axioms

MDPI

*Article*

# Cutting-Edge Monte Carlo Framework: Novel "Walk on Equations" Algorithm for Linear Algebraic Systems

**Venelin Todorov [1,2] and Ivan Dimov [2,*]**

[1] Institute of Mathematics and Informatics, Bulgarian Academy of Sciences, Acad. G. Bonchev Str. Bl. 8, 1113 Sofia, Bulgaria
[2] Institute of Information and Communication Technologies, Bulgarian Academy of Sciences, Acad. G. Bonchev Str. Bl. 25A, 1113 Sofia, Bulgaria
* Correspondence: ivdimov@bas.bg

**Abstract:** In this paper, we introduce the "Walk on Equations" (WE) Monte Carlo algorithm, a novel approach for solving linear algebraic systems. This algorithm shares similarities with the recently developed WE MC method by Ivan Dimov, Sylvain Maire, and Jean Michel Sellier. This method is particularly effective for large matrices, both real- and complex-valued, and shows significant improvements over traditional methods. Our comprehensive comparison with the Gauss–Seidel method highlights the WE algorithm's superior performance, especially in reducing relative errors within fewer iterations. We also introduce a unique dominancy number, which plays a crucial role in the algorithm's efficiency. A pivotal outcome of our research is the convergence theorem we established for the WE algorithm, demonstrating its optimized performance through a balanced iteration matrix. Furthermore, we incorporated a sequential Monte Carlo method, enhancing the algorithm's efficacy. The most-notable application of our algorithm is in solving a large system derived from a finite-element approximation in constructive mechanics, specifically for a beam structure problem. Our findings reveal that the proposed WE Monte Carlo algorithm, especially when combined with sequential MC, converges significantly faster than well-known deterministic iterative methods such as the Jacobi method. This enhanced convergence is more pronounced in larger matrices. Additionally, our comparative analysis with the preconditioned conjugate gradient (PCG) method shows that the WE MC method can outperform traditional methods for certain matrices. The introduction of a new random variable as an unbiased estimator of the solution vector and the analysis of the relative stochastic error structure further illustrate the potential of our novel algorithm in computational mathematics.

**Keywords:** Monte Carlo methods; Markov chain; linear algebraic systems

**MSC:** 60J22; 62P12; 65C05; 68W20



**Citation:** Todorov, V.; Dimov, I. Cutting-Edge Monte Carlo Framework: Novel "Walk on Equations" Algorithm for Linear Algebraic Systems. *Axioms* **2024**, *13*, 53. https://doi.org/10.3390/axioms13010053

Academic Editors: Milan Gnjatović and Nemanja Maček

Received: 30 November 2023
Revised: 10 January 2024
Accepted: 11 January 2024
Published: 15 January 2024



## 1. Introduction

Linear systems play a pivotal role in various fields of mathematics, science, engineering, and technology, showcasing their fundamental significance [1–5]. The study and solution of linear systems of algebraic equations provide a versatile framework for modeling and analyzing a wide array of real-world phenomena [6–14]. These systems offer a structured approach to represent relationships and dependencies among variables, enabling the formulation of mathematical models for diverse applications, including physics, economics, biology, and computer science. Linear systems also serve as the foundation for understanding more-complex mathematical structures, paving the way for advanced techniques and methodologies in numerical analysis, optimization, and data analysis. The ability to efficiently solve linear systems is crucial for addressing practical problems, making them an indispensable tool in the toolkit of researchers, scientists, and engineers

across various disciplines. Whether in simulating physical systems, optimizing resource allocation, or processing data, the significance of linear systems lies in their capacity to provide a systematic and powerful framework for problem-solving and decision-making.

Beyond their foundational role in mathematics, linear systems offer a practical and indispensable tool for solving real-world problems [15–18]. The linear nature of these systems simplifies their mathematical representation, making them particularly amenable to analytical and computational methods. In engineering, for instance, linear systems are pervasive in control theory, signal processing, and structural analysis. The ability to solve linear systems efficiently contributes to the design and optimization of systems ranging from electronic circuits to mechanical structures. Moreover, in economics and finance, linear models are frequently employed to analyze relationships between variables, aiding in decision-making processes. The significance of linear systems extends to scientific research, where they are utilized for data analysis, image processing, and simulations. In essence, the versatility and applicability of linear systems underscore their significance across a broad spectrum of disciplines, emphasizing their role as a cornerstone in the theoretical understanding and practical application of mathematical concepts in the real world.

The significance of linear systems transcends their mere existence as mathematical constructs; rather, they embody a fundamental and pervasive concept with profound implications across various domains of human inquiry. At its core, a linear system encapsulates relationships, dependencies, and interactions within a structured framework, providing a mathematical language to express and decipher complex phenomena. In physics, the laws governing many natural phenomena exhibit linearity, allowing scientists to model and predict behavior. In the intricate world of engineering, linear systems are the backbone of control theory, enabling the design of stable and efficient systems, whether in aerospace engineering or telecommunications.

Moreover, linear systems [19–21] serve as a gateway to understanding more-sophisticated mathematical structures. The study of linear algebra and the solutions to linear systems lay the groundwork for advanced mathematical techniques that underpin numerical analysis, optimization, and statistical modeling. In essence, linear systems act as a bridge between theoretical abstraction and real-world problem-solving, facilitating a deeper comprehension of intricate mathematical concepts. The practical implications of efficiently solving linear systems are profound. From the optimization of resource allocation in logistics to the simulation of complex physical processes, the ability to navigate and manipulate linear systems empowers researchers and engineers to tackle challenges of increasing complexity. In economics, linear models elucidate intricate relationships within financial systems, offering insights crucial for informed decision-making.

Linear Algebraic Equations (LAEs) [22–25] establish the fundamental scaffolding for an array of scientific and engineering pursuits, effectively translating abstract concepts into tangible, real-world applications. Whether navigating intricate simulations in the realm of computational fluid dynamics or delving into intricate optimizations within the field of structural engineering, LAEs provide a mathematical prism that allows us to not only comprehend but also address a diverse range of complex challenges. As computational power continues to burgeon and the intricacy of problems intensifies, attention has gravitated towards large-scale LAEs. These expansive systems not only push the boundaries of analytical methods but also necessitate a computational infrastructure capable of grappling with their sheer magnitude. Within these vast systems lie intricate interrelationships and voluminous datasets, amplifying the critical nature and complexity of their resolution. Successfully addressing these extensive systems often hinges on the application of sophisticated stochastic techniques, highlighting the necessity for advanced and nuanced approaches [26,27].

LAEs serve as a foundational framework in mathematics and its applications, enabling the modeling and analysis of diverse systems and phenomena. The widespread use of LAEs underscores their significance as a foundational and unifying mathematical framework. As computational power continues to advance, the application of linear algebra in

interdisciplinary research and problem-solving is likely to expand, further cementing its role as an indispensable tool in both theoretical and applied mathematics. Their broad applicability and the development of specialized techniques for their solution make them a central topic in both theoretical and applied mathematics. The application of linear algebraic equations extends across a wide array of fields, demonstrating their versatility in addressing complex problems and providing essential tools for understanding, modeling, and solving diverse systems and phenomena. Here are some broader perspectives on the applications of LAEs [28–32].

In engineering disciplines, LAEs are fundamental to the analysis and design of structures, circuits, and mechanical systems. They play a pivotal role in solving problems related to fluid dynamics, heat transfer, and electrical networks. In physics, linear algebra is essential for describing quantum mechanics, classical mechanics, and the behavior of physical systems. Linear algebra is at the core of many algorithms and techniques in computer science. From image processing and computer graphics to machine learning and artificial intelligence, linear algebraic equations are used for tasks like image transformation, data representation, and solving systems of equations arising in algorithmic processes. LAEs are applied in economic modeling and financial analysis. Systems of linear equations are used to model economic relationships, optimize resource allocation, and analyze market trends. Techniques such as input-output analysis and linear programming rely on linear algebraic methods. Linear algebra is a fundamental tool in statistics and data analysis. Methods such as linear regression, principal component analysis, and singular value decomposition involve solving and manipulating systems of linear equations. These techniques are crucial for extracting meaningful information from large datasets. LAEs play a central role in optimization problems and control systems engineering. Techniques like linear programming and control theory heavily rely on linear algebraic methods to optimize processes, design controllers, and ensure stability in dynamic systems. Cryptography, the science of secure communication, employs linear algebra in areas such as coding theory and the design of encryption algorithms. Techniques like error-correcting codes and cryptographic protocols often involve solving linear algebraic equations to ensure the security and integrity of transmitted information. LAEs find applications in biomedical imaging, genetics, and bioinformatics. Techniques like image reconstruction in medical imaging, genetic mapping, and the analysis of biological networks involve solving linear systems to interpret and understand complex biological phenomena. Environmental scientists use LAEs to model and analyze environmental systems. This includes studying the flow of pollutants in ecosystems, analyzing climate models, and optimizing resource management strategies to address environmental challenges.

Monte Carlo (MC) algorithms [33–35] for LAPs leverage probabilistic sampling techniques to address challenges associated with large-scale linear systems. These algorithms are particularly useful when traditional methods become computationally expensive or impractical. MC methods are often incorporated into randomized numerical linear algebra algorithms. These methods introduce randomness into the computation, allowing for efficient approximations of matrix factorizations and solutions to linear systems. These techniques are useful for tasks like low-rank approximation, singular value decomposition, and solving linear systems. MC algorithms for LAPs can be applied to solve systems of linear equations. Instead of directly computing the solution, these algorithms use random sampling to approximate the solution or components of the solution vector. This is especially advantageous when dealing with large and sparse matrices. MC integration techniques are adapted for LAPs, where the goal is to approximate integrals involving high-dimensional vectors or matrices. This approach involves random sampling to estimate expectations or integrals, offering a stochastic alternative to deterministic methods. In some applications, matrices involved in linear algebraic problems are too large to be explicitly formed or stored. Matrix-free MC methods address this challenge by performing matrix-vector products on-the-fly, utilizing random samples to compute these products without explicitly representing the entire matrix. MC algorithms provide a stochastic

approximation framework for solving LAPs. By iteratively updating the solution based on random samples, these algorithms converge to an approximate solution over multiple iterations. This approach is particularly relevant for large-scale problems where deterministic methods may be impractical. MC algorithms naturally lend themselves to estimating errors and confidence intervals in the computed solutions. By generating multiple random samples and observing the variability in the results, these algorithms provide insights into the uncertainty associated with the computed solution. MC algorithms for LAPs are well-suited for parallel and distributed computing environments. The inherent randomness and independence of MC samples make it possible to perform computations concurrently, improving the scalability of these algorithms for large-scale problems. MC techniques for LAPs find applications in data science and machine learning, especially in tasks involving large datasets and high-dimensional spaces. These algorithms are used for tasks such as approximating covariance matrices, solving linear regression problems, and conducting randomized dimensionality reduction. Some MC algorithms for LAEs incorporate adaptive sampling strategies, where the sampling distribution is adjusted based on the observed outcomes. This adaptability allows the algorithm to focus computational efforts on the most influential components of the problem. MC algorithms for LAPs provide a powerful and flexible approach to handling large-scale computations. Their stochastic nature and adaptability make them well-suited for addressing challenges in diverse fields, ranging from scientific computing to data-driven applications. As computational resources continue to advance, the role of MC methods in addressing complex LAPs is likely to expand further.

