these errors:

$$\left|\lambda_{max} - \frac{\overline{\theta}_N^{(k)}}{\overline{\theta}_N^{(k-1)}}\right| = \left|\lambda_{max} - \lambda_{max}^{(k)} + \lambda_{max}^{(k)} - \frac{\overline{\theta}_N^{(k)}}{\overline{\theta}_N^{(k-1)}}\right|$$

$$\leq \left|\lambda_{max} - \lambda_{max}^{(k)}\right| + \left|\lambda_{max}^{(k)} - \frac{\overline{\theta}_N^{(k)}}{\overline{\theta}_N^{(k-1)}}\right| < \varepsilon_1 + \varepsilon_2 \tag{23}$$

We note the balance between the two errors is achieved when $\varepsilon_1 = \varepsilon_2 = \varepsilon$. Theoretical estimates of the balance between the two errors are presented in [44, 45].

Definition 3 PMC and PQMC methods for estimating the maximum eigenvalue of the symmetric matrix when choosing an initial density vector $\boldsymbol{p}$ and a transition density matrix P according to formulas (16), are called *Almost Optimal* PMC/PQMC methods.

3.2 *Almost Optimal PMC and Almost Optimal PQMC algorithms: computational complexity*

The pseudocode presented in Algorithm 1 describes the steps for estimating the maximum eigenvalue of a symmetric dense matrix based on formula (18) according to Corollary 1.

Algorithm 1 Pseudo code for computing the *Almost Optimal* PQMC algorithm

---

1. **Input:** matrix $A = \{a_{ij}\} \in \mathbb{R}^{n\times n}$; vectors $\boldsymbol{h}, \boldsymbol{f} \in \mathbb{R}^n$ and positive integers $n,\ N,\ k$
2. **Compute:** $\|\boldsymbol{a}_i\| = \sum_{j=1}^{n} |a_{ij}|,\ 1 \le i \le n$ and $\|\boldsymbol{h}\| = \sum_{i=1}^{n} |h_i|$
3. **Compute:** The coordinates of a permissible density vector $\boldsymbol{p}$ and the elements of a permissible density matrix P

$$p_i = \frac{|h_i|}{\|\boldsymbol{h}\|}, \quad p_{ij} = \frac{|a_{ij}|}{\|\boldsymbol{a}_i\|},\ i,\ j = 1,\ \ldots,\ n$$

4. **Generate:** N elements of the $(k+1)$-dimensional Sobol (or Halton) sequence
5. **Construct:** N realizations of the Markov chain with integer elements

$$l_0^{(s)} \to \ldots \to l_t^{(s)} \to \ldots \to l_{k-1}^{(s)} \to l_k^{(s)},\ 1 \le l_t^{(s)} \le n,\ 0 \le t \le k,\ 1 \le s \le N$$

6. **Compute:**

$$\overline{\theta}_N^{(k)} = \frac{1}{N}\sum_{s=1}^{N} sign\left(h_{l_0^{(s)}}\right) \|\boldsymbol{h}\| \left\{ \prod_{t=1}^{t=k} sign\left(a_{l_{t-1}^{(s)} l_t^{(s)}}\right) \left\|\boldsymbol{a}_{l_{t-1}^{(s)}}\right\| \right\} f_{l_k^{(s)}}$$

$$\overline{\theta}_N^{(k-1)} = \frac{1}{N}\sum_{s=1}^{N} sign\left(h_{l_0^{(s)}}\right) \|\boldsymbol{h}\| \left\{ \prod_{t=1}^{t=k-1} sign\left(a_{l_{t-1}^{(s)} l_t^{(s)}}\right) \left\|\boldsymbol{a}_{l_{t-1}^{(s)}}\right\| \right\} f_{l_{k-1}^{(s)}}$$

7. **Output:**

$$\lambda_{max}^{(k)} := \frac{\overline{\theta}_N^{(k)}}{\overline{\theta}_N^{(k-1)}}$$

Remark 4 Algorithm 1 is referred to as an *Almost Optimal* PQMC algorithm because low-discrepancy sequences are used in step 4. If PRNGs are used instead in step 4, Algorithm 1 is called an *Almost Optimal* PMC algorithm.

Remark 5 If the elements of the symmetric matrix A satisfy the conditions in Corollary 2 or Corollary 3, then the formulas in step 6 of Algorithm 1 can be replaced with the simpler ones, (20) or (22).

It is very important to estimate the computational complexity (or number of operations) of the Algorithm 1. Such estimates are significant when constructing multiple algorithms to solve a given problem. For Algorithm 1, the

computational complexity depends on the number of realisations of the Markov chain, the computational time for one realisation of the random variable $\theta^{(k)}$, and the computational cost for the transition density matrix P.

For steps 2 and 3 in Algorithm 1, we require $\mathcal{O}(n^2)$ operations. This part of Algorithm 1 is known as preprocessing.

The essential part of the algorithm lies in steps 4-7. For steps 4-6, the computational cost is estimated as $N\tau_k$, where N is the number of Markov chain realisations and τ_k is the average computational time for one realisation of the random variable $\theta^{(k)}$ and $\theta^{(k-1)}$ at fixed k.

Let τ_0 be the mean value of computational time for one transition in the Markov chain $l_{t-1} \to l_t$ when we construct the r.v. $\overline{\theta}_N^{(k)}$ or $\overline{\theta}_N^{(k-1)}$ by weight W_t (see formula (7)). To create a transition in the Markov chain, we need an algorithm to sample a discrete random variable with a finite number of states n and probabilities derived from the corresponding row of index l_t in the transition density matrix P. The computational cost for sampling such a discrete random variable is $\mathcal{O}(n)$. If we denote by τ_0 the average computational time of one jump in the Markov chain, we have $\tau_0 = \mathcal{O}(n)$, as this time depends on the matrix dimension. Thus, we obtain that $\tau_k = k\tau_0 = \mathcal{O}(nk)$. In this way, the computational cost of Algorithm 1 for steps 4-6 requires:

$$Cost(Alg1) = \mathcal{O}(Nnk) + \mathcal{O}(n^2) = \mathcal{O}(Nnk + n^2).$$

Algorithm 1 requeres $\mathcal{O}(1)$ operations in Step 7. In summary, the computational complexity of Algorithm 1 is $\mathcal{O}(Nnk)$ when $N > n$. In many cases, preprocessing operations are excluded, as there are an order of magnitude more operations in the main part. For sparse symmetric matrices where the number of nonzero elements is approximately n, the estimate for computation cost is expected to decrease significantly.

We expect that in cases involving dense symmetric matrices with dimensions of a few hundred, the balance between systematic and stochastic errors is achieved when $N = \mathcal{O}(n)$. The numerical experiments in the following section confirm this only in the scenario where we employ the *Almost Optimal* density matrices.

4. Numerical tests and discussion

The PMC algorithm is implemented in our numerical experiments using the Mersenne Twister (MT) [48] and Middle Square (MS) [49, 53] PRNGs. The Mersenne Twister PRNG has a period of 2^{19937}-1, and its output is free of long-term correlations when considered in 623 dimensions. The standard implementation employs a 32-bit word length, while an alternative implementation utilises a 64-bit word length [54]. The MT PRNG was designed to rectify most of the shortcomings found in older PRNGS. The MS PRNG was created using the Middle Square method.

We run the PMC algorithm using two types of probabilities in the Markov chain: (a) if we apply the initial density vector and the transition density matrix according to the formula (16), we have two versions of the *Almost Optimal* PMC algorithm, denoted as $\text{PMC}_{(MT)}$ and $\text{PMC}_{(MS)}$ depending on the PRNGs; (b) if we use the formula (15) for an initial density vector and a transition density matrix, we consider the *classical* PMC algorithm again in two versions referred to as $\text{PMC}^{(n)}_{(MT)}$ and $\text{PMC}^{(n)}_{(MS)}$.

Similarly, when we run the PQMC algorithm as *Almost Optimal* PQMC, we again consider two versions referred to as $\text{PQMC}_{(S)}$ and $\text{PQMC}_{(H)}$, depending on what kind of low-discrepancy sequences we use: Sobol (S) or Halton (H). In analogy, the two versions of the *classical* PQMC algorithm that employ Sobol sequences and Halton sequences are denoted as $\text{PQMC}^{(n)}_{(S)}$ and $\text{PQMC}^{(n)}_{(H)}$.

In practice, Sobol (S) and Halton (H) sequences are typically scrambled to enhance their effectiveness, facilitate statistical error estimation, and minimise certain artefacts. Various scrambling constructions are known and employed for both sequences, with a trade-off between complexity, computational time requirements, and uniformity properties. In our numerical tests, the Halton sequences are constructed using the "RR2" scramble type, which applies a reverse-radix operation to permute the radical inverse coefficients [31, 52]. This method effectively redistributes the points to achieve better uniformity. The Sobol sequences are constructed using the "Matousek-Affine-Owen" scramble type, which

combines a random linear scramble [50] with a random digital shift [51]. We note that the MT PRNG and the scrambling Sobol and Halton sequences are implemented in MATLAB version 9.10 (R2021a). Numerical results for estimating the largest eigenvalue of symmetric dense matrices are derived from two test matrices with randomly generated elements. The eigenvalues of these matrices satisfy condition (2), and the two cases are as follows:

i. Matrix A with dimension $n = 100$, $\lambda_{max} = \lambda_1 = 50.0408$, and $\lambda_2 = 4.0522$;

ii. Matrix A with dimension $n = 500$, $\lambda_{max} = \lambda_1 = 250.2454$, and $\lambda_2 = 9.0721$.

The arbitrary vectors $\boldsymbol{h}$ and $\boldsymbol{f}$ are chosen with norm 1, namelly $\|\boldsymbol{h}\| = \|\boldsymbol{f}\| = \sum_{i=1}^{n} \frac{1}{n}$, where $n = 100$ and $n = 500$.

In our experiments, the eigenvalues λ_1 and λ_2 were selected to be sufficiently well-separated to ensure favourable convergence properties for the *classical* Power method. Consequently, the systematic error becomes negligible after approximately four to five iterations.

This setup enables us to analyse the balance between statistical and systematic errors in stochastic algorithms. In particular, we investigate how the number of Markov chain transitions k and the number of realisations N of the random variables $\theta^{(k)}$ and $\theta^{(k-1)}$ affect this balance. Furthermore, we examine the impact of variance reduction in the *Almost Optimal* PMC/PQMC algorithms on the accuracy of the approximate solution, while maintaining constant computational complexity.

It is worth noting that in *classical* PMC/PQMC algorithms, the matrix A must be scaled by a positive parameter $q = \frac{1}{|\lambda_1| + \delta_1}$, with $\delta_1 > 0$, to ensure that all eigenvalues by modulus of the transformed matrix qA are less than 1. In the case where we do not have a prior estimate for λ_1, the parameter q can be chosen as follows: $q = \frac{1}{\max_i \|\boldsymbol{a}_i\| + \delta_1}$, where $(i = 1, \ldots, n)$, In contrast, for *Almost Optimal* PMC/PQMC algorithms, this scaling is not required, as the values of the random variables in step 6 of the Algorithm 1 primarily depend on the signs of the elements of the matrix A.

4.1 *Case (i): Matrix size n = 100*

Table 1 presents the numerical results for estimating the largest eigenvalue of the symmetric matrix in case (i), using the *Almost Optimal* $\text{PMC}_{(MT)}$ and *classical* $\text{PMC}^{(n)}_{(MT)}$ algorithms, across various values of N and k. Similarly, Table 2 displays the corresponding results for the *Almost Optimal* $\text{PMC}_{(MS)}$ and *classical* $\text{PMC}^{(n)}_{(MS)}$ algorithms. Both tables also include the absolute errors associated with the estimates. In both tables, the maximum eigenvalue $\lambda_1 = 50.0408$ of a 100×100 symmetric matrix is estimated across a range of realisations of N and iteration depths k.

The numerical experiments demonstrate that, for a fixed number of realisations N, increasing the number of transitions k in the Markov chain reduces the absolute error. This reduction stabilises beyond a certain value of k, suggesting an optimal balance between systematic and stochastic errors. When we fix $N = 128$, this value is $k = 9$. In cases where $N > 128$, this value is $k = 8$. With additional increases in k beyond this point, we observe that the resulting estimates do not enhance accuracy any further.

Note that the minimal absolute errors for each algorithm ($\text{PMC}_{(MT)}$, $\text{PMC}^{(n)}_{(MT)}$, $\text{PMC}_{(MS)}$, and $\text{PMC}^{(n)}_{(MS)}$) are highlighted in the Table 1 and Table 2. The lowest absolute error is observed at $(k = 8,\ N = 512)$ for the *Almost Optima* PMC algorithms (see Table 1 and Table 2), indicating that these values of k and N provide the best balance between systematic and stochastic errors. In the case of *classical* PMC algorithms, the best balance between systematic and stochastic errors is indicated when $k = 8$ and $N = 2,048$ (see Table 1 and Table 2).

When we compare the results of the *Almost Optimal* PMC algorithms with those of the *classical* PMC algorithms (see Table 1 and Table 2), we observe that for a fixed N and k, the absolute error for the *Almost Optimal* PMC algorithms is significantly lower than that of the *classical* PMC algorithms. For instance, at $N = 512$, $\text{PMC}_{(MT)}$ yields an error of 0.00005, while $\text{PMC}^{(n)}_{(MT)}$ yields 0.0015 (see Table 1). The same is situation when we compere the errors for $\text{PMC}_{(MS)}$ and $\text{PMC}^{(n)}_{(MS)}$ in Table 2. This is because, in constructing $\theta^{(k)}$ and $\theta^{(k-1)}$ for the *Almost Optimal* PMC algorithms, the variance is significantly reduced.

Table 1. $PMC_{(MT)}$ and $PMC^{(n)}_{(MT)}$ algorithm results for different N and k, on a 100×100 symmetric matrix with $\lambda_2/\lambda_1 \approx 0.081$, where $\lambda_1 = 50.0408$ and $\lambda_2 = 4.0522$ (* Highlighted rows represent optimal k values for each N)

N	k	$\lambda_1^{(k)}$ using $PMC_{(MT)}$	Absolute error $PMC_{(MT)}$	$\lambda_1^{(k)}$ using $PMC^{(n)}_{(MT)}$	Absolute error $PMC^{(n)}_{(MT)}$
128	6	50.012316	0.02848	49.893384	0.1474
	7	50.025579	0.01522	49.997142	0.0437
	8	50.039901	0.00090	50.019883	0.0209
	9	**50.040709**	**0.00009**	**50.037795**	**0.0030**
	10	50.045295	0.00450	50.066104	0.0253
	11	50.054378	0.01358	50.104543	0.0637
256	6	50.017864	0.02294	49.919982	0.1208
	7	50.023923	0.01688	50.012158	0.0286
	8	**50.040628**	**0.00017**	**50.039528**	**0.0013**
	9	50.044896	0.00410	50.050593	0.0098
	10	50.050896	0.01010	50.081757	0.0410
	11	50.070174	0.02937	50.093002	0.0522
512	6	50.032910	0.00789	49.676084	0.3647
	7	50.035045	0.00576	49.996040	0.0448
	8	**50.040750**	**0.00005**	**50.039264**	**0.0015**
	9	50.052027	0.01123	50.050158	0.0094
	10	50.058593	0.01779	50.072966	0.0322
	11	50.060141	0.01934	50.109447	0.0686
1,024	6	50.015868	0.02493	49.770608	0.2702
	7	50.029828	0.01097	50.019435	0.0214
	8	**50.040720**	**0.00008**	**50.039735**	**0.0011**
	9	50.045403	0.00460	50.054351	0.0136
	10	50.053595	0.01280	50.069878	0.0291
	11	50.060302	0.01950	50.080112	0.0393
2,048	6	50.035629	0.00517	49.851866	0.1889
	7	50.036291	0.00451	50.029563	0.0112
	8	**50.041086**	**0.00029**	**50.041728**	**0.0009**
	9	50.043850	0.00305	50.057179	0.0164
	10	50.066103	0.02530	50.077047	0.0362
	11	50.069460	0.02866	50.091154	0.0504

Figure 1 displays eigenvalue estimates across increasing numbers of realisations N (along the X-axis), using the optimal iteration depth k (along the Y-axis) identified by boldface in Table 1 and Table 2. Here, we compare *Almost Optimal* PMC variants against their *classical* PMC variants. $PMC^{(n)}_{(MT)}$ achieves accuracy and stability, closely approximating λ_{max} as N increases. According to Table 1, $PMC^{(n)}_{(MT)}$ produces an absolute error as low as 9×10^{-4} at $N = 2,048$, outperforming $PMC^{(n)}_{(MS)}$, which reaches an error of 1.8×10^{-3} (see Table 2).

The visual evidence in Figure 1 confirms this trend: $PMC^{(n)}_{(MT)}$ steadily converges towards the true value. In contrast, $PMC^{(n)}_{(MS)}$ fluctuates and even overshoots for larger N, indicating instability possibly due to pseudorandom variability and weaker generator uniformity in high dimensions.

Table 2. $\text{PMC}_{(MS)}$ and $\text{PMC}^{(n)}_{(MS)}$ algorithm results for different N and k, on a 100×100 symmetric matrix with $\lambda_2/\lambda_1 \approx 0.081$, where $\lambda_1 = 50.0408$ and $\lambda_2 = 4.0522$ (* Highlighted rows represent optimal k values for each N)