A comprehensive overview of these algorithms is provided in [36]. The renowned Power method [37] furnishes an evaluation for the main eigenvalue $\lambda_1$, employing the well-known Rayleigh quotient In [38], the authors delve into matrix powers' bilinear forms, employing them to devise their solution. Both theoretical and experimental examinations delve into the algorithm's robustness. It is elucidated that as perturbations in the entries of perfectly balanced matrices increase, both error and variance witness a corresponding escalation. Particularly, smaller matrices exhibit a heightened variance. As the power of $A$ increases, a rise in the relative error becomes apparent.

Iterative Monte Carlo (IMC) methods [36,39,40] are a class of computational techniques that utilize iterative processes inspired by Monte Carlo (MC) simulations to solve complex problems, particularly those involving probabilistic or random elements. These methods are iterative in nature, meaning that they involve repeated cycles of computation to refine an estimate or solution. IMC methods are powerful tools that provide flexible and efficient solutions to a wide range of problems across different disciplines. Their ability to handle complex, high-dimensional problems makes them particularly valuable in situations where traditional analytical methods may be impractical or infeasible. As computational resources continue to advance, the application of Iterative Monte Carlo methods is likely to expand further, contributing to advancements in scientific research, optimization, and data-driven decision-making. IMC methods designed to solve LAEs (LAEs) can be conceptualized as truncated Markov chains [41]:

$$T = \{\alpha_{t_0} \to \alpha_{t_1} \to \alpha_{t_2} \to \cdots \to \alpha_{t_k}\}, \tag{1}$$

where each state $\alpha_{t_q}$ for $q = 1, \ldots, i$ represents an absorbing state in the chain.

To devise an algorithm for assessing the minimal eigenvalue by modulus, denoted as $\lambda_n$, a matrix polynomial $p_k(A) = \sum_{k=0}^{\infty} q^k C_{m+k-1}^k A^k$ is essential. Here, $C_{m+k-1}^k$ represents binomial coefficients, and $q$ serves as an acceleration parameter [23,42,43]. This approach aligns with a discrete analog of the resolvent analytical continuation method utilized in [44]. There are instances where the polynomial converges to the resolvent matrix [25,36,42]. Notably, implementing an acceleration parameter taking into account the resolvent is one avenue for reducing computational complexity. Alternatively, employing a variance reduction technique [45] serves to achieve the required solution approximation with fewer operations. The variance reduction technique proposed in [45] for eigenvalue calculations in particle transport utilizes MC approximations of fluxes. In [46], an unbiased estimator

for solving linear algebraic systems (LAS) is introduced. This estimator is adept at finding specific components of the solution, accompanied by comprehensive results elucidating its quality and properties. The author leverages this estimator to establish error bounds and construct confidence intervals for the solution components. Furthermore, ref. [47] introduces a MC method for matrix inversion, rooted in the solution of simultaneous LAEs. In our subsequent exploration, insights from both [36,47] will be incorporated to demonstrate how the proposed algorithm can effectively approximate the inverse of a matrix.

## 2. Problem Formulation

Examine the subsequent LAP:

- Computing elements of the solution vector, which could involve a subset of components or all components.
- Assessing linear functionals of the solution.
- Performing matrix inversion.
- Determining the extremal eigenvalues.

Let $A$ and $B$ are matrices of size $n \times n$, i.e., $A, B \in \mathbb{R}^{n\times n}$.

$$A = \{a_{ij}\}_{i,j=1}^{n} = (a_1, \ldots, a_i, \ldots, a_n)^t,$$

where $a_i = (a_{i1}, \ldots, a_{in})$, $i = 1, \ldots, n$. Norms of vectors ($l_1$-norm): $\| b \| = \| b \|_1 = \sum_{i=1}^{n} |b_i|$, $\| a_i \| = \| a_i \|_1 = \sum_{j=1}^{n} |a_{ij}|$ and matrices $\| A \| = \| A \|_1 = \max_j \sum_{i=1}^{n} |a_{ij}|$. Consider a matrix $A \in \mathbb{R}^{n\times n}$ and a vector $b = (b_1, \ldots, b_n)^t \in \mathbb{R}^{n\times 1}$. The matrix A can be treated as a linear operator $A[\mathbb{R}^n \to \mathbb{R}^n]$; the linear transformation $Ab \in \mathbb{R}^{n\times 1}$ determines a new vector in $\mathbb{R}^{n\times 1}$.

Suppose that this LAE is given:

$$Bx = f, \quad B \in \mathbb{R}^{n\times n}; \quad f, x \in \mathbb{R}^{n\times 1}. \tag{2}$$

The inverse matrix task is the same as solving $n$-times the task (2), i.e.,

$$Bg_j = f_j, \quad j = 1, \ldots, n \tag{3}$$

$f_j \equiv e_j \equiv (0, \ldots, 0, \underbrace{1}_{j}, 0, \ldots, 0)$ $g_j \equiv (g_{j1}, g_{j2}, \ldots, g_{jn})^t$ is the $j$-th column of $G = B^{-1}$.

Let the matrix $A = \{a_{ij}\}_{ij=1}^{n}$, such that

$$A = I - DB, \tag{4}$$

$D$ is a diagonal matrix $D = diag(d_1, \ldots, d_n)$ $d_i = \frac{\gamma}{b_{ii}}$, $i = 1, \ldots, n$, $\gamma \in (0, 1]$ is a parameter for accelerating the convergence. The system (2) can be expressed as:

$$x = Ax + b, \tag{5}$$

where $B = Df$.

Let's assume that the matrix $B$ exhibits diagonal dominance. However, it's worth noting that the algorithms presented are applicable to a broader class of matrices, as demonstrated later. Specifically, if $B$ is diagonally dominant, certain conditions on the matrix' elements of $A$ must be met:

$$\sum_{j=1}^{n} |a_{ij}| \leq 1 \qquad i = 1, \ldots, n. \tag{6}$$

### 2.1. Problem Settings

We shall consider the following problems.

**Problem 1 (Evaluating the inner product).**

$$J(x) = (v, x) = \sum_{i=1}^{n} v_i x_i$$

*of the solution* $x \in \mathbb{R}^{n\times 1}$ *of the LAS*

$$Bx = f,$$

*where* $B = \{b_{ij}\}_{i,j=1}^{n} \in \mathbb{R}^{n\times n}$ *is a given matrix;* $f = (f_1, \ldots, f_n)^t \in \mathbb{R}^{n\times 1}$ *and* $v = (v_1, \ldots, v_n)^t \in \mathbb{R}^{n\times 1}$ *are given vectors.*

It is feasible to select a non-singular matrix $M \in \mathbb{R}^{n\times n}$ for which $MB = I - A$, $I \in \mathbb{R}^{n\times n}$ is the identity matrix and $Mf = b$, $b \in \mathbb{R}^{n\times 1}$.

Then

$$x = Ax + b.$$

It is taken into consideration that

$$(i) \quad \begin{cases} 1. & \text{The matrices } M \text{ and } A \text{ are both non-singular;} \\ 2. & |\lambda(A)| < 1 \text{ for all eigenvalues } \lambda(A) \text{ of } A, \end{cases}$$

so, every value of $\lambda(A)$ for which

$$x = Ax + b. \tag{7}$$

is met. If the criteria outlined in $(i)$ are satisfied, then a *stationary linear iterative algorithm* can be employed:

$$x_k = Ax_{k-1} + b, \quad k = 1, 2, \ldots \tag{8}$$

and the solution $x$ could be expressed as $x = \sum_{k=0}^{\infty} A^k b = b + Ab + A^2 b + A^3 b + \ldots$ known as von Neumann series.

**Problem 2 (Assessing all components $x_i$, $i = 1, \ldots n$ of the solution vector).**

Here, we express the Equation (7) with $A = I - MB$, and likewise, $b = Mf$.

**Problem 3 (Inverting of matrices).** *This consists of computation:* $G = B^{-1}$, *where* $B \in \mathbb{R}^{n\times n}$ *a specified real matrix.*

*It is presupposed that*

$$(ii) \quad \begin{cases} 1. & \textit{The matrix } B \textit{ is non-singular;} \\ 2. & ||\lambda(B)| - 1| < 1 \textit{ for all eigenvalues } \lambda(B) \textit{ of } B. \end{cases}$$

Here the subsequent *iterative* matrix: $A = I - B$ can be defined. When (ii) are met, $G = B^{-1}$ can be expressed as $G = \sum_{i=0}^{\infty} A^i$.

### *2.2. Almost Optimal Monte Carlo Algorithm*

We will employ the algorithm known as the MAO algorithm. It is an algorithm that defines Almost Optimal Method (Method Almost Optimal, (MAO)) according the definition given by I. Dimov in [48].

Take an initial density vector $p = p_i i = 1^n \in \mathbb{R}^n$, where $p_i \geq 0, i = 1, \ldots, n$ and $\sum i = 1^n p_i = 1$. Also, contemplate a matrix representing transition density $P = p_{ij} i, j = 1^n \in \mathbb{R}^{n\times n}$, where $pij \geq 0$, $i, j = 1, \ldots, n$ and $\sum_{j=1}^{n} p_{ij} = 1$, for any $i = 1, \ldots, n$. Establish sets of *permissible* densities $\mathcal{P}_b$ and $\mathcal{P}_A$.

**Definition 1.** *The initial density vector* $p = \{p_i\}_{i=1}^{n}$ *is called permissible to the vector* $b = \{b_i\}_{i=1}^{n} \in \mathbb{R}^n$ *, i.e.,* $p \in \mathcal{P}_b$, *if*

$$\begin{cases} p_{\alpha_s} > 0 & v_{\alpha_s} \neq 0 \\ p_{\alpha_s} = 0 & v_{\alpha_s} = 0. \end{cases} \tag{9}$$

*The transition density matrix* $P = \{p_{ij}\}_{i,j=1}^{n}$ *is called permissible to the matrix* $A = \{a_{ij}\}_{i,j=1}^{n}$, *i.e.*, $P \in \mathcal{P}_A$, *if*

$$\begin{cases} p_{\alpha_{s-1},\alpha_s} > 0 & a_{\alpha_{s-1},\alpha_s} \neq 0 \\ p_{\alpha_{s-1},\alpha_s} = 0 & a_{\alpha_{s-1},\alpha_s} = 0. \end{cases} \tag{10}$$

In this study, we will focus on densities that adhere to the defined criteria. This strategy ensures that the generated random trajectories, designed to address the problems at hand, consistently avoid encountering elements with zero values in the matrix. This method not only reduces the computational intricacies of the algorithms but also proves advantageous, particularly when handling large sparse matrices.

Suppose we have a Markov chain:

$$T = \alpha_0 \to \alpha_1 \to \dots \alpha_k \to \dots \tag{11}$$

with $n$ states. The random trajectory (chain) $T_k$ of length $k$ originating from the initial state $\alpha_0$, is specified as:

$$T_k = \alpha_0 \to \alpha_1 \to \dots \alpha_j \to \dots \alpha_k. \tag{12}$$

Here $\alpha_j$ refers to the state selected, indicating $j = 1, \dots, n$.

Assume that

$$P(\alpha_0 = \alpha) = p_\alpha \text{ and } P(\alpha_j = \beta | \alpha_{j-1} = \alpha) = p_{\alpha\beta}, \tag{13}$$

where $p_\alpha$ represents the probability of the chain initiating in state $\alpha$, and $p_{\alpha\beta}$ is the transition probability to state $\beta$ after being in state $\alpha$. These probabilities $p_{\alpha\beta}$ collectively form a transition matrix $P$.

So our MAO WE algorithm, is configured with a distinctive selection of absorbing states, entails the presence of $n+1$ states denoted as $1, \dots, n, n+1$.

The transition density matrix is designed in a way that $\sum_{\beta=1}^{n} p_{\alpha\beta} \leq 1$, and the probability of absorption at each stage, excluding the initial one, is

$p_{\alpha,n+1} = p_\alpha = 1 - \sum_{\beta=1}^{n} p_{\alpha\beta}, \alpha = 1, \dots, n.$

### *2.3. Absorbing States Monte Carlo Algorithm*

Rather than dealing with the finite random trajectory $T_i$, we examine an infinite trajectory characterized by a state coordinate $\delta_i (i = 1, 2, \dots)$. Suppose $\delta_i = 0$ if the trajectory is broken (absorbed) and $\delta_i = 1$ otherwise. Let

$$\Delta_i = \delta_0 \times \delta_1 \times \dots \times \delta_i.$$

Thus, $\Delta_i$ equals 1 until the trajectory's first termination, and thereafter, $\Delta_q$ becomes 0.

It can be demonstrated that, given conditions (i) and (ii), these equalities hold:

$$E\{Q_i f_{k_i}\} = (h, A^i b), \quad i = 1, 2, \dots;$$

$$E\{\sum_{i=0}^{N} \Delta_i Q_i b_{k_i}\} = (h, x), \quad (P1),$$

$$E\{\sum_{i|k_i=r'} \Delta_i Q_i\} = c_{rr'}, \quad (P3).$$

## 3. Probabilistic Representation for the WE Algorithm

### *3.1. Simple Examples*

- The positive case:

$a_{ij} \geq 0,\ i,j = 1,\ldots,n$. We also assume that $\sum_{j=1}^{n} a_{ij} \leq 1$.

Consider a system of two equations:

$$x_1 = \frac{1}{2}x_1 + \frac{1}{4}x_2 + 1,\ \ x_2 = \frac{1}{3}x_1 + \frac{1}{3}x_2 + 2$$

with unknowns $x_1$ and $x_2$. Clearly, we have $x_1 = 1 + E(X)$, where

$$P(X = x_1) = \frac{1}{2}, P(X = x_2) = \frac{1}{4}, P(X = 0) = \frac{1}{4}$$

and $x_2 = 2 + E(Y)$, where

$$P(Y = x_1) = \frac{1}{3}, P(Y = x_2) = \frac{1}{3}, P(Y = 0) = \frac{1}{3}.$$

To estimate $x_1$, we start with an initial score of 1 in our walk. Subsequently, we either terminate with a probability of $\frac{1}{4}$ since we already know the value of $X$ (which is zero), or we proceed to approximate either $x_1$ or $x_2$ again with probabilities $\frac{1}{2}$ or $\frac{1}{4}$. If the walk continues and the goal is to approximate $x_2$, we add 2 to the current score. We then stop with a probability of $\frac{1}{3}$ or continue to approximate $x_1$ or $x_2$ again with a probability of $\frac{1}{3}$. This procedure persists until either of the random variables $X$ or $Y$ reaches zero. The score increases incrementally during the walk as outlined. The score functions as an impartial estimator of $x_1$, and by averaging scores obtained from independent walks, we derive the Monte Carlo estimation of $x_1$.

This algorithm shares a striking similarity with those utilized in calculating Feynman-Kac boundary value problems' representation, employing walks that are either discrete or continuous such as the discrete grids' walk or the spheres method' walk. At walk' each step, the current position solution $x$ corresponds to the anticipated value of the solution on a discrete or continuous set, potentially augmented by a source term. The pivotal concept is the double randomization principle, asserting that the expected solution on this set can be substituted by the solution at a randomly selected point, following the appropriate distribution.

### *3.2. Probabilistic Representation for Real Matrices*

Consider a system $x = Ax + b$, where $\varrho(A) < 1$, and $\sum_{j=1}^{n} |a_{i,j}| \leq 1,\ \forall 1 \leq i \leq n$. Define a Markov chain $T_k$ with $n+1$ states $\alpha_1,\ldots,\alpha_n, n+1$, with

$$P(\alpha_{k+1} = j/\alpha_k = i) = |a_{i,j}|$$

if $i \neq n+1$ and

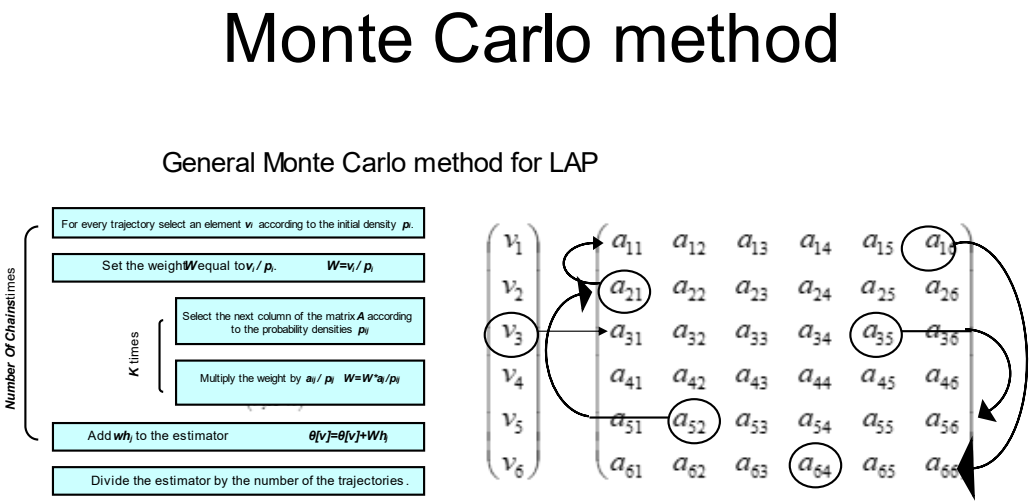
$$P(\alpha_{k+1} = n+1/\alpha_k = n+1) = 1.$$

Let $c$ be a vector defined as $c(i) = b(i)$ if $1 \leq i \leq n$ and $c(n+1) = 0$. Represent $\tau = (\alpha_0, \alpha_1, \ldots, \alpha_k, n+1)$ as a random trajectory that initiates at the initial state $\alpha_0 < n+1$ and traverses through $(\alpha_1, \ldots, \alpha_k)$ until reaching the absorbing state $\alpha_{k+1} = n+1$. The probability of following the trajectory $\tau$ is given by $P(\tau) = p_{\alpha_0} p_{\alpha_0\alpha_1}, \ldots p_{\alpha_{k-1,k}\alpha_k} p_{\alpha_k}$. Use the MAO algorithm: $p = \{p_\alpha\}_{\alpha=1}^{n}$ and for the transition density matrix $P = \{p_{\alpha\beta}\}_{\alpha,\beta=1}^{n}$. Define the weights:

$$Q_m = Q_{m-1}\frac{a_{\alpha_{m-1},\alpha_m}}{p_{\alpha_{m-1},\alpha_m}},\ \ m = 1,\ldots,k,\ \ Q_0 = \frac{c_{\alpha_0}}{p_{\alpha_0}}. \tag{14}$$

The estimator $\theta_\alpha(\tau)$ can be expressed as $\theta_\alpha(\tau) = c_\alpha + Q_k \frac{a_{\alpha_k \alpha}}{p_{\alpha_k}}$, $\alpha = 1, \ldots n$ with $P(\tau) = p_{\alpha_0} p_{\alpha_0 \alpha_1}, \ldots p_{\alpha_{k-1} \alpha_k} p_{\alpha_k}$.

An example of the general MC method for LAP is shown on Figure 1. For the new algorithm the following theorem can be proved.




$$W = \frac{v_3}{p_3} \frac{a_{35}}{p_{35}} \frac{a_{52}}{p_{52}} \frac{a_{21}}{p_{21}} \frac{a_{16}}{p_{16}} \frac{a_{64}}{p_{64}}$$

**Figure 1.** MC-scheme explanation.

**Theorem 1.** *The random variable $\theta_\alpha(\tau)$ is an unbiased estimator of $x_\alpha$, i.e.,*

$$E\{\theta_\alpha(\tau)\} = x_\alpha.$$

**Proof.**

$$\begin{aligned}
E\{\theta_\alpha(\tau)\} &= \sum_{\tau=(\alpha_0,\ldots,\alpha_k,n+1)} \theta_\alpha(\tau) P(\tau) \\
&= \sum_{\tau=(\alpha_0,\ldots,\alpha_k,n+1)} \left( c_\alpha + Q_k(\tau) \frac{a_{\alpha_k \alpha}}{p_{\alpha_k}} \right) P(\tau) \\
&= c_\alpha + \sum_{\tau=(\alpha_0,\ldots,\alpha_k,n+1)} Q_k(\tau) \frac{a_{\alpha_k \alpha}}{p_{\alpha_k}} P(\tau) \\
&= c_\alpha + \sum_{\tau=(\alpha_0,\ldots,\alpha_k,n+1)} \frac{c_{\alpha_0}}{p_{\alpha_0}} \frac{a_{\alpha_0 \alpha_1} \ldots a_{\alpha_{k-1} \alpha_k}}{p_{\alpha_0 \alpha_1} \ldots p_{\alpha_{k-1} \alpha_k}} \frac{a_{\alpha_k \alpha}}{p_{\alpha_k \alpha}} p_{\alpha_0} \\
&\times \; p_{\alpha_0 \alpha_1} \ldots p_{\alpha_{k-1} \alpha_k} p_{\alpha_k} \\
&= c_\alpha + \sum_{k=0}^{\infty} \sum_{\alpha_0=1}^{n} \ldots \sum_{\alpha_k=1}^{n} a_{\alpha_k \alpha} a_{\alpha_{k-1} \alpha_k} \ldots a_{\alpha_0 \alpha_1} c_{\alpha_0} \\
&= c_\alpha + (Ac)_\alpha + (A^2 c)_\alpha + \ldots = \left( \sum_{i=0}^{\infty} \left( A^i c \right) \right)_\alpha .
\end{aligned} \tag{15}$$

Since $\varrho(A) < 1$, the last sum is finite and

$$\left( \sum_{i=0}^{\infty} \left( A^i c \right) \right)_\alpha = x_\alpha, \quad \alpha = 1, \ldots, n.$$

Thus,

$$E\{\theta_\alpha(\tau)\} = x_\alpha.$$

□

The conjugate to $\theta_\alpha(\tau)$ random variable $\theta^*_\alpha(\tau) = \sum_{i=0}^{k} Q_i \frac{a_{\alpha_i \alpha}}{p_{\alpha_i}} c_\alpha$ has the same mean value, i.e., $x_\alpha$. This could be established by the same technique used above.