N	k	$\lambda_1^{(k)}$ using $\text{PMC}_{(MS)}$	Absolute error $\text{PMC}_{(MS)}$	$\lambda_1^{(k)}$ using $\text{PMC}^{(n)}_{(MS)}$	Absolute error $\text{PMC}^{(n)}_{(MS)}$
128	6	50.019125	0.02167	49.564508	0.4763
	7	50.023944	0.01686	49.988941	0.0519
	8	50.039440	0.00136	50.019388	0.0214
	9	**50.040595**	**0.00020**	**50.029883**	**0.0109**
	10	50.051262	0.01046	50.060923	0.0201
	11	50.084762	0.04396	50.109231	0.0684
256	6	50.011148	0.02965	49.574876	0.4659
	7	50.031044	0.00976	49.984883	0.0559
	8	**50.042490**	**0.00169**	**50.028211**	**0.0126**
	9	50.050490	0.00969	50.051251	0.0105
	10	50.076397	0.03560	50.084891	0.0441
	11	50.094549	0.05375	50.103125	0.0623
512	6	50.028121	0.01268	49.679757	0.3610
	7	50.030518	0.01028	49.997989	0.0428
	8	**50.040617**	**0.00018**	**50.033570**	**0.0072**
	9	50.058913	0.01811	50.053570	0.0128
	10	50.061048	0.02025	50.091035	0.0502
	11	50.064531	0.02373	50.109567	0.0688
1,024	6	50.012038	0.02876	49.752343	0.2885
	7	50.031472	0.00933	50.009835	0.0310
	8	**50.041209**	**0.00041**	**50.038850**	**0.0020**
	9	50.044938	0.00414	50.056396	0.0156
	10	50.048139	0.00734	50.076755	0.0360
	11	50.053169	0.01237	50.093669	0.0529
2,048	6	50.031902	0.00890	49.839506	0.02013
	7	50.038998	0.00180	50.022658	0.0181
	8	**50.041243**	**0.00044**	**50,038951**	**0.0018**
	9	50.043045	0.00225	50.063537	0.0227
	10	50.058875	0.01808	50.078798	0.0380
	11	50.084113	0.04331	50.098828	0.0580

Both *Almost Optimal* versions ($\text{PMC}_{(MT)}$ and $\text{PMC}_{(MS)}$) maintain a relatively flat trajectory close to the target eigenvalue, and demonstrate better accuracy at smaller N than their *classical* counterparts. This is due to the reduced variance resulting from the choice of the transition density matrix for the Markov chain. These results highlight that *Almost Optimal* PMC algorithms outperform *classical* PMC algorithms regarding computational complexity and accuracy, particularly when employing a robust generator such as MT.

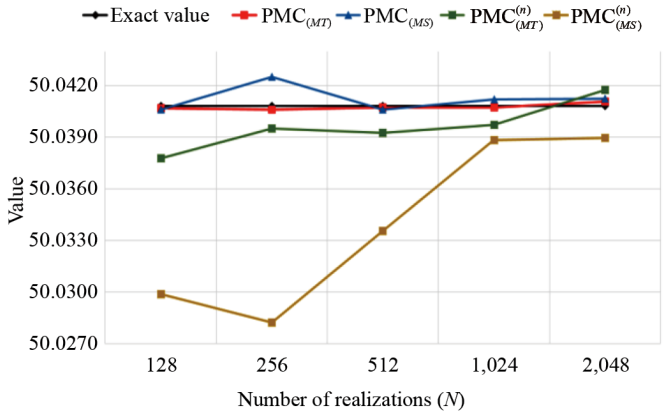

Figure 1. ($PMC_{(MT)}$ and $PMC_{(MS)}$) vs ($PMC^{(n)}_{(MT)}$ and $PMC^{(n)}_{(MS)}$). Comparison of the maximum eigenvalue λ_{max} of the symmetric 100×100 matrix with estimated values $\lambda^{(k)}_{max}$ obtained in the balancing cases (in the optimal iteration depth k), i. e. when $(k = 9,\ N = 128)$ and $(k = 8,\ N > 128)$

Table 3. $PQMC_{(H)}$ and $PQMC^{(n)}_{(H)}$ algorithm results for various N and k, applied to a 100×100 symmetric matrix. The absolute errors for the case $\lambda_2/\lambda_1 \approx 0.081$ (* Highlighted rows represent optimal k values for each N)

N	k	$\lambda_1^{(k)}$ using $PQMC_{(H)}$	Absolute error $PQMC_{(H)}$	$\lambda_1^{(k)}$ using $PQMC^{(n)}_{(H)}$	Absolute error $PQMC^{(n)}_{(H)}$
128	9	49.667323	0.37348	49.051280	0.9895
	10	49.732875	0.30792	49.313447	0.7274
	11	49.801798	0.23900	49.574856	0.4659
	12	**50.037989**	**0.00281**	**50.206208**	**0.1654**
	13	50.152986	0.11219	50.314121	0.2733
	14	50.246958	0.20616	50.542813	0.5020
256	9	49.982000	0.05880	49.674872	0.3659
	10	50.001834	0.03897	49.739820	0.3010
	11	**50.038996**	**0.00180**	**50.199525**	**0.1587**
	12	50.067986	0.02719	50.229280	0.1885
	13	50.135149	0.09435	50.299014	0.2582
	14	50.379962	0.33916	50.387742	0.3469
512	9	49.987751	0.05305	49.676566	0.3642
	10	50.017989	0.02281	49.749688	0.2911
	11	**50.039899**	**0.00090**	**50.176724**	**0.1359**
	12	50.178512	0.13771	50.196566	0.1558
	13	50.228992	0.18819	50.299461	0.2587
	14	50.357840	0.31704	50.358776	0.3180
1,024	9	49.452553	0.58825	49.453967	0.5868
	10	49.976987	0.06381	49.806135	0.2347
	11	**50.039772**	**0.00103**	**50.172345**	**0.1315**
	12	50.107985	0.06719	50.295866	0.2551
	13	50.282004	0.24120	50.298343	0.2575
	14	50.329662	0.28886	50.339248	0.2984
2,048	9	49.681987	0.35881	49.670730	0.3701
	10	49.973344	0.06746	49.907160	0.1336
	11	**50.039982**	**0.00082**	**50.111519**	**0.0707**
	12	50.183245	0.14245	50.275169	0.2344
	13	50.238237	0.19744	50.291489	0.2507
	14	50.407066	0.36627	50.407572	0.3668

Table 4. $\mathrm{PQMC}_{(S)}$ and $\mathrm{PQMC}_{(S)}^{(n)}$ algorithm results for different N and k, applied to a 100×100 symmetric matrix. The absolute errors for the case $\lambda_2/\lambda_1 \approx 0.081$ (* Highlighted rows represent optimal k values for each N)

N	k	$\lambda_1^{(k)}$ using $\mathrm{PQMC}_{(S)}$	Absolute error $\mathrm{PQMC}_{(S)}$	$\lambda_1^{(k)}$ using $\mathrm{PQMC}_{(S)}^{(n)}$	Absolute error $\mathrm{PQMC}_{(S)}^{(n)}$
128	9	49.393214	0.64759	49.215148	0.8257
	10	49.597734	0.44307	49.491180	0.5496
	11	49.679383	0.36142	49.624160	0.4166
	12	**50.040544**	**0.00026**	**50.200481**	**0.1597**
	13	50.091116	0.05032	50.299826	0.2590
	14	50.181771	0.14097	50.399651	0.3589
256	9	49.780806	0.25999	49.428552	0.6122
	10	49.994318	0.04648	49.742961	0.2978
	11	**50.039989**	**0.00081**	**50.186264**	**0.1455**
	12	50.049407	0.00861	50.225854	0.1851
	13	50.153503	0.11270	50.263529	0.2227
	14	50.327859	0.28706	50.386282	0.3455
512	9	49.966398	0.07440	49.543755	0.4970
	10	50.050000	0.00920	49.866152	0.1746
	11	**50.040585**	**0.00021**	**50.166400**	**0.1256**
	12	50.148019	0.10722	50.223877	0.1831
	13	50.209448	0.16865	50.271020	0.2302
	14	50.456838	0.41604	50.383010	0.3422
1,024	9	49.765493	0.27531	49.617372	0.4234
	10	49.986946	0.05385	49.933245	0.1076
	11	**50.040913**	**0.00011**	**50.132603**	**0.0918**
	12	50.087435	0.04664	50.167252	0.1265
	13	50.218753	0.17795	50.280149	0.2393
	14	50.298935	0.25814	50.288215	0.2474
2,048	9	49.707851	0.33295	49.681502	0.3593
	10	49.972883	0.06792	49.957605	0.0832
	11	**50.040854**	**0.00005**	**50.074364**	**0.0336**
	12	50.155992	0.11519	50.186986	0.1462
	13	50.222261	0.18146	50.239941	0.1991
	14	50.304974	0.26417	50.407318	0.3665

Table 3 and Table 4 report the performance of the *Almost Optimal* PQMC and *classical* PQMC algorithms using Halton and Sobol sequences, respectively, for computing the dominant eigenvalue of a 100×100 symmetric matrix. In both tables, the maximum eigenvalue $\lambda_1 = 50.0408$ is estimated across a range of realisations of N and iteration depths k. For each algorithm ($\mathrm{PQMC}_{(H)}$, $\mathrm{PMC}_{(H)}^{(n)}$, $\mathrm{PQMC}_{(S)}$, $\mathrm{PQMC}_{(S)}^{(n)}$), the absolute error decreases as k increases, with optimal convergence typically observed at $k = 11$ or $k = 12$ as highlighted. $\mathrm{PQMC}_{(S)}$ consistently outperforms $\mathrm{PQMC}_{(H)}$ across all N, demonstrating both lower errors and smoother convergence. For instance, at $N = 2,048$, $\mathrm{PQMC}_{(S)}$ yields an error of 0.00005, while $\mathrm{PQMC}_{(H)}$ yields 0.00082 (see Table 3 and Table 4). This reflects the superior uniformity of Sobol sequences for higher dimensions.

The *classical* variants ($\mathrm{PQMC}_{(H)}^{(n)}$, $\mathrm{PQMC}_{(S)}^{(n)}$) improve robustness, particularly at higher N. Although initially larger in error, $\mathrm{PMC}_{(H)}^{(n)}$ and $\mathrm{PQMC}_{(S)}^{(n)}$ converge with increasing N. At $N = 2,048$, $\mathrm{PQMC}_{(S)}^{(n)}$ achieves 0.0336 compared to 0.0707 for $\mathrm{PQMC}_{(H)}^{(n)}$, confirming the continued advantage of the Sobol-based PQMC algorithm even in the *classical* configuration.

When we compare the results of the *Almost Optimal* PQMC algorithm with those of the *classical* PQMC algorithm (see Table 3 and Table 4), we observe that for a fixed N and k, the absolute error for the *Almost Optimal* PQMC algorithm

is significantly lower than that of the *classical* PQMC algorithm. For instance, at $N = 256$ and $k = 11$, $\text{PQMC}_{(S)}$ yields an error of 0.00081, while $\text{PQMC}_{(S)}^{(n)}$ yields 0.1455 (see Table 4).

The same situation occurs when we compare the errors for $\text{PQMC}_{(H)}$ and $\text{PQMC}_{(H)}^{(n)}$ in Table 3. At $N = 256$ and $k = 11$, $\text{PQMC}_{(H)}$ yields an error of 0.0018, while $\text{PQMC}_{(H)}^{(n)}$ yields 0.1587. This is because the variance is considerably reduced in constructing $\theta^{(k)}$ and $\theta^{(k-1)}$ for the *Almost Optimal* PQMC algorithm.

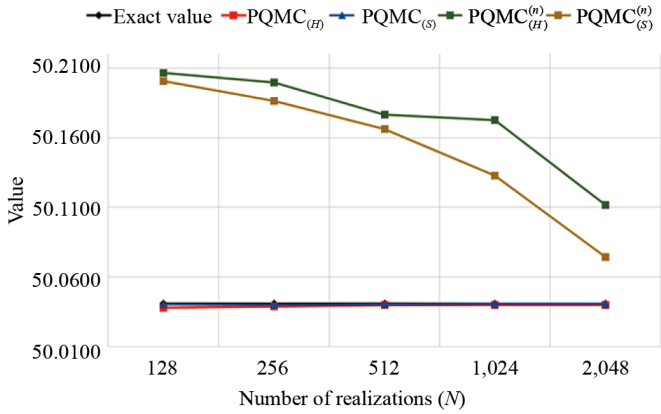

Figure 2. ($\text{PQMC}_{(S)}$ and $\text{PQMC}_{(H)}$) vs ($\text{PQMC}_{(S)}^{(n)}$ and $\text{PQMC}_{(H)}^{(n)}$). Comparison of the maximum eigenvalue λ_{max} of the symmetric 100×100 matrix with estimated values $\lambda_{max}^{(k)}$ obtained in the balancing cases, i. e. when $(k = 12,\ N = 128)$ and $(k = 11,\ N > 128)$

Figure 2 extends the analysis by comparing the *classical* PQMC algorithms with the *Almost Optimal* PQMC variants. Notably, $\text{PQMC}_{(S)}$ demonstrates superior performance, closely tracking the exact eigenvalue across all sample sizes. The same is true for $\text{PQMC}_{(H)}$. In contrast, $\text{PQMC}_{(S)}^{(n)}$ and $\text{PQMC}_{(H)}^{(n)}$ initially overestimate λ_{max}, but subsequently converge slowly as N increases.

These results underscore that *Almost Optimal* PQMC algorithms outperform *classical* PQMC algorithms in terms of computational complexity and accuracy, attributable to the reduced variance stemming from the choice of the transition density matrix for the Markov chain.

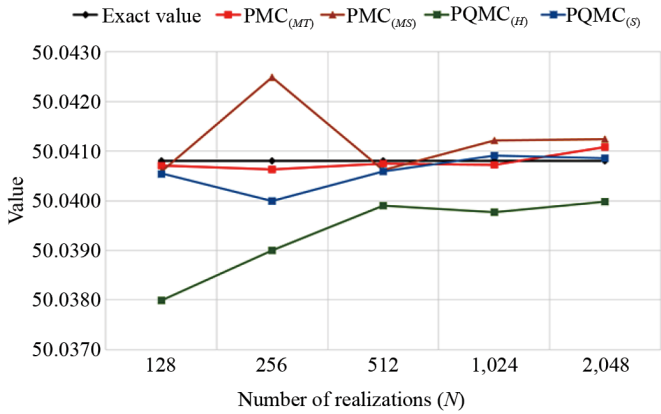

Figure 3. Comparison of λ_{max} of the symmetric 100×100 matrix with $\lambda_{max}^{(k)}$ in the balancing cases using the *Almost Optimal* PMC algorithm $(k = 9,\ N = 128),\ (k = 8,\ N > 128)$ and the *Almost Optimal* PQMC algorithm $(k = 12,\ N = 128),\ (k = 11,\ N > 128)$

Figure 3 compares *Almost Optimal* PMC algorithm using the MT and MS PRNGs with *Almost Optimal* PQMC algorithm using Halton (H) and Sobol (S) sequences.

We refine the scale on the Y-axis to observe that all approximate values of $\lambda_{max}^{(k)}$ fall within the interval $\delta < 0.005$ surrounding the maximum eigenvalue λ_{max}. Thus, we can notice differences when comparing the four variants of the *Almost Optimal* PMC and *Almost Optimal* PQMC algorithms. $\text{PMC}_{(MT)}$ shows strong convergence behaviour, with errors reaching a minimum of 8×10^{-5} at $N = 1{,}024$ and $k = 8$ (see Table 1). In contrast, $\text{PMC}_{(MS)}$ (see Table 2) demonstrates greater variance and slightly higher errors, although still within acceptable limits (e.g., 4.4×10^{-4} at $N = 2{,}048$ and $k = 8$). This is further confirmed in Figure 3, where $\text{PMC}_{(MT)}$ tracks the true eigenvalue more closely than $\text{PMC}_{(MS)}$.

Concerning the *Almost Optimal* PQMC algorithms, $\text{PQMC}_{(S)}$ stands out as the most precise among them, demonstrating consistent performance and minimal deviation from the exact value, as illustrated both visually in Figure 3 and numerically in Table 4. For example, at $N = 2{,}048$ and $k = 11$, the absolute error is just 5×10^{-5}. In contrast, $\text{PQMC}_{(H)}$, while still improving with larger N, lags due to its poorer multidimensional uniformity, achieving an error of 8.2×10^{-4} under the same configuration (Table 3). Regarding the optimal balance between stochastic and systematic errors, the test experiments for a dense symmetric matrix with size $n = 100$ indicate that it is achieved when $k = 8$ for $\text{PMC}_{(MT)}$ and $k = 11$ for $\text{PQMC}_{(S)}$ across all realisations of N.

These results demonstrate that *Almost Optimal* PMC/PQMC algorithms, particularly those based on the MT and scrambled Sobol sequences, offer significantly greater accuracy than their *classical* counterparts for the same computational cost.

4.2 *Case (ii): Matrix size n = 500*

Table 5 presents the numerical results for estimating $\lambda_{max}^{(k)}$ of a 500×500 symmetric dense matrix across a range of realisations of N and iteration depths k. The results were obtained using the *Almost Optimal* $\text{PMC}_{(MT)}$ and *classical* $\text{PMC}_{(MT)}^{(n)}$ algorithms. Similarly, Table 6 displays the corresponding results for the *Almost Optimal* $\text{PMC}_{(MS)}$ and *classical* $\text{PMC}_{(MS)}^{(n)}$ algorithms. Both tables also include the absolute errors associated with the estimates.

The numerical experiments demonstrate that, for a fixed number of realisations N, increasing the number of transitions k in the Markov chain reduces the absolute error. This reduction stabilises beyond a certain value of k, suggesting an optimal balance between systematic and stochastic errors. This value is $k = 9$, (see Tables 5 and Table 6). With additional increases in k beyond this point, we observe that the resulting estimates do not enhance accuracy any further. Note that the minimal absolute errors for each algorithm ($\text{PMC}_{(MT)}$, $\text{PMC}_{(MT)}^{(n)}$, $\text{PMC}_{(MS)}$, and $\text{PMC}_{(MS)}^{(n)}$) when $k = 9$ are highlighted in the Table 5 and Table 6.

The lowest absolute error is observed at $(k = 9,\ N = 512)$ for $\text{PMC}_{(MT)}$ and at $(k = 9,\ N = 128)$ for $\text{PMC}_{(MS)}$ (see Table 5 and Table 6), indicating that these values of k and N provide the best balance between systematic and stochastic errors.

In the case of *classical* PMC algorithms, the best balance between systematic and stochastic errors is indicated when $k = 9$ (see Table 5 and Table 6).

When we compare the results of the *Almost Optimal* PMC algorithms with those of the *classical* PMC algorithms (see Table 5 and Table 6), we observe that for a fixed N and k, the absolute error for the *Almost Optimal* PMC algorithms is significantly lower than that of the *classical* PMC algorithms. The absolute error in the balanced case of $k = 9$ in the *Almost Optimal* PMC algorithms is on the order of 0.0001, whereas in the *classical* PMC algorithms it is on the order of 0.01. This is because, in constructing $\theta^{(k)}$ and $\theta^{(k-1)}$ for the *Almost Optimal* PMC algorithms, the variance is significantly reduced.