Suppose it is possible to calculate $N$ values of $\theta_\alpha$, namely $\theta_{\alpha,i}$, $i = 1, \ldots, N$. Taking into account $\bar{\theta}_{\alpha,N} = \frac{1}{N} \sum_{i=1}^{N} \theta_{\alpha,i}$ as a MC approximation of the component $x_\alpha$ of the solution. For the random variable $\bar{\theta}_{\alpha,N}$ one may prove [1]:

**Theorem 2.** *The random variable $\bar{\theta}_{\alpha,N}$ is an unbiased estimator of $x_\alpha$, i.e.,*

$$E\{\bar{\theta}_{\alpha,N}\} = x_\alpha \tag{16}$$

*and the variance $D\{\bar{\theta}_{\alpha,N}\}$ vanishes when $N$ goes to infinity, i.e.,*

$$\lim_{N \to \infty} D\{\bar{\theta}_{\alpha,N}\} = 0. \tag{17}$$

**Proof.** The equation can be readily demonstrated by employing the mean value's properties. Similarly, the validity of (22) can be established using the variance properties. □

Furthermore, by applying the Kolmogorov theorem and considering the independence of the sequences $(\theta_{\alpha,N})N \geq 1$, which are also independently distributed with a finite mathematical expectation of $x\alpha$, it can be shown that $\bar{\theta}_{\alpha,N}$ almost surely converges to $x_\alpha$:

$$P(\lim_{N \to \infty} \bar{\theta}_{\alpha,N} = x_\alpha) = 1.$$

The following theorem provides the composition of the variance in the proposed algorithm [1].

**Theorem 3.**

$$D\{\psi^k_\alpha(\tau)\} = \frac{c_{\alpha_0}}{p_{\alpha_0} p_\alpha} (\bar{A}^k_c \hat{c})_\alpha - (A^k_c c)^2_\alpha.$$

**Proof.** We are concerned with the variance

$$D\{\psi^k_\alpha(\tau)\} = E\{(\psi^k_\alpha(\tau))^2\} - (E\{\psi^k_\alpha(\tau))^2. \tag{18}$$

Consider the first term of (18).

$$\begin{aligned}
E\{(\psi^k_\alpha(\tau))^2\} &= E\left\{ \frac{c^2_{\alpha_0}}{p^2_{\alpha_0}} \frac{a^2_{\alpha_0\alpha_1} \ldots a^2_{\alpha_{k-1}\alpha_k}}{p^2_{\alpha_0\alpha_1} \ldots p^2_{\alpha_{k-1}\alpha_k}} \frac{c^2_{\alpha_k}}{p_{\alpha_k}} \right\} \\
&= \sum_{\alpha_0, \ldots \alpha_k = 1}^{n} \frac{c^2_{\alpha_0}}{p^2_{\alpha_0}} \frac{a^2_{\alpha_0\alpha_1} \ldots a^2_{\alpha_{k-1}\alpha_k}}{p^2_{\alpha_0\alpha_1} \ldots p^2_{\alpha_{k-1}\alpha_k}} \frac{c^2_{\alpha_k}}{p^2_{\alpha_k}} p_{\alpha_0} p_{\alpha_0\alpha_1} \ldots p_{\alpha_{k-1}\alpha_k} p_{\alpha_k} \\
&= \sum_{\alpha_0, \ldots \alpha_k = 1}^{n} \frac{c^2_{\alpha_0}}{p_{\alpha_0}} |a_{\alpha_0\alpha_1}| \ldots |a_{\alpha_{k-1}\alpha_k}| \frac{c^2_{\alpha_k}}{p_{\alpha_k}} \\
&= \sum_{\alpha_0, \ldots \alpha_k = 1}^{n} \frac{|c_{\alpha_0}|}{p_{\alpha_0} p_{\alpha_k}} |c_{\alpha_0}| |a_{\alpha_0\alpha_1}| \ldots |a_{\alpha_{k-1}\alpha_k}| c^2_{\alpha_k} \\
&= \frac{|c_{\alpha_0}|}{p_{\alpha_0} p_\alpha} \times (\bar{A}^k_c \hat{c})_\alpha.
\end{aligned}$$

Similarly, it can be demonstrated that the second term of (18) is equal to $(A^k_c c)^2_\alpha$.

Thus, $D\{\psi_\alpha^k(\tau)\} = \frac{c_{\alpha_0}}{p_{\alpha_0} p_\alpha}(\bar{A}_c^k \hat{c})_\alpha - (A_c^k c)_\alpha^2 \diamondsuit \quad \square$

### 3.3. Balancing

The robustness of the MC algorithm is demonstrated in [38] with balanced matrices possessing matrix norms much smaller than 1, showcasing significant variance improvement compared to cases where matrices have norms close to 1. This underscores the crucial importance of input matrix balancing in MC computations, suggesting that a balancing procedure should serve as an initial preprocessing step to enhance the MC algorithms' overall quality. Furthermore, for matrices that are *close* to stochastic matrices, the MC algorithm' accuracy remains notably high.

**Theorem 4.** *Consider a perfectly balanced stochastic matrix*

$$A = \begin{pmatrix} \frac{1}{n} & \cdots & \frac{1}{n} \\ \vdots & & \\ \frac{1}{n} & \cdots & \frac{1}{n} \end{pmatrix}$$

*and the vector* $c = (1, \ldots, 1)^t$. *Then MC algorithm defined by trajectory probability* $P^k(\tau)$ *is a zero-variance MC algorithm.*

It is sufficient to show that the variance $Var\{X_\alpha^k(\tau)\}$ is zero. Obviously,

$$Ac = \begin{pmatrix} \frac{1}{n} & \cdots & \frac{1}{n} \\ \vdots & & \\ \frac{1}{n} & \cdots & \frac{1}{n} \end{pmatrix} \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix},$$

and also,

$$A^k c = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}.$$

Since

$$\bar{A} = \begin{pmatrix} \left|\frac{1}{n}\right| & \cdots & \left|\frac{1}{n}\right| \\ \vdots & & \\ \left|\frac{1}{n}\right| & \cdots & \left|\frac{1}{n}\right| \end{pmatrix} = A$$

and $\hat{c} = \{c_i^2\}_{i=1}^n = c$, we have: $\frac{|c_{\alpha_0}|}{p_{\alpha_0} p_\alpha} \times (\bar{A}_c^k \hat{c})_\alpha = 1$. Thus, we proved that $Var\{X_\alpha^k(\tau)\} = 0$.

It is feasible to explore matrix-vector iterations $A^k c$ as they form the foundation for Neumann series, which serves as an approximation for solving systems of linear algebraic equations. The underlying concept is to illustrate that as the matrix $A$ approaches a perfectly balanced matrix, the probability error of the algorithm diminishes. The theorem implies that the initial algorithm can be enhanced by achieving balance in the iteration matrix.

### 3.4. Interpolation MC Algorithms

**Definition 2.** *A MC algorithm with a probability error of zero is referred to as an interpolation MC algorithm.*

The subsequent theorem outlines the variance structure for the MAO algorithm. Let's introduce the following symbols: $\hat{h} = \{h_i^2\}_{i=1}^n$, $\bar{v} = \{|v_i|\}_{i=1}^n$, $\bar{A} = \{|a_{ij}|\}_{i,j=1}^n$.

**Theorem 5.**

$$D\{\theta^{(k)}\} = \| A_v^k \| \left(\bar{v}, \bar{A}^k \hat{h}\right) - (v, A^k h)^2.$$

**Corrolary 1.** *For a stochastic matrix $A$ and vectors $h = (1, \ldots, 1)^T$, $v = (\frac{1}{n}, \ldots, \frac{1}{n})$ the MC algorithm, characterized by the density distributions outlined earlier, is classified as an interpolation MC algorithm.*

The well-known Power method brings an approximation for the dominant eigenvalue $\lambda_1$. It uses the so-called *Rayleigh quotient* $\mu_k = \frac{(v, A^k h)}{(v, A^{k-1} h)}$:

$$\lambda_1 = \lim_{k \to \infty} \frac{(v, A^k h)}{(v, A^{k-1} h)},$$

where $v, h \in \mathbb{R}^n$ are arbitrary vectors. The Rayleigh quotient is used to obtain an approximation to $\lambda_1$:

$$\lambda_1 \approx \frac{(v, A^k h)}{(v, A^{k-1} h)}, \tag{19}$$

where $k$ is arbitrary large natural number.

Considering the MAO density distributions, the random variable $\theta^{(k)}$ can be expressed in the following manner:

$$\begin{aligned} \theta^{(k)} &= sing\{A_v^k\} \| A_v^k \| h_{\alpha_k} \\ &= \frac{v_{\alpha_0}}{|v_{\alpha_0}|} \frac{a_{\alpha_0 \alpha_1} \ldots a_{\alpha_{k-1} \alpha_k}}{|a_{\alpha_0 \alpha_1}| \ldots |a_{\alpha_{k-1} \alpha_k}|} \|v\| \|a_{\alpha_0}\| \ldots \|a_{\alpha_{k-1}}\| \qquad (20) \\ &= \frac{v_{\alpha_0}}{p_{\alpha_0}} \frac{a_{\alpha_0 \alpha_1} \ldots a_{\alpha_{k-1} \alpha_k}}{p_{\alpha_0 \alpha_1} \ldots p_{\alpha_{k-1} \alpha_k}} \qquad (21) \end{aligned}$$

We deal with the variance

$$D\{\theta^{(k)}\} = E\{(\theta^{(k)})^2\} - (E\{\theta^{(k)}\})^2. \tag{22}$$

The following equalities are true:

$$Ah = \begin{pmatrix} \frac{1}{n} & \ldots & \frac{1}{n} \\ \vdots & & \\ \frac{1}{n} & \ldots & \frac{1}{n} \end{pmatrix} \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}.$$

Obviously,

$$A^k h = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix},$$

and

$$(v, A^k h) = \left(\frac{1}{n}, \ldots, \frac{1}{n}\right) \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} = 1.$$

**Theorem 6.** *The Markov chain trajectory length to absorption is less than 1 over the smallest absorption probability.*

**Proof.** The chance for absorption at each step is at least m (where m is the smallest absorption probability). Thus, interpreting the steps to absorption as a geometrically distributed random variable, we see that the expected length of a trajectory is at most 1/m. Moreover, the standard deviation is finite and at most 1/m as well. □

### *3.5. Eigenvalue Problems*

For the Randomized Inverse Shifted Power Method the iteration $x^{new} = Ax^{old}$ is replaced by $x^{new} = Bx^{old}$, where $A$ and $B$ have the same eigenvectors. The power iteration $x^{new} = Bx^{old}$ converges to the eigenvector associated with $B$'s dominant eigenvalue. Since $A$ and $B$ have the same eigenvectors, we have also computed an eigenvector of $A$. Letting $\sigma$ denote a scalar, there are 3 common choices for $B$: $B = A - \sigma I$ named as the shifted power method, $B = A^{-1}$ named as the inverse power method, and $B = (A - \sigma I)^{-1}$ named as the inverse shifted power method. These observations are shown on Table 1.