Table 5. $\text{PMC}_{(MT)}$ and $\text{PMC}^{(n)}_{(MT)}$ algorithm results for different N and k, on a 500×500 symmetric matrix with $\lambda_2/\lambda_1 \approx 0.0363$, where $\lambda_1 = 250.2454$ and $\lambda_2 = 9.0721$ (* Highlighted rows represent optimal k values for each N)

N	k	$\lambda_1^{(k)}$ using $\text{PMC}_{(MT)}$	Abs. error $\text{PMC}_{(MT)}$	$\lambda_1^{(k)}$ using $\text{PMC}^{(n)}_{(MT)}$	Abs. error $\text{PMC}^{(n)}_{(MT)}$
128	6	250.181100	0.06430	248.863095	1.3823
	7	250.196274	0.04913	249.073556	1.1718
	8	250.222396	0.02300	249.492834	0.7526
	9	**250.232288**	**0.01311**	250.032970	0.2124
	10	250.295027	0.04963	**250.149293**	**0.0961**
	11	250.301310	0.05591	250.391291	0.1459
256	6	250.071617	0.17378	249.499761	0.7456
	7	250.114415	0.13098	249.807572	0.4378
	8	250.228998	0.01640	250.124994	0.1204
	9	**250.250130**	**0.00473**	**250.222275**	**0.0231**
	10	250.283657	0.03826	250.311181	0.0658
	11	250.327734	0.08233	250.363882	0.1185
512	6	250.072767	0.17263	249.680711	0.5647
	7	250.161043	0.08436	250.072663	0.1727
	8	250.235302	0.01011	250.115109	0.1303
	9	**250.245110**	**0.00029**	**250.192305**	**0.0531**
	10	250.268302	0.02290	250.328365	0.0830
	11	250.306227	0.06083	250.458641	0.2132
1,024	6	250.167399	0.07800	249.592590	0.6528
	7	250.211423	0.03398	250.075384	0.1700
	8	250.235306	0.01009	250.164200	0.0812
	9	**250.245015**	**0.00038**	**250.205542**	**0.0399**
	10	250.269913	0.02451	250.300328	0.0549
	11	250.287417	0.04202	250.495723	0.2503
2,048	6	250.160017	0.08538	249.864998	0.3804
	7	250.193403	0.05200	250.039826	0.2056
	8	250.226172	0.01923	250.175939	0.0695
	9	**250.246938**	**0.00154**	**250.224397**	**0.0210**
	10	250.251779	0.00638	250.295497	0.0501
	11	250.258464	0.01306	250.480189	0.2348

Figure 4 displays eigenvalue estimates across increasing numbers of realisations N, using the optimal iteration depth $k = 9$ identified by boldface in Tables 5 and 6. Here, we compare *Almost Optimal* PMC variants against their *classical* PMC variants. $\text{PMC}^{(n)}_{(MT)}$ achieves accuracy and stability, closely approximating λ_{max} as N increases. The visual evidence in Figure 4 confirms this trend: $\text{PMC}^{(n)}_{(MT)}$ converges steadily towards the true value, while $\text{PMC}^{(n)}_{(MS)}$ fluctuates and even overshoots for larger N, indicating instability possibly due to pseudo-random variability and weaker generator uniformity in high dimensions. The Figure 4 illustrates that $\text{PMC}_{(MT)}$ and $\text{PMC}_{(MS)}$ maintain a relatively flat trajectory very close to the target eigenvalue, λ_{max}, and demonstrate improved accuracy at smaller N compared to their *classical* counterparts. This is due to the reduced variance resulting from the choice of the transition density matrix for the Markov chain. These results again highlight that *Almost Optimal* PMC algorithms outperform *classical* PMC algorithms in computational complexity and accuracy, particularly when employing a robust generator such as MT.

Table 6. $PMC_{(MS)}$ and $PMC^{(n)}_{(MS)}$ algorithm results for different N and k, on a 500×500 symmetric matrix with $\lambda_2/\lambda_1 \approx 0.0363$, where $\lambda_1 = 250.2454$ and $\lambda_2 = 9.0721$ (* Highlighted rows represent optimal k values for each N)

N	k	$\lambda_1^{(k)}$ using $PMC_{(MS)}$	Abs. error $PMC_{(MS)}$	$\lambda_1^{(k)}$ using $PMC^{(n)}_{(MS)}$	Abs. error $PMC^{(n)}_{(MS)}$
128	6	250.085322	0.16008	248.522547	1.7229
	7	250.185322	0.06008	248.931561	1.3138
	8	250.211445	0.03395	249.462722	0.7827
	9	**250.245247**	**0.00015**	250.003068	0.2423
	10	250.279476	0.03408	**250.120230**	**0.1252**
	11	250.356308	0.11109	250.408768	0.1634
256	6	250.104179	0.14122	249.129460	1.1159
	7	250.177974	0.06743	249.701820	0.5436
	8	250.203720	0.04168	250.015811	0.2296
	9	**250.252044**	**0.00664**	**250.221130**	**0.0243**
	10	250.290513	0.04511	250.363882	0.1185
	11	250.308344	0.06294	250.668091	0.4227
512	6	250.086724	0.15868	249.219447	1.0260
	7	250.177427	0.06797	249.501899	0.7435
	8	250.206884	0.03852	250.058060	0.1873
	9	**250.244890**	**0.00051**	**250.183410**	**0.0620**
	10	250.283279	0.03788	250.356760	0.1114
	11	250.297526	0.05213	250.501899	0.2565
1,024	6	250.017168	0.22823	249.602843	0.6426
	7	250.189383	0.05602	250.051538	0.1939
	8	250.238977	0.00642	250.143770	0.1016
	9	**250.244751**	**0.00065**	**250.268955**	**0.0236**
	10	250.270855	0.02546	250.396782	0.1514
	11	250.306134	0.06073	250.493687	0.2483
2,048	6	250.118546	0.12685	249.911061	0.3343
	7	250.187878	0.05752	250.034691	0.2107
	8	250.237612	0.00779	250.103221	0.1422
	9	**250.247710**	**0.00231**	**250.320873**	**0.0755**
	10	250.275206	0.02981	250.347819	0.1024
	11	250.290595	0.04520	250.494239	0.2488

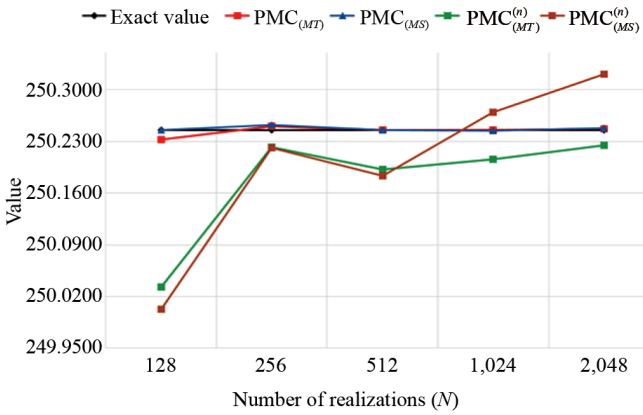

Figure 4. ($PMC_{(MT)}$ and $PMC_{(MS)}$) vs ($PMC^{(n)}_{(MT)}$ and $PMC^{(n)}_{(MS)}$). Comparison of the maximum eigenvalue λ_{max} of the symmetric 500×500 matrix with estimated values $\lambda^{(k)}_{max}$ obtained in the balancing cases, i. e. when ($k = 9$, $N \geq 128$)

Table 7. $\text{PQMC}_{(H)}$ and $\text{PQMC}_{(H)}^{(n)}$ algorithm results for different N and k, applied to a 500×500 symmetric matrix. The absolute for the case $\lambda_2/\lambda_1 \approx 0.0363$ (* Highlighted rows represent optimal k values for each N)

N	k	$\lambda_1^{(k)}$ using $\text{PQMC}_{(H)}$	Absolute error $\text{PQMC}_{(H)}$	$\lambda_1^{(k)}$ using $\text{PQMC}_{(H)}^{(n)}$	Absolute error $\text{PQMC}_{(H)}^{(n)}$
128	9	249.471697	0.77370	245.712684	4.5327
	10	250.066438	0.17896	246.512652	3.7327
	11	250.167707	0.07769	251.486594	1.2412
	12	**250.249391**	**0.00399**	**249.309133**	**0.9363**
	13	250.357823	0.11242	251.999063	1.7537
	14	250.586571	0.34117	252.780009	2.5346
256	9	249.858433	0.38697	247.418360	2.8270
	10	249.991341	0.25406	248.248562	1.9968
	11	250.197841	0.04756	249.470953	0.7744
	12	**250.239655**	**0.00574**	**250.770441**	**0.5250**
	13	250.357573	0.11217	251.711421	1.4660
	14	250.472703	0.22730	251.797723	1.5523
512	9	249.592272	0.65313	248.042800	2.2026
	10	249.896185	0.34921	249.080394	1.1650
	11	250.089218	0.15618	249.591525	0.6539
	12	**250.236033**	**0.00937**	**250.669970**	**0.4246**
	13	250.369509	0.12411	250.978674	0.7333
	14	250.534697	0.28930	251.788278	1.5429
1,024	9	249.959642	0.28576	248.998184	1.2472
	10	250.059089	0.18631	249.131528	1.1139
	11	250.127756	0.11764	249.672942	0.5725
	12	**250.250718**	**0.00532**	**250.090116**	**0.1553**
	13	250.387049	0.14165	250.829267	0.5839
	14	250.481235	0.23584	251.662872	1.4175
2,048	9	249.972251	0.27315	248.999883	1.2455
	10	250.055927	0.18947	249.408467	0.8369
	11	250.166549	0.07885	249.987224	0.2582
	12	**250.249986**	**0.00459**	**250.172833**	**0.0726**
	13	250.321274	0.07587	250.581459	0.3361
	14	250.439084	0.19368	251.216429	0.9710

Table 7 and Table 8 report numerical results for estimating $\lambda_{max}^{(k)}$ obtained by the *Almost Optimal* PQMC and *classical* PQMC algorithms using Halton (H) and Sobol (S) sequences, respectively. In both tables, the maximum eigenvalue $\lambda_1 = 250.2454$ is estimated across a range of realisations of N and iteration depths k. For each algorithm ($\text{PQMC}_{(H)}$, $\text{PMC}_{(H)}^{(n)}$, $\text{PQMC}_{(S)}$, $\text{PQMC}_{(S)}^{(n)}$), the absolute error decreases as k increases, with optimal convergence typically observed at $k = 12$, as highlighted.

$\text{PQMC}_{(S)}$ and $\text{PQMC}_{(H)}$ across all N, demonstrating both lower errors and smoother convergence comparing with *classical* PQMC variants. For instance, at $N = 2,048$, $\text{PQMC}_{(S)}$ yields an error of 0.00054, while $\text{PQMC}_{(H)}$ yields 0.00459 (see Table 7 and Table 8). This reflects the superior uniformity of Sobol sequences for higher dimensions of the matrices. We note the absolute error in the balanced case of $k = 12$ is on the order of 0.0001 for $\text{PQMC}_{(S)}$, whereas it is on the order of 0.001 for $\text{PQMC}_{(H)}$. Although initially larger in error, the *classical* PQMC algorithms enhance robustness, particularly at higher N when the absolute error reaches the order of 0.01 for $\text{PQMC}_{(H)}^{(n)}$ and 0.1 for $\text{PQMC}_{(H)}^{(n)}$. At $N = 2,048$, $\text{PQMC}_{(S)}^{(n)}$ achieves 0.0455 compared to 0.0726 for $\text{PQMC}_{(H)}^{(n)}$, confirming the continued advantage of the Sobol-based PQMC algorithm even in the *classical* configuration.

When we compare the results of the *Almost Optimal* PQMC algorithm with those of the *classical* PQMC algorithm (see Table 7 and Table 8), we observe that for a fixed N and k, the absolute error for the *Almost Optimal* PQMC algorithm is significantly lower than that of the *classical* PQMC algorithm. For instance, at $N = 256$ and $k = 12$, $\text{PQMC}_{(S)}$ yields an error of 0.00028, while $\text{PQMC}_{(S)}^{(n)}$ yields 0.3172 (see Table 8). The same situation occurs when we compare the errors for $\text{PQMC}_{(H)}$ and $\text{PQMC}_{(H)}^{(n)}$ in Table 7. At $N = 256$ and $k = 12$, $\text{PQMC}_{(H)}$ yields an error of 0.00574, while $\text{PQMC}_{(H)}^{(n)}$ yields 0.525. This is because the variance is considerably reduced in constructing $\theta^{(k)}$ and $\theta^{(k-1)}$ for the *Almost Optimal* PQMC algorithm.

Table 8. $\text{PQMC}_{(S)}$ and $\text{PQMC}_{(S)}^{(n)}$ algorithm results for different N and k, applied to a 500×500 symmetric matrix. The absolute errors for the case $\lambda_2/\lambda_1 \approx 0.0363$ (* Highlighted rows represent optimal k values for each N)

N	k	$\lambda_1^{(k)}$ using $\text{PQMC}_{(S)}$	Absolute error $\text{PQMC}_{(S)}$	$\lambda_1^{(k)}$ using $\text{PQMC}_{(S)}^{(n)}$	Absolute error $\text{PQMC}_{(S)}^{(n)}$
128	9	249.987838	0.25756	248.764573	1.4808
	10	250.099240	0.14616	248.990612	1.2548
	11	250.236415	0.00898	249.415659	0.8297
	12	**250.245037**	**0.00036**	**250.777310**	**0.5319**
	13	250.330000	0.08460	250.989461	0.7441
	14	250.407154	0.16175	251.062045	0.8166
256	9	249.744662	0.50074	248.778926	1.4665
	10	249.982894	0.26251	249.094364	1.1510
	11	250.199189	0.04621	249.483392	0.7620
	12	**250.245119**	**0.00028**	**250.562622**	**0.3172**
	13	250.291367	0.04597	250.704555	0.4592
	14	250.369617	0.12422	251.039904	0.7945
512	9	249.638588	0.60681	248.786901	1.4585
	10	249.899316	0.34608	249.301034	0.9444
	11	250.099484	0.14592	249.632500	0.6129
	12	**250.244872**	**0.00053**	**250.486175**	**0.2408**
	13	250.378047	0.13265	250.679597	0.4342
	14	250.475610	0.23021	250.935292	0.6899
1,024	9	249.879108	0.36629	249.156996	1.0884
	10	250.077200	0.16820	249.326758	0.9186
	11	250.140000	0.10540	249.811533	0.4339
	12	**250.244996**	**0.00040**	**250.148126**	**0.0973**
	13	250.351139	0.10574	250.651883	0.4065
	14	250.440000	0.19460	250.854128	0.6087
2,048	9	250.062966	0.18243	249.216417	1.0290
	10	250.147824	0.09758	249.396625	0.8488
	11	250.193656	0.05174	249.937040	0.3084
	12	**250.245937**	**0.00054**	**250.290864**	**0.0455**
	13	250.272562	0.02716	250.555662	0.3103
	14	250.487070	0.24167	250.713019	0.4676

Figure 5 illustrates and compares the obtained numerical results from the *classical* PQMC algorithms and the *Almost Optimal* PQMC variants at the optimal value of $k = 12$. Notably, $\text{PQMC}_{(S)}$ demonstrates superior performance, closely tracking the exact eigenvalue across all sample sizes. The same applies to $\text{PQMC}_{(H)}$. In contrast, $\text{PQMC}_{(S)}^{(n)}$ and $\text{PQMC}_{(H)}^{(n)}$ initially overestimate λ_{max}, but subsequently converge slowly as N increases.

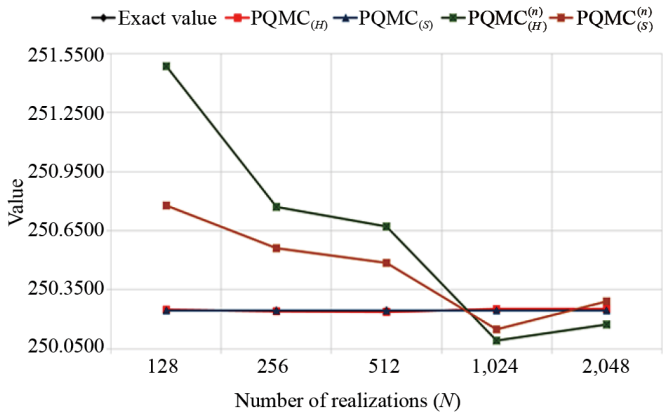

Figure 5. ($PQMC_{(S)}$ and $PQMC_{(H)}$) vs ($PQMC^{(n)}_{(S)}$ and $PQMC^{(n)}_{(H)}$). Comparison of the maximum eigenvalue λ_{max} of the symmetric 500×500 matrix with estimated values $\lambda^{(k)}_{max}$ obtained in the balancing cases, i. e. when ($k = 12,\ N \geq 128$)

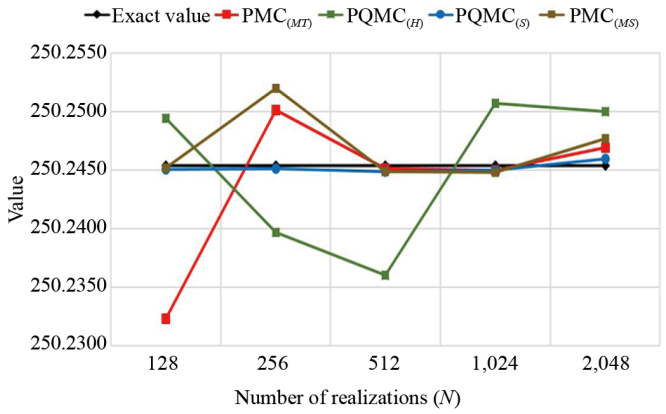

Figure 6. Comparison of of λ_{max} of the symmetric 500×500 matrix with $\lambda^{(k)}_{max}$ in the balancing cases using the *Almost Optimal* PMC algorithm ($k = 9,\ N \geq 128$) and the *Almost Optimal* PQMC algorithm ($k = 12,\ N \geq 128$)

These results underscore that *Almost Optimal* PQMC algorithms outperform *classical* PQMC algorithms regarding computational complexity and accuracy in the case (ii), due to the reduced variance resulting from the choice of the transition density matrix for the Markov chain. Figure 6 compares the *Almost Optimal* PMC algorithm using the MT and MS PRNGs with the *Almost Optimal* PQMC algorithm using Halton (H) and Sobol (S) sequences.