**Table 1.** Relationship between eigenvalues of $A$ and $B$.

| $B$ | Eigenvalue of $B$ | Eigenvalue of $A$ |
|---|---|---|
| $A^{-1}$ | $\frac{1}{\lambda_A}$ | $\frac{1}{\lambda_B}$ |
| $A - \sigma I$ | $\lambda_A - \sigma$ | $\lambda_B + \sigma$ |
| $(A - \sigma I)^{-1}$ | $\frac{1}{\lambda_A - \sigma}$ | $\sigma + \frac{1}{\lambda_B}$ |

### *3.6. Convergence and Mapping*

To analyze the MC algorithms' convergence, taking into account the following functional equation

$$u - \lambda Lu = f, \tag{23}$$

where $\lambda$ is some parameter. Observe that the matrices can be treated as linear operators. Determine resolvent operator (matrix) $R_\lambda$ by

$$I + \lambda R_\lambda = (I - \lambda L)^{-1},$$

in which $I$ is the identity operator.

Let $\lambda_1, \lambda_2, \dots$ be the eigenvalues of (23), where it is supposed that

$$|\lambda_1| \geq |\lambda_2| \geq \dots$$

MC algorithms rely on the depiction

$$u = (I - \lambda L)^{-1} f = f + \lambda R_\lambda f,$$

where

$$R_\lambda = L + \lambda L^2 + \dots, \tag{24}$$

The systematic error of (24), when $m$ terms are used, is

$$r_s = O[(|\lambda|/|\lambda_1|)^{m+1} m^{\rho-1}], \tag{25}$$

where $\rho$ is the multiplicity of the root $\lambda_1$.

In the neighborhood of the point $\lambda = 0$ ($\lambda = 0 \in \Omega$) the resolvent can be determined by the series

$$R_\lambda f = \sum_{k=0}^{\infty} c_k \lambda^k,$$

where

$$c_k = L^{k+1} f.$$

Contemplate the variable $\alpha$ situated within the unit circle on the complex plane $\Delta(|\alpha| < 1)$.

The function

$$\lambda = \psi(\alpha) = a_1\alpha + a_2\alpha^2 + \ldots,$$

maps the domain $\Delta$ into $\Omega$.

$$R_{\psi(\alpha)}f = \sum_{j=0}^{\infty} b_j\alpha^j \,, \tag{26}$$

where $b_j = \sum_{k=1}^{j} d_k^{(j)} c_k$ and $d_k^{(j)} = \frac{1}{j!}\left[\frac{\partial^j}{\partial\alpha^j}[\psi(\alpha)]^k\right]_{\alpha=0}$.

Let us assume that all eigenvalues $\lambda_k$ are real and $\lambda_k \in (-\infty, -a]$, where $a > 0$ . Contemplate a mapping for the case ($\lambda = \lambda_* = 1$):

$$\lambda = \psi(\alpha) = \frac{4a\alpha}{(1-\alpha)^2}. \tag{27}$$

The sequence $R_{\psi(\alpha)}f$ for the transformation given by (27) exhibits both absolute and uniform convergence.

$$R_{\lambda_*}f \approx \sum_{k=1}^{m} b_k\alpha_k^k = \sum_{k=1}^{m} \alpha_*^k \sum_{i=1}^{k} d_i^{(k)} c_i = \sum_{k=1}^{m} g_k^{(m)} c_k, \tag{28}$$

where

$$g_k^{(m)} = \sum_{j=k}^{m} d_k^{(j)} \alpha_*^j. \tag{29}$$

The coefficients

$$d_k^{(j)} = (4a)^k q_{k,j}$$

and $g_k^{(m)}$ can be calculated in advance.

If $q < 0$, then

$$\frac{(v, AR_q^m h)}{(v, R_q^m h)} \approx \frac{1}{q}\left(1 - \frac{1}{\mu^{(k)}}\right) \approx \lambda_n, \tag{30}$$

where $\lambda_n = \lambda_{min}$ is the minimal by modulo eigenvalue, and $\mu^{(k)}$ is the approximation to the dominant eigenvalue of $R_q$.

If $|q| > 0$, then

$$\frac{(v, AR_q^m h)}{(v, R_q^m h)} \approx \lambda_1, \tag{31}$$

where $\lambda_1 = \lambda_{max}$ is the dominant

## 4. Description of the WE MC Algorithms

Now we will give the description of the new WE MC algorithm, the old WE MC algorithm, and a combination between the two algorithms—the hybrid WE MC algorithm.

### *4.1. The First Monte Carlo Algorithm for Computing One Component of the Solution*

In this Section 4.1 we will give the description of the first (old) WE algorithm for linear systems described in details in [1]. For the Algorithm 1 we need in advance as input data the matrix $B$, the vector $\mathbf{f}$, the constant parameter $\gamma \in (0, 1]$ and the number of random trajectories $N$.

**Algorithm 1:** Computing one component $x_{i_0}$ of the solution $x_i,\ i = 1, \ldots n$.

Initialization **Input** $B,\ \mathbf{f},\ \gamma \quad N$.
Preliminary calculations (preprocessing):
**Compute** the matrix $A$ using $\gamma \in (0, 1]$:

$$\{a_{ij}\}_{i,j=1}^{n} = \begin{cases} 1-\gamma & i = j \\ -\gamma \frac{b_{ij}}{b_{ii}} & i \neq j\,. \end{cases}$$

**Compute** the vector $asum$:

$$asum(i) = \sum_{j=1}^{n} |a_{ij}| \quad \text{for} \quad i = 1, 2, \ldots, n.$$

**compute** the vector of initial probabilities,

$$\pi_i = \frac{v_i}{\sum_{j=1}^{m} v_j},\ i = 1, 2, \ldots, m;$$

**Compute** the transition probability matrix $P = \{p_{ij}\}_{i,j=1}^{n}$,

$$p_{ij} = \frac{|a_{ij}|}{asum(i)},\ \ i = 1, 2, \ldots, n \quad j = 1, 2, \ldots, n\,.$$

**Set** $S := 0$ .
**for** $k = 1$ **to** $N$ **do** (MC loop)
**set** $m := i_0$ .
**set** $S := S + b(m)$.
$test = 0;\ sign = 1$
**while** $(test \neq 0)$ **do**:
generate a udr variable $r \in (0, 1)$
**if** $r \geq asum(m)$ **then** $test = 1$;
**else** chose the index $j$ according to $p_{ij}$;
**set** $sign = sign * sign\{a_{mj}\};\ m = j$;
**update** $S := S + sign * b_m$;
**endif**.
**endwhile**
**enddo**
$x_{i_0} = \frac{S}{N}$

### *4.2. The First WE Monte Carlo Algorithm for Evaluating the Solution' All Components*

In this Section 4.2, to simplify the situation in Algorithm 2, let's assume that all the entries in the matrix are nonnegative, i.e., $a_{ij} \geq 0$, for $i, j = 1, \ldots n$.

### *4.3. The New WE Monte Carlo Algorithm for Computing One Component of the Solution*

In this Section 4.3, we introduce the novel Monte Carlo algorithm designed for computing a specific component $x_{i_0}$ of the solution in a linear system with real coefficients. The algorithm initiates its walk on row $i_0$ of the system, adding $b_{i_0}$ to the walk's score. Subsequently, the walk either undergoes absorption or moves to another equation based on probabilities $p_{i_0 j}$ outlined in Algorithm 3 below. The contributions to the score are adjusted in accordance with the signs of the coefficients in the system. Further details are provided in the following Algorithm 3.

**Algorithm 2:** Computing all components $x_i,\ i = 1, \ldots n$ of the solution.

Initialization.
Preprocessing.
**for** $i = 1$ **to** $n$ **do**
  $S(i) := 0$; $V(i) := 0$
  **for** $k = 1$ **to** $N$ **do** (MC loop)
    **set** $m := rand(1 : n)$ .
    **set** $test := 0$; $m_1 := 0$;
    **while** $(test \neq 1)$ **do**:
      $V(m) := V(m) + 1$; $m = m_1 + 1$; $l(m_1) = m$;
      **for** $q = 1, m_1$ **do**:
      $S(l(q)) := S(l(q)) + b(l(q))$;
      **if** $r > asum(m)$, **then** $test = 1$;
      **else** chose $j$ according to $p_{i,j}$;
      **set** $m = j$.
      **endif**
    **endwhile**
  **enddo**
  **for** $j = 1, n$ **do**:
    $V(j) := \max\{1, V(j)\}$;
    $S(j) = \frac{S(j)}{V(j)}$.

**Algorithm 3:** Computing one component $x_{i_0}$ of the solution $x_i,\ i = 1, \ldots, n$.

Initialization
  **Input** initial data: the matrix $A$, the vector $\mathbf{b}$, and the number of random trajectories $N$.
Preliminary calculations (preprocessing):
  **Compute** the vector $asum$:

$$asum(i) = \sum_{j=1}^{n} a_{ij} \quad \text{for} \quad i = 1, 2, \ldots, n.$$

  **Compute** the sum of elements in vector $b$:

$$sb = \sum_{i=1}^{n} b_i \quad \text{for} \quad i = 1, 2, \ldots, n.$$

**Set** $S := 0$ .
**for** $k = 1$ **to** $N$ **do** (MC loop)
  $test = 0$.
  choose the index $j$, according to the probability density vector $(b_1, b_2, \ldots, b_n)$; $pj = j$;
  **while** $test \neq 1$ **do**:
    choose the new value of the index $j$, according to the probability density vector $\left(a_{pj,1}, a_{pj,2}, \ldots, a_{pj,n}, 1 - asum(pj)\right)$;
    **If** $j \neq n + 1$ **then** $pj = j$
    **else** $test = 1$; $S = S + b(i_0) + sb * a_{pj,i_0} / \left(1 - asum(pj)\right)$
    **endif**
  **endwhile**
**enddo**
$x_{i_0} = \dfrac{S}{N}$.

### *4.4. The New WE Monte Carlo Algorithm for Computing All Components of the Solution*

In this Section 4.4, we present the novel Algorithm 4 designed for the computation of all components of the solution. The core idea is to calculate scores for all states, treated as new starting points, encountered throughout a given trajectory. The initial equation is randomly selected from the $n$ equations with uniform probability. Subsequently, for each state $i$, we define the total score $S(i)$ and the total number of visits $V(i)$, both of which are adjusted whenever state $i$ is visited during a walk. The parameter $cc$ represents the number of trajectories for component $x_i$ of the solution. The algorithm's key element involves the application of the Sequential Monte Carlo (Sequential MC) method for linear systems, as introduced by John Halton [49–52], based on the control variate method [39].

**Algorithm 4:** Computing all components of the solution $x_i,\ i = 1, \ldots, n$.

Initialization

**Input** initial data: the matrix $A$, the vector $\mathbf{b}$, and the number of random trajectories $N$.

Preliminary calculations (preprocessing):

**Compute** the vector $asum$:

$$asum(i) = \sum_{j=1}^{n} a_{ij} \quad \text{for} \quad i = 1, 2, \ldots, n.$$

**Compute** the sum of elements in vector $b$:

$$sb = \sum_{i=1}^{n} b_i \quad \text{for} \quad i = 1, 2, \ldots, n.$$

**Set** $S(i) = 0$ and $cc(i) = 0$ for $i = 1$ to $n$ .

**for** $k = 1$ **to** $N$ **do** (MC loop)

choose component $i_0$ to be estimated uniformly at random from $1, 2, \ldots$

increment $cc(i_0)$ by 1.