We refine the scale on the Y-axis to observe that all approximate values of $\lambda^{(k)}_{max}$ fall within the interval $\delta < 0.02$ surrounding the maximum eigenvalue λ_{max}. Thus, we can notice differences when comparing the four variants of the *Almost Optimal* PMC and *Almost Optimal* PQMC algorithms.

Indeed, $PMC_{(MT)}$ shows convergence behaviour, with errors reaching a minimum of 3.8×10^{-4} at $N = 1{,}024$ and $k = 9$ (see Table 5), although it demonstrates higher errors for small N. $PMC_{(MS)}$ (see Table 6) demonstrates greater variance and slightly higher errors, although still within acceptable limits (e.g., 6.5×10^{-4} at $N = 1{,}024$ and $k = 9$). Regarding the *Almost Optimal* PQMC algorithms, $PQMC_{(S)}$ stands out again when compared to $PQMC_{(H)}$, as the most precise. It demonstrates consistent performance and minimal deviation from the exact value, as illustrated both visually in Figure 6 and numerically in Table 8. In contrast, $PQMC_{(H)}$, while still improving with increasing N, lags due to its poorer multidimensional uniformity, achieving an error of 4.6×10^{-3} under the same configuration (Table 7). Regarding the optimal balance between stochastic and systematic errors, the test experiments for a dense symmetric matrix with size $n = 500$ indicate that it is achieved when $k = 9$ for $PMC_{(MT)}$ and $k = 12$ for $PQMC_{(S)}$.

The results of case (ii) show that the *Almost Optimal* PMC algorithms provide significant accuracy at precise computational costs. The results also demonstrate that *Almost Optimal* PQMC algorithms based on scrambled Sobol sequences offer significantly greater accuracy than PQMC algorithms based on scrambled Halton sequences for the same computational cost.

5. Conclusions and future work

In this paper, we investigated the stochastic Power Monte Carlo (PMC) and Power Quasi-Monte Carlo (PQMC) methods as effective tools for estimating the maximum eigenvalue of high-dimensional, dense symmetric matrices. Building on the *classical* Power method, we reformulated the problem through stochastic iterations based on Markov chains, which enabled us to systematically incorporate randomness while preserving mathematical rigour.

One of the main contributions of our study is the introduction of *Almost Optimal* initial density vectors and *Almost Optimal* transition density matrices, which significantly reduce variance and improve convergence. Constructing the random variables $\theta^{(k)}$ and $\theta^{(k-1)}$ using these *Almost Optimal* components simplifies their structure and provides a clearer view of the trade-off between systematic errors (caused by truncated Markov chain iterations) and stochastic errors (arising from sampling variability). This refinement led to the development of *Almost Optimal* PMC and PQMC algorithms that balance these errors more effectively than their *classical* counterparts.

Our numerical experiments confirmed the superiority of the proposed algorithms over *classical* stochastic power algorithms, particularly when robust pseudorandom number generators and scrambled low-discrepancy sequences were employed. The choice of transition density matrices and initial probability vectors in the *Almost Optimal* PMC/PQMC algorithms proved especially effective in reducing variance. Using low-discrepancy sequences-most notably scrambled Sobol sequences in the *Almost Optimal* PQMC algorithms-further enhanced convergence rates, especially in higher dimensions. The results also show that optimal error balancing is achieved when parameters k and N are carefully selected, with N typically on the order of the matrix size $N = \mathcal{O}(n)$, Moreover, the optimal parameter k in the Markov chain is generally lower for *Almost Optimal* PMC algorithms than for their PQMC counterparts. These findings suggest that the algorithms remain effective for larger symmetric dense matrices and confirm their practical viability in large-scale settings where precision to the third or fourth decimal digit is required.

In summary, the *Almost Optimal* PMC and PQMC algorithms provide robust stochastic alternatives to *classical* algorithms for estimating maximum eigenvalues. Their computational complexities remain both manageable and scalable, particularly when error balancing is taken into account. While the *Almost Optimal* PQMC algorithm requires careful sequence selection, it delivers superior convergence in high-accuracy regimes for large symmetric matrices. Theoretical insights and numerical results together demonstrate the strength of these algorithms and underscore their potential for broader applications in numerical linear algebra and high-dimensional data analysis.

Despite these advances, several opportunities for future research remain. First, while our analysis concentrated on symmetric matrices, a natural extension is to investigate non-symmetric matrices, including the challenging case where dominant eigenvalues may be complex. This will require adapting stochastic iterative schemes to handle non-real spectra effectively. Second, beyond matrices, the framework invites exploration of eigenpairs in symmetric tensors-a rapidly developing field with significant applications in data analysis, signal processing, and machine learning. Developing stochastic tensor power methods that retain efficiency while managing higher-order interactions would represent a meaningful step forward. Third, applying the proposed algorithms to sparse symmetric matrices of high dimensionality, which frequently arise in scientific computing and machine learning, offers another promising direction. By leveraging sparsity patterns when constructing *Almost Optimal* transition density matrices, computational costs can be further reduced. Finally, implementing these algorithms in parallel environments such as multicore CPUs and GPUs holds considerable potential. Parallelizing Markov chain realizations and exploiting efficient libraries for PRNGs and matrix operations would greatly enhance performance and scalability, making these algorithms suitable for very large-scale problems.

Author contributions

Conceptualization, S.-M. and T.G.; methodology, T.G.; software, S.-M. G.; validation, S.-M. G., T. G., and A.K.; formal analysis, S-M.G., T.G., and A. K.; investigation, S.-M. G.; resources, S.-M. G.; data curation, S.-M. G.; writing-original draft preparation, S.-M. G.; writing-review and editing, T.G. and A. K.; visualization, S.-M. G.; supervision, A. K.; project administration, T.G. All authors have read and agreed to the published version of the manuscript.

Acknowledgement

The authors thank the anonymous reviewers for their valuable comments and suggestions, which greatly improved the quality of the present paper. The work of the author (S.-M. Gurova) was supported by the National Geoinformation Centre (part of the National Roadmap for Research Infrastructures) through grant No. D01-321/30.11.2023, funded by the Ministry of Education and Science of the Republic of Bulgaria and was partially supported by the Centre of Excellence in Informatics and ICT under the Grant No BG16RFPR002-1.014-0018-C01, financed by the Research, Innovation and Digitalisation for Smart Transformation Programme 2021-2027 and co-financed by the European Union. The author's work (T. Gurov) was supported by the Centre of Competence on Digitisation of the economy in an environment of Big Data-second stage, established under Grant No. BG16RFPR002-1.014-0013-C01, financed by the Science and Education for Smart Growth Operational Program and co-financed by the European Union through the European Structural and Investment Funds. The authors gratefully acknowledge that the work was carried out using the infrastructure purchased under the National Roadmap for RI, which was financially coordinated by the Ministry of Education and Science (MES) of the Republic of Bulgaria (Grant No. D01-98/26.06.2025).

Data availability

The data will be available on reasonable request to the corresponding author.

Conflict of interest

The authors declare no conflicts of interest.

References

[1] Chung F. *Spectral Graph Theory*. Providence, R.I: American Mathematical Society; 1997.

[2] Ali M, Ali Almaameri IM, Malik A, Khan F, Rabbani MK, Alamgir. Link adaptation strategy for underwater acoustic sensor networks: A machine learning approach. *Journal of Smart Internet of Things*. 2023; 2023(1): 56-64. Available from: https://doi.org/10.2478/jsiot-2023-0006.

[3] Griffiths D, Schroeter D. *Introduction to Quantum Mechanics*. 3rd ed. Cambridge: Cambridge University Press; 2018.

[4] Manafian J. Optical soliton solutions for Schrödinger type nonlinear evolution equations by the $tan(\phi(\varepsilon)/2)$-expansion method. *Optik*. 2016; 127(10): 4222-4245. Available from: https://doi.org/10.1016/j.ijleo.2016.01.078.

[5] Rometsch M. *Quasi-Monte Carlo Methods in Finance: With Application to Optimal Asset Allocation*. 1st ed. Hamburg, Germany: Diplomica Verlag GmbH; 2008.

[6] Laloux L, Cizeau P, Bouchaud JP, Potters M. Noise dressing of financial correlation matrices. *Physical Review Letters*. 1999; 83(7): 1467-1470. Available from: https://doi.org/10.1103/PhysRevLett.83.1467.

[7] Loperfido N, Shushi T. Solving extended mean-variance models using tensor analysis. *The European Journal of Finance*. 2025: 1-11. Available from: https://doi.org/10.1080/1351847X.2025.2513502.

[8] Kolda T, Bader B. Tensor decompositions and applications. *SIAM Review*. 2009; 51(3): 455-500. Available from: https://doi.org/10.1137/07070111X.
[9] Sturmfels B. Tensors and their eigenvectors. *Notices of the American Mathematical Society*. 2016; 63(6): 604-606. Available from: https://doi.org/10.1090/noti1389.
[10] Loperfido N. Skewness-based projection pursuit: A computational approach. *Computational Statistics & Data Analysis*. 2018; 120: 42-57. Available from: https://doi.org/10.1016/j.csda.2017.11.001.
[11] Golub G, Van Loan C. *Matrix Computations*. 4th ed. Baltimore, MD: Johns Hopkins University Press; 2013.
[12] Isaacson E, Keller H. *Analysis of Numerical Methods*. Mineola, NY, USA: Dover Publications; 1993.
[13] Arnoldi WE. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quarterly of Applied Mathematics*. 1951; 9(1): 17-29. Available from: https://doi.org/10.1090/qam/42792.
[14] L'Ecuyer P. Uniform random number generation. *Annals of Operations Research*. 1994; 53(1): 77-120. Available from: https://doi.org/10.1007/BF02136827.
[15] Sugita H. Monte Carlo method, random number, and pseudorandom number. *MSJ Memories*. 2011; 25: 133. Available from: https://doi.org/10.1142/e027.
[16] Kroese DP, Taimre T, Botev ZI. *Handbook of Monte Carlo Methods*. New York: John Wiley & Sons; 2011.
[17] Sobol IM. *Computational Monte Carlo Methods*. Moscow, Russia: Nauka Publishing House; 1973.
[18] Niederreiter H. Low-discrepancy and low-dispersion sequences. *Journal of Number Theory*. 1988; 30(1): 51-70. Available from: https://doi.org/10.1016/0022-314X(88)90025-X.
[19] Halton J. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numerical Mathematics*. 1960; 2(1): 84-90. Available from: https://doi.org/10.1007/BF01386213.
[20] Halton J. Algorithm 247: Radical-inverse quasi-random point sequence. *Communications of the ACM*. 1964; 7(12): 701-702. Available from: https://doi.org/10.1145/355588.365104.
[21] Atanassov E, Ivanovska S. On the use of Sobol' sequence for high dimensional simulation. In: Groen D, de Mulatier C, Paszynski M, Krzhizhanovskaya V, Dongarra J, Sloot P. (eds.) *Computational Science-ICCS 2022*. Cham: Springer; 2022. p.646-652.
[22] Joe S, Kuo FY. Constructing Sobol sequences with better two-dimensional projections. *SIAM Journal on Scientific Computing*. 2008; 30(5): 2635-2654. Available from: https://doi.org/10.1137/070709359.
[23] Sobol I, Shukhman B. Integration with quasirandom sequences: Numerical experience. *International Journal of Modern Physics C*. 1995; 6(2): 263-275. Available from: https://doi.org/10.1142/S0129183195000204.
[24] Sobol I, Asotsky D, Kreinin A, Kucherenko S. Construction and comparison of high-dimensional Sobol' generator. *Wilmott Magazine*. 2011; 2011(56): 64-79. Available from: https://doi.org/10.1002/wilm.10056.
[25] Sobol I. Uniformly distributed sequences with an additional uniform property. *USSR Computational Mathematics and Mathematical Physics*. 1976; 16(5): 236-242. Available from: https://doi.org/10.1016/0041-5553(76)90154-3.
[26] Zhao W, Wu Y, Chen Y, Ou Y. Reliability sensitivity analysis by the axis orthogonal importance sampling method based on the Box-Muller transformation. *Applied Sciences*. 2022; 12(19): 9860. Available from: https://doi.org/10.3390/app12199860.
[27] Morokoff W, Caflisch R. Quasi-Monte Carlo integration. *Journal of Computational Physics*. 1995; 122(2): 218-230. Available from: https://doi.org/10.1006/jcph.1995.1209.
[28] Caflisch R. Monte Carlo and quasi-Monte Carlo methods. *Acta Numerica*. 1998; 7: 1-49.
[29] Owen A. Scrambling Sobol' and Niederreiter-Xing points. *Journal of Complexity*. 1998; 14(4): 466-489. Available from: https://doi.org/10.1006/jcom.1998.0487.
[30] Wang X, Hickernell FJ. Randomized Halton sequences. *Mathematical and Computer Modelling*. 2000; 32(7-8): 887-899. Available from: https://doi.org/10.1016/S0895-7177(00)00178-3.
[31] Schlier C. On scrambled Halton sequences. *Applied Numerical Mathematics*. 2008; 58(10): 1467-1478. Available from: https://doi.org/10.1016/j.apnum.2007.09.001.
[32] Dimov I, Dimov T, Gurov T. A new iterative Monte Carlo approach for inverse matrix problem. *Journal of Computational and Applied Mathematics*. 1998; 92(1): 15-35. Available from: https://doi.org/10.1016/S0377-0427(98)00043-0.
[33] Kalos M, Whitlock P. *Monte Carlo Methods*. 2nd ed. Germany: Wiley-VCH; 2008.
[34] Alexandrov V, Esquivel-Flores O, Ivanovska S, Karaivanova A. On the preconditioned quasi-Monte Carlo algorithm for matrix computations. In: Lirkov I, Margenov S, Waśniewski J. (eds.) *Large-Scale Scientific Computing*. Lecture Notes in Computer Science. Cham: Springer; 2015. p.163-171.

[35] Mascagni M, Karaivanova A. Matrix computations using quasirandom sequences. In: Vulkov L, Yalamov P, Waśniewski J. (eds.) *Numerical Analysis and Its Applications*. Lecture Notes in Computer Science. Berlin: Springer; 2001. p.552-559.

[36] Wang H, Zheng Z. Randomly shifted lattice rules with importance sampling and applications. *Mathematics*. 2024; 12(5): 630. Available from: https://doi.org/10.3390/math12050630.

[37] Sobol I. On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics*. 1967; 7(4): 86-112. Available from: https://doi.org/10.1016/0041-5553(67)90144-9.

[38] Halton J. Sequential Monte Carlo techniques for the solution of linear systems. *Journal of Scientific Computing*. 1994; 9(2): 213-257. Available from: https://doi.org/10.1007/BF01578388.

[39] Ökten G. Solving linear equations by Monte Carlo simulation. *SIAM Journal on Scientific Computing*. 2005; 27(2): 511-531. Available from: https://doi.org/10.1137/04060500X.

[40] Alexandrov V, et al. On Monte Carlo and Quasi-Monte Carlo for matrix computations. In: Lirkov I, Margenov S. (eds.) *Large-Scale Scientific Computing, Lecture Notes in Computer Science*. Cham: Springer; 2018. p.249-257. Available from: https://doi.org/10.1007/978-3-319-73441-5_26.

[41] Forsythe GE, Leibler RA. Matrix inversion by a Monte Carlo method. *Mathematics of Computation*. 1950; 4(31): 127-129.

[42] Alexandrov V. Efficient parallel Monte Carlo methods for matrix computations. *Mathematics and Computers in Simulation*. 1998; 47(2-5): 113-122. Available from: https://doi.org/10.1016/S0378-4754(98)00097-4.

[43] Alexandrov A, Esquivel-Flores OA. Towards Monte Carlo preconditioning approach and hybrid Monte Carlo algorithms for matrix computations. *Computers & Mathematics with Applications*. 2015; 70(11): 2709-2718. Available from: https://doi.org/10.1016/j.camwa.2015.08.035.

[44] Dimov I, Philippe B, Karaivanova A, Weihrauch C. Robustness and applicability of Markov chain Monte Carlo algorithms for eigenvalue problems. *Applied Mathematical Modelling*. 2008; 32(8): 1511-1529. Available from: https://doi.org/10.1016/j.apm.2007.04.012.

[45] Dimov I. *Monte Carlo Methods for Applied Scientists*. Singapore: World Scientific; 2007.

[46] Dimov I, Karaivanova A. A power method with Monte Carlo iterations. In: Dimov I, Lirkov I, Margenov S, Waśniewski J, Yalamov P. (eds.) *Recent Advances in Numerical Methods and Applications II*. Singapore: World Scientific Publishing Co Pte Ltd; 1999. p.239-247.

[47] Mascagni M, Karaivanova A. A parallel quasi-Monte Carlo method for computing extremal eigenvalues. In: Fang KT, Hickernell FJ, Niederreiter H. (eds.) *Monte Carlo and Quasi-Monte Carlo Methods 2000*. Berlin: Springer; 2002. p.369-380. Available from: https://doi.org/10.1007/978-3-642-56046-0_25.

[48] Matsumoto M, Nishimura T. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*. 1998; 8(1): 3-30. Available from: https://doi.org/10.1145/272991.272995.

[49] Padányi V, Herendi T. Generalized middle-square method. *Annals of Mathematics and Computer Science*. 2022; 56: 95-108. Available from: https://doi.org/10.33039/ami.2022.12.003.

[50] Matoušek J. On the L_2-discrepancy for anchored boxes. *Journal of Complexity*. 1998; 14(4): 527-556. Available from: https://doi.org/10.1006/jcom.1998.0489.

[51] Owen A. Randomly permuted (t, m, s)-nets and (t, s)-sequences. In: Niederreiter H, Shiue PJS. (eds.) *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*. New York: Springer; 1995. p.299-317.

[52] Boisvert R, Kocis L, Whiten W. Computational investigations of low-discrepancy sequences. *ACM Transactions on Mathematical Software*. 1997; 23(2): 266-294. Available from: https://doi.org/10.1145/264029.264064.

[53] von Neumann J. Various techniques used in connection with random digits. *Applied Math Series*. 1951; 12: 36-38.

[54] Saito M, Matsumoto M. SIMD-oriented fast Mersenne twister: A 128-bit pseudorandom number generator. In: Keller A, Heinrich S, Niederreiter H. (eds.) *Monte Carlo and Quasi-Monte Carlo Methods 2006*. Berlin: Springer; 2008. p.607-622.