$test = 0$.

choose the index $j$, according to the probability density vector $(b_1, b_2, \ldots, b_n)$; $pj = j$;

**while** $test \neq 1$ **do**:

choose the new value of the index $j$, according to the probability density vector $\left(a_{pj,1}, a_{pj,2}, \ldots, a_{pj,n}, 1 - asum(pj)\right)$;

**If** $j \neq n + 1$ **then** $pj = j$

**else** $test = 1$; $S(i_0) = S(i_0) + b(i_0) + sb * a_{pj,i_0} / \left(1 - asum(pj)\right)$

**endif**

**endwhile**

**enddo**

**for** $i = 1$ **to** $n$ **do** $x_i = \dfrac{S(i)}{cc(i)}$.

### *4.5. The Hybrid Monte Carlo Algorithm for Computing One Component of the Solution*

In this Section 4.5 we will give the description of the hybrid WE algorithm for linear systems. The hybrid WE Algorithm 5 seems like the old algorithm WE but used if you got into an absorbing state (the state where the trajectory is broken) like the new algorithm.

**Algorithm 5:** Computing one component $x_{i_0}$ of the solution $x_i,\ i = 1, \dots n$.

Initialization **Input**: $B, \mathbf{f},\ \gamma\ \ N$.
Preliminary calculations (preprocessing):
**Compute** $A$ using $\gamma \in (0, 1]$:

$$\{a_{ij}\}_{i,j=1}^{n} = \begin{cases} 1-\gamma & i = j \\ -\gamma \frac{b_{ij}}{b_{ii}} & i \neq j\,. \end{cases}$$

**Compute** the vector $asum$:

$$asum(i) = \sum_{j=1}^{n} |a_{ij}| \quad \text{for} \quad i = 1, 2, \dots, n.$$

**compute**

$$\pi_i = \frac{v_i}{\sum_{j=1}^{m} v_j},\ i = 1, 2, \dots, m;$$

**Compute** $P = \{p_{ij}\}_{i,j=1}^{n}$, where

$$p_{ij} = \frac{|a_{ij}|}{asum(i)},\ \ i = 1, 2, \dots, n \quad j = 1, 2, \dots, n\,.$$

**Set** $S := 0$ .
**for** $k = 1$ **to** $N$ **do** (MC loop)
**set** $m := i_0$ .
**set** $S := S + b(m)$.
$test = 0;\ sign = 1$
**while** $(test \neq 0)$ **do**:
generate a udr variable $r \in (0, 1)$
choose the new value of the index $j$, according to the probability density vector $\left(a_{pj,1}, a_{pj,2}, \dots, a_{pj,n}, 1 - asum(pj)\right)$;
**if** $j = n + 1$ **then** $test = 1$;
**else** $sign = sign * sign\{a_{mj}\};\ m = j$;
**update** $S := S + sign * b_m$;
**endif**.
**endwhile**
**enddo**

### *4.6. The Hybrid WE Monte Carlo Algorithm for Computing All Components of the Solution*

Finally, in this Section 4.6 we will give the description of the Hybrid WE MC algorithm for computing all components of the solution. Later we will see that in some situations this hybrid Algorithm 6 gives the best results compared to the new and the old algorithm.

### *4.7. Sequential Monte Carlo*

The Sequential Monte Carlo (SMC) approach for LAEs was initially pioneered by John Halton in the 1960s [49] and has seen subsequent advancements in more recent publications [50–52]. This technique has been applied across various domains, including the computation of approximations on orthogonal bases [39] and the precise solution of linear partial differential equations [53].

**Algorithm 6:** Computing all components $x_i,\ i = 1, \ldots n$ of the solution.

Initialization.
Preprocessing.
**for** $i = 1$ **to** $n$ **do**
  $S(i) := 0$; $V(i) := 0$
  **for** $k = 1$ **to** $N$ **do** (MC loop)
    **set** $m := rand(1 : n)$ .
    **set** $test := 0$; $m_1 := 0$;
    **while** $(test \neq 1)$ **do**:
      $V(m) := V(m) + 1$; $m = m_1 + 1$; $l(m_1) = m$;
      **for** $q = 1, m_1$ **do**:
        $S(l(q)) := S(l(q)) + b(l(q))$;
      choose the new value of the index $j$, according to the probability density vector $\left(a_{pj,1}, a_{pj,2}, \ldots, a_{pj,n}, 1 - asum(pj)\right)$;
      **if** $j = n + 1$, **then** $test = 1$;
      **set** $m = j$.
      **endif**
    **endwhile**
  **enddo**
  **for** $j = 1, n$ **do**:
    $V(j) := \max\{1, V(j)\}$;
    $S(j) = \frac{S(j)}{V(j)}$.

The SMC method is based on the simple yet highly efficient concept of iteratively applying the control variate method. In the realm of linear systems, where the objective is to determine the vector solution $u$ for the equation $u = Au + b$, the method capitalizes on the fact that minimizing the variance of the algorithm is achieved when $b$ is small, and when $b = 0$, the variance becomes zero. Exploiting the linearity of our problem, a new system is constructed, with its solution representing the residual of the original system and having a small $b$. Let $u^{(1)}$ denote the approximate solution obtained using the aforementioned algorithm. The residual

$$r = u - u^{(1)}$$

satisfies the new equation

$$r = Ar + b - Au^{(1)},$$

where $b - Au^{(1)}$ is expected to be close to zero. We calculate the approximation $r^{(1)}$ for this equation using our algorithm, resulting in the approximation

$$u^{(2)} = u^{(1)} + r^{(1)}$$

for the original equation. This iterative process is applied to obtain the estimate $u^{(k)}$ after $k$ steps of the algorithm.

Demonstrated in [53], it has been established that when the number of samples $N$ utilized in computing $u^{(k)}$ is sufficiently large, the variance of $u^{(k)}$ decreases exponentially with respect to $k$. This form of convergence, referred to as zero-variance algorithms [36,38,39,48,54–56], enables these algorithms to effectively compete with optimal deterministic algorithms based on Krylov subspace methods and conjugate gradient methods (CGM), which have numerous variations.

However, a critical consideration is warranted. Performing the matrix-vector multiplication $Ar$ deterministically requires $O(n^2)$ operations. This implies that for a high-quality WE algorithm combined with SMC to be efficient, only a small number of sequential steps is necessary. The subsequent section will present numerical results showcasing high accuracy achieved in just a few sequential steps.

*4.8. Dominancy Number*

One of the key novelties in our paper is that we will introduce the special dominancy parameter $\mathcal{D}$ .

$$\mathcal{D} = \min\left\{ \frac{|b_{ii}| - \sum_{j \neq i} |b_{ij}|}{|b_{ii}|} \;\middle|\; 1 \leq i \leq n \right\}.$$

The condition number is consistently positive when dealing with diagonally dominant matrices, providing a rough indication of the dominance strength of the matrix's diagonal over the remaining elements. This concept is closely intertwined with the absorption probabilities in the Markov chain employed for the stochastic computation of solutions to linear systems. It is noteworthy that when $\mathcal{D} \geq 0.5$ then the newly developed algorithm performs with a several orders of magnitude higher accuracy compared to the old one. Later we will give experiments when $\mathcal{D}$ is below and above 0.5 to see the behavior of the three algorithms under consideration.

We may introduce the following statement

**Theorem 7.** *If the dominancy number $\mathcal{D}$ is closer to 1, then the new algorithm is significantly better than the previous algorithm.*

**Proof.** When the dominancy number is close to 1, the preconditioned matrix has high absorption probabilities at each step. Thus, the length of each trajectory is small. Observe that in the new algorithm, the variable which estimates the solution is calculated once—at the end of each trajectory, and is irrespective of the length of the trajectory. On the other hand, in the old algorithm, the variable estimating the solution is updated along each step of a trajectory. Thus, it is natural to observe that more lengthy trajectories would produce better estimating variables. Whereas in the new algorithm, the length of a trajectory does not affect the calculation of the estimating variable. Thus, it is expected that the new algorithm performs better when the dominancy number is closer to 1. □

## 5. Numerical Tests

In this section we give ample numerical simulations to demonstrate the capabilities of the proposed approaches. The algorithms are implemented and run in a MATLAB® R2021b environment, using a mobile computer with 6-core processor and 16 GB RAM.

*5.1. The New Algorithm versus Deterministic Algorithms*

On the graphs we will illustrate the value of

$$\varepsilon(\hat{x}) = \frac{\|\hat{x} - x_{exact}\|}{\|x_{exact}\|},$$

where we check the accuracy of a computed solution $\hat{x}$, while $x_{exact}$ is the exact solution-vector. One may observe that the results for all other components exhibit similar trends.

To assess whether the weighted residual [57,58]:

$$\rho := \frac{||B\hat{x} - f||}{||B||\,||\hat{x}||}, \tag{32}$$

can serve as a reliable indicator of error, we juxtapose the relative error against the residual for a matrix of dimensions $n = 5000$ (refer to Figure 2). It is evident [1] that the curves representing the relative error and the residual closely align. The number of random trajectories for MC is $5n$. It is already established in [1] that for much large dense matrix of size $n$ = 25,000 the WE method is clearly better than the Jacobi method—see Figure 3.

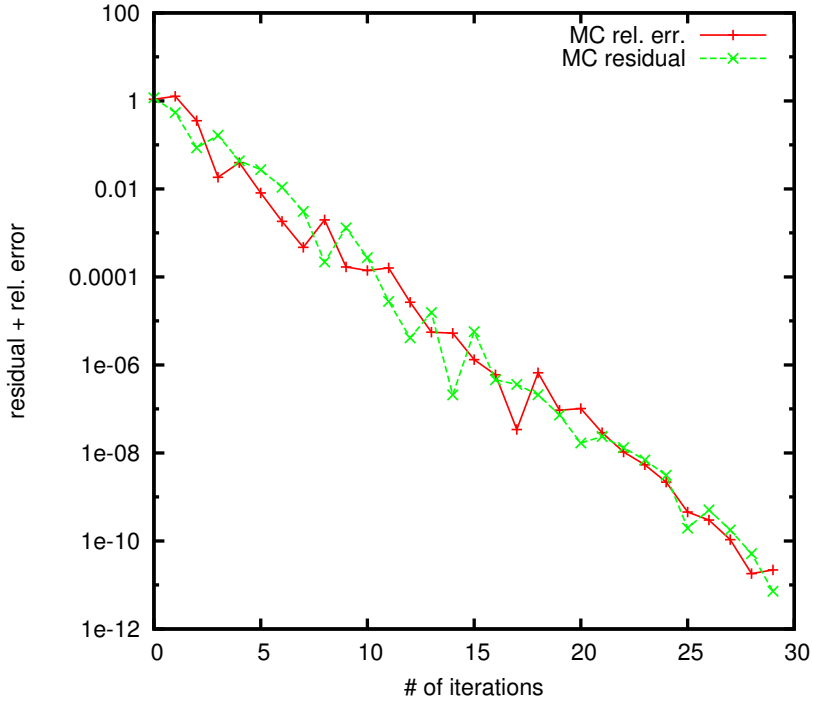



**Figure 2.** Comparison the relative error with the residual for the WE MC.

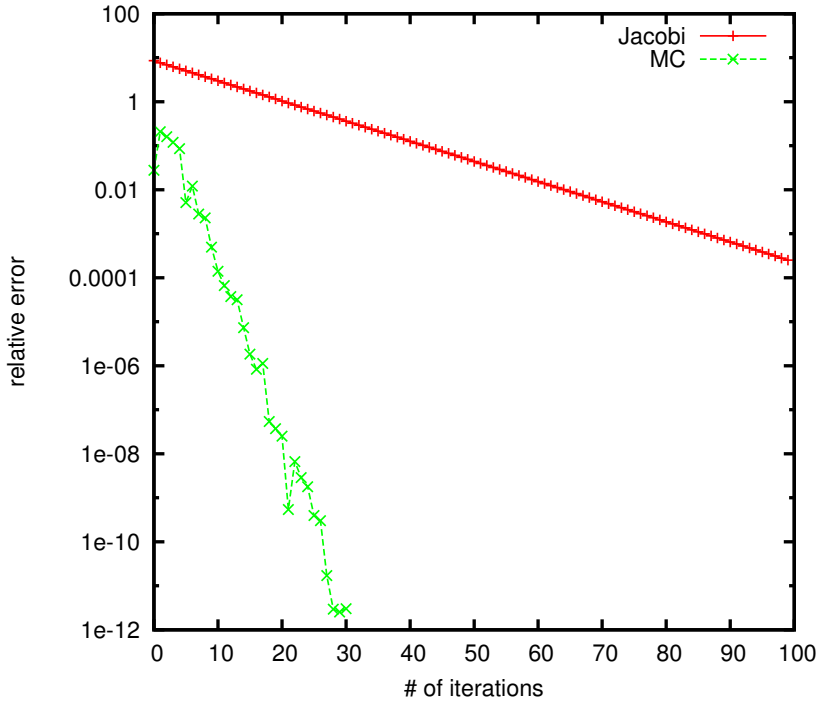



**Figure 3.** Relative error for the WE MC and Jacobi method for a matrix of size 25,000 × 25,000.

Here we conduct a number of numerical experiments with dense matrices of size $n = 10$, 100, 1000, 10,000. A comparative analysis is conducted with well-established deterministic methods, specifically Gauss–Seidel and Jacobi algorithms. Key observations are as follows. Firstly, across all experiments, Gauss–Seidel consistently outperforms the Jacobi method by a margin of at least three orders of magnitude. Notably, this study introduces a comparison with Gauss–Seidel for the first time, as previous comparisons in [1] exclusively involved the Jacobi method. For the matrix of size 10 × 10 on Figure 4 for 20 iterations, the relative error

produced by the Monte Carlo approach is about two magnitudes better than the relative error for Gauss–Seidel method and 7 magnitudes better than the Jacobi approach. For the matrix of size $100 \times 100$ on Figure 4 after just 15 iterations, the relative error from the Monte Carlo approach is approximately six orders of magnitude better than the Gauss–Seidel method and ten orders of magnitude better than the Jacobi method.

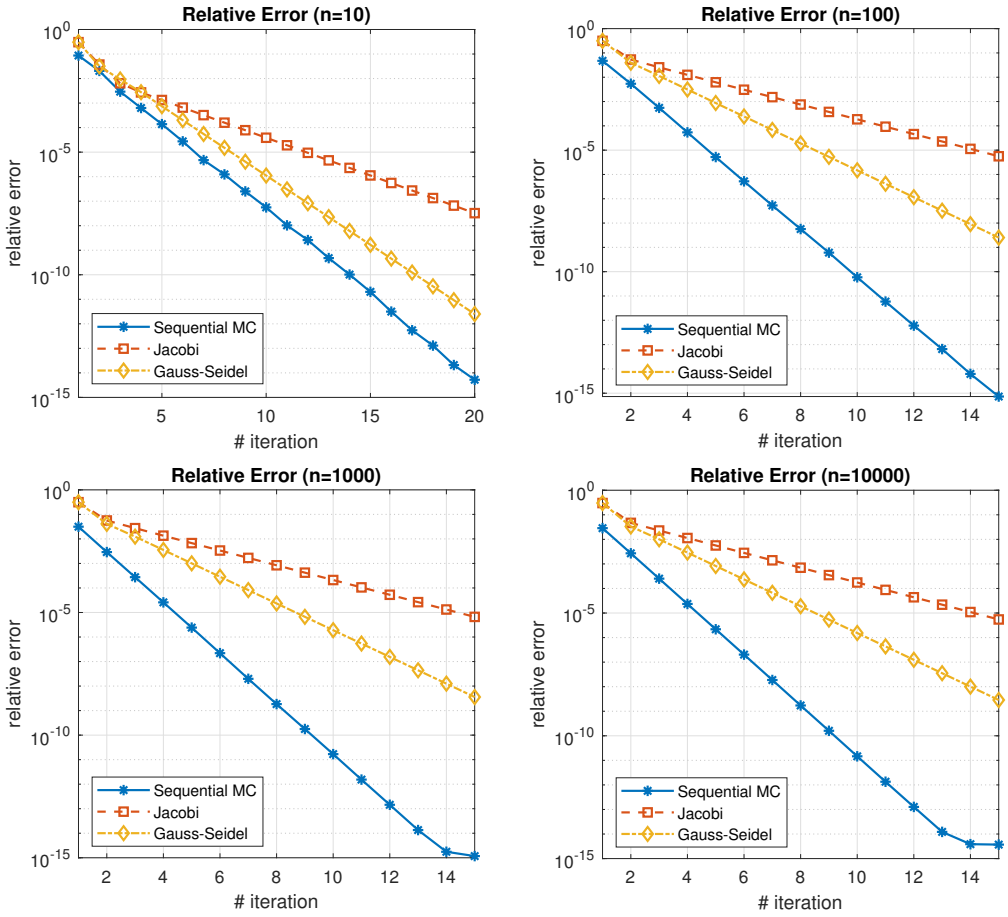



**Figure 4.** Relative error of random dense matrices $n \times n$ for $n = 10$ (top left), $n = 100$ (top right), $n = 1000$ (bottom left) and $n$ = 10,000 (bottom right).

As we increase the matrix dimension to $1000 \times 1000$ the new Monte Carlo approach continues to exhibit superior performance. In Figure 4, after 15 iterations, the relative error from the Monte Carlo approach is approximately seven orders of magnitude better than that of the Gauss–Seidel method and ten orders of magnitude better than the Jacobi method. This trend becomes even more pronounced with higher matrix dimensions. The most interesting case will be very high dimensional matrix of size $10{,}000 \times 10{,}000$. The results for this case are presented on Figure 4. Here, just for 14 iterations, the Monte Carlo method achieves a relative error of $10^{-16}$, while for the same number iterations Gauss–Seidel achieves $10^{-8}$ and Jacobi produces $10^{-5}$. Consequently, for very high dimensions, the developed Monte Carlo approach emerges as the most accurate, surpassing other methods by a significant margin across multiple orders of magnitude.

We conduct also a range of computational experiments involving dense matrices with dimensions $n = 2^{\{6,8,10,12\}}$. These experiments are compared against well-established deterministic algorithms, specifically the Gauss–Seidel and Jacobi methods, to facilitate a thorough analysis. Notably, the results reveal consistent superiority in accuracy for the Gauss–Seidel algorithm compared to the Jacobi method, with a minimum difference of three orders of magnitude in each computational test. This juxtaposition with Gauss–Seidel is a distinctive contribution, as prior studies such as [1] exclusively focused on comparing

with the Jacobi method. Taking the example of a $64 \times 64$ matrix (illustrated in Figure 5) after 15 iterations, our Monte Carlo approach demonstrates a relative error approximately five orders of magnitude lower than Gauss–Seidel and eight orders of magnitude lower than Jacobi.

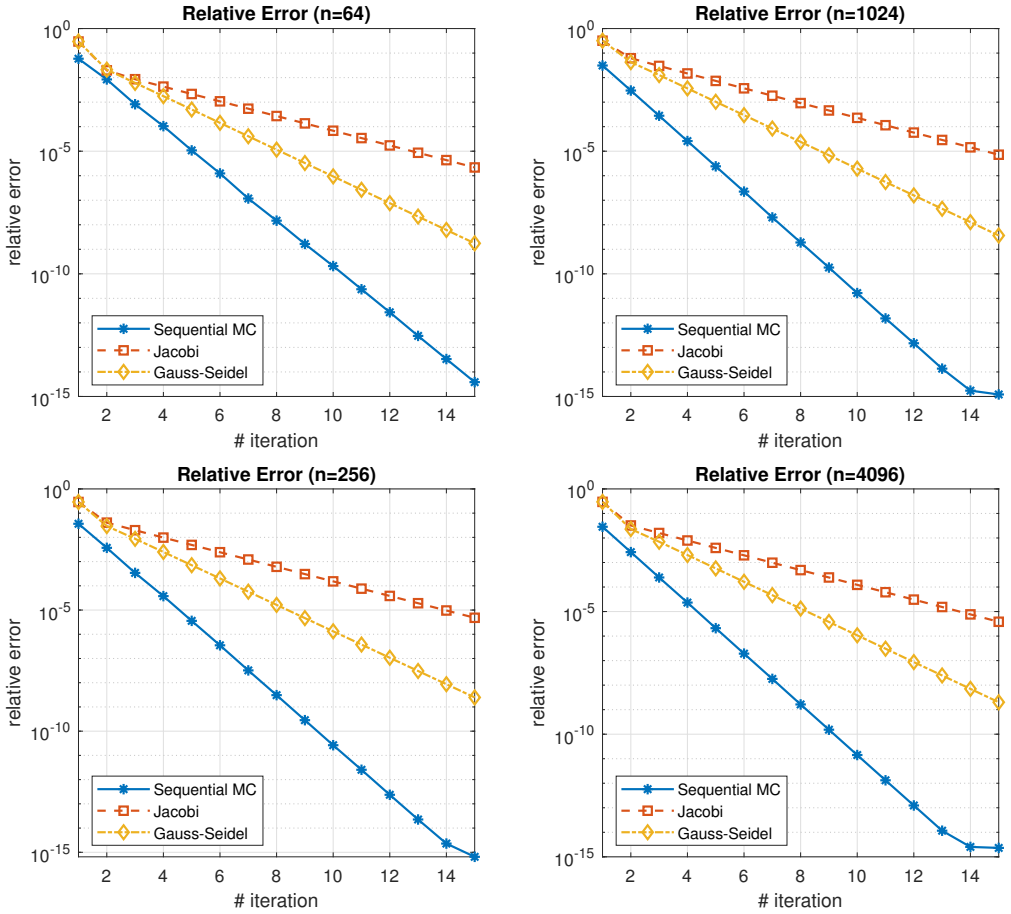



**Figure 5.** Relative error of random dense matrices $n \times n$ for $n = 64$ (top left), $n = 256$ (top right), $n = 1024$ (bottom left) and $n = 4096$ (bottom right).

Examining a larger matrix with dimensions $256 \times 256$ (illustrated in Figure 5) and restricting the iterations to just 15, we observe that the Monte Carlo approach outperforms Gauss–Seidel by roughly six orders of magnitude and surpasses the Jacobi method by approximately ten orders of magnitude in relative error.

As we further elevate the matrix size to $1024 \times 1024$, the performance differential becomes even more striking. With just 15 iterations (see Figure 5), the Monte Carlo method demonstrates a relative error that is seven orders of magnitude superior to Gauss–Seidel and ten orders of magnitude better than Jacobi. Hence, we can affirm that the performance disparity between our Monte Carlo approach and the Gauss–Seidel algorithm expands as we increase the matrix dimensions.

The apex of our analysis is focused on the colossal $4096 \times 4096$ matrix, the outcomes for which are illustrated in Figure 5. In a mere 14 iterations, the Monte Carlo methodology achieved an impressively low relative error of $10^{-15}$. Comparatively, Gauss–Seidel registered a relative error of $10^{-8}$ and Jacobi languished further behind with $10^{-5}$. Hence, for exceedingly large-scale problems, our developed Monte Carlo approach stands out, surpassing other algorithms by a substantial margin in terms of accuracy.

### *5.2. WE versus PCG*

We evaluate our outcomes against the optimal deterministic preconditioned conjugate gradient (PCG) method [57,59,60].

Here's more detailed information regarding our PCG method implementation. Our objective is to address a linear system of equations, $Bx = b$, utilizing the PCG iterative method. The matrix $B$ in our context is the square and sizable matrix $NOS4$ from the widely recognized Harwell-Boeing collection, with the requirement that $B$ is symmetric and positive definite. The NOS4 matrix is a notable example from this collection, primarily because of its distinct characteristics and the challenges it presents in computational processing.

The NOS4 matrix is a sparse matrix, a common type found in the Harley-Boeing collection, known for its large size and sparsity. This matrix is particularly representative of structural engineering problems, often derived from finite element analysis. The specific features of the NOS4 matrix are as follows.

The NOS4 matrix exhibits a highly sparse structure, which is characteristic of matrices derived from finite element methods in engineering. This sparsity is a significant factor in computational processing, as it requires specialized algorithms that can efficiently handle non-dense data distributions.

The matrix is relatively large, which adds to the computational challenge. Its size and the complexity of its elements make it an ideal candidate for testing the scalability and efficiency of our algorithm.

The matrix typically contains numerical values that represent physical properties in structural engineering contexts, such as stiffness and material properties. These values are crucial for accurately simulating and analyzing structural behavior under various conditions.

Specifically, in the context of our study, the NOS4 matrix is utilized to model problems in constructive mechanics. This involves simulations that are critical for understanding and predicting the behavior of structures, particularly in the design and analysis of buildings, bridges, and other infrastructural elements.

In our implementation, the parameters employed are as follows: $b$ represents the right-hand side vector; $tol$ stands for the required relative tolerance for the residual error, denoted as $b - Bx$, signifying that the iteration halts if $\parallel b - Bx \parallel \leq tol * \parallel b \parallel$; $maxit$ denotes the maximum permissible number of iterations; $m = m1 * m2$ corresponds to the (left) preconditioning matrix, ensuring that the iteration is theoretically equivalent to solving by PCG $Px = m\, b$, with $P = m\, B$; and $x_0$ signifies the initial guess.

Some detailed information about heaviest diagonals and conditioning for the matrix NOS4 is given on Figure 6. The illustration of the PCG for NOS4 is given on Figure 7. Our approach, devoid of such an optimization process, yields superior results. It is noteworthy that the specific matrix (NOS4) appears to pose a challenging test for PCG, but generalizing this observation is complex, as PCG success depends on the specific functional minimized in the method. Further analysis is required to determine the conditions under which the IWE and WE Monte Carlo methods outperform the widely used PCG.

Figure 8 depicts the convergence analysis of the WE Monte Carlo and PCG methods when applied to the matrix $NOS4$. This matrix originates from an application involving the finite element approximation of a problem associated with a beam structure in structural mechanics [61].

The outcomes depicted in Figure 8 indicate that the WE Monte Carlo exhibits superior convergence. The curve representing the residual with the number of iterations is smoother, reaching values as low as $10^{-6}$ or $10^{-7}$, compared to the PCG curve, which achieves an accuracy of $10^{-3}$. It is important to emphasize that, for the experiments showcased, the desired accuracy is specified as $tol = 10^{-8}$. One of the most significant distinctions between the Preconditioned Conjugate Gradient (PCG) method and our Walk on Equations (WE) Monte Carlo algorithm lies in their respective operational mechanisms within Krylov subspaces. While both methods operate within these optimal subspaces, the approach each takes is fundamentally different, leading to a crucial advantage for the WE algorithm.

**Heaviest diagonals**

| offset from main | 0 | -10 | 10 | -2 | 2 | -12 | -8 | 8 | 12 | -13 |
|---|---|---|---|---|---|---|---|---|---|---|
| nonzeros | 100 | 45 | 45 | 40 | 40 | 36 | 36 | 36 | 36 | 18 |
| accumulated percent | 16.84 | 24.41 | 31.99 | 38.72 | 45.45 | 51.52 | 57.58 | 63.64 | 69.70 | 72.73 |

Top 10 out of 19 nonvoid diagonals.

**Conditioning**

| Frobenius norm | 4.2 | condition number (est.) | 2.7e+03 |
|---|---|---|---|
| 2-norm (est.) | 0.85 | diagonal dominance | no |

**Figure 6.** Info for NOS4.

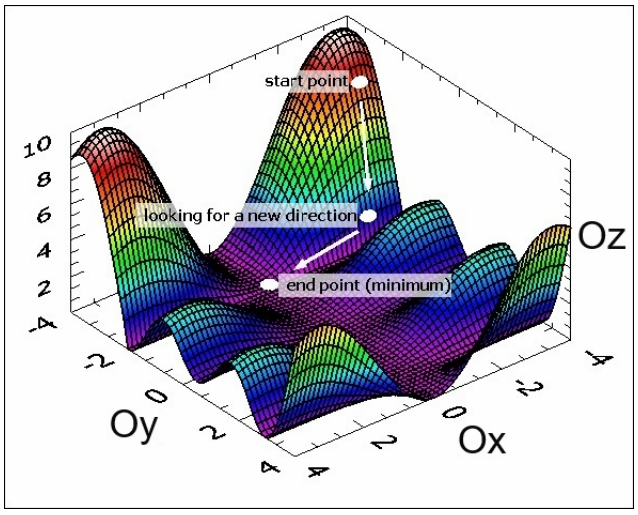



**Figure 7.** Illustration of PCG for NOS4 matrix.

The PCG method is fundamentally an optimization-based approach. It seeks to minimize the residual of the solution iteratively within the Krylov subspace. This process, while effective under certain conditions, inherently runs the risk of becoming ensnared in local minima. These local minima, which are close but not equivalent to the global minimum, can significantly hinder the convergence of the PCG method towards the most accurate solution. This susceptibility is illustrated in Figure 8, where the PCG method's trajectory towards convergence is impeded by the presence of a local minimum.

In contrast, our WE algorithm operates within the same Krylov subspaces but diverges from the PCG method by eschewing the optimization paradigm entirely. Instead, it leverages the probabilistic nature of Monte Carlo methods to traverse the solution space. This approach inherently bypasses the pitfalls of optimization-based methods, such as getting trapped in local minima. The WE algorithm, by not being constrained to follow the gradient descent pathway that optimization methods inherently use, is not prone to stalling or slowing in areas that only represent local solutions. This attribute allows the WE algorithm to explore the solution space more freely and thoroughly, increasing the likelihood of reaching the global solution without the hindrance of local minima.

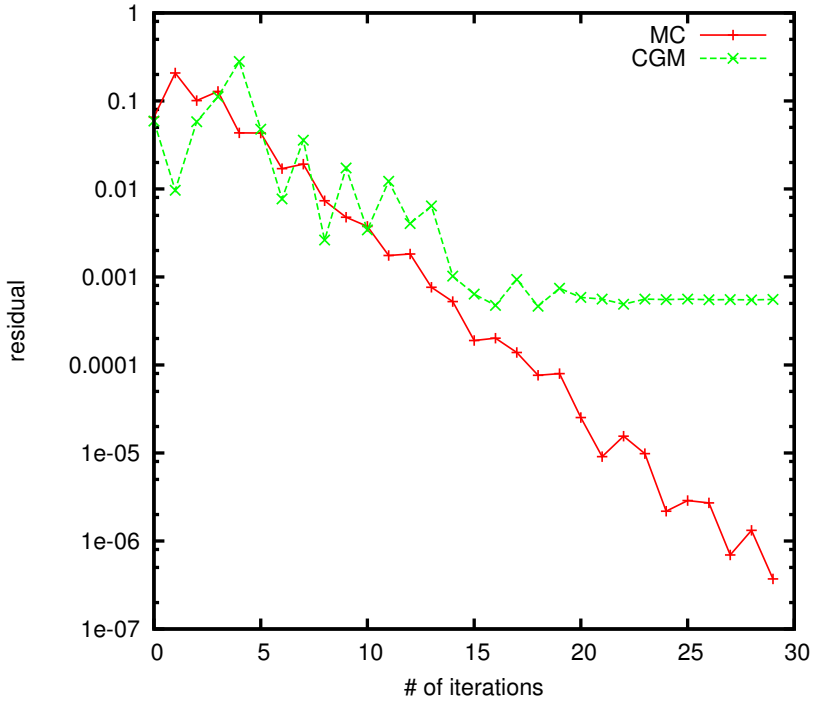



**Figure 8.** Comparison of the WE MC and the PCG method for *NOS*4.

Therefore, the fundamental difference in operational philosophy between the PCG and WE methods not only delineates their theoretical underpinnings but also translates into a practical advantage for the WE algorithm. By navigating the solution space without the constraints of an optimization problem, the WE algorithm demonstrates a robustness and efficiency in finding the global solution, particularly in complex linear algebraic systems where local minima can be a significant obstacle.

### *5.3. Testing the Balancing*

It is noteworthy that matrix balancing is a crucial concern for deterministic algorithms. Consequently, an arbitrary matrix necessitates a balancing procedure as part of the preprocessing steps. While MC algorithms are generally less sensitive to balancing, balancing does influence the convergence of the WE MC algorithm. As it is already established in [1], in the first instance, illustrated in Figure 9 (top left), we examine two small matrices with size $n = 10$. The balanced matrix is characterized by $a_{ij} = 0.1$, resulting in a parameter $a$ of 0.9. On the other hand, the unbalanced matrix has $a = 0.88$, with an extreme level of imbalance (the ratio between the largest and smallest elements by magnitude is 40). It is worth noting that a ratio is considered large when it reaches 5.

Certainly, for the unbalanced matrix, convergence is entirely absent. In contrast, the balanced matrix exhibits rapid convergence, albeit with a somewhat erratic *red* curve, attributable to the limited number of random trajectories. To explore this further, we revisited the same matrices, this time augmenting the number of random trajectories to $N = 500$. Concurrently, we reduced the number of random trajectories (and consequently the computational time) for the balanced matrix from $N = 50$ to $N = 30$. The outcomes of this experiment are illustrated in Figure 9 (top right).

The results clearly indicate that the WE algorithm initiates convergence, reaching three correct digits after approximately 35 to 40 iterations. While the balanced matrix still demonstrates swift convergence, the *red* curve retains its roughness, reflective of the limited number of random trajectories.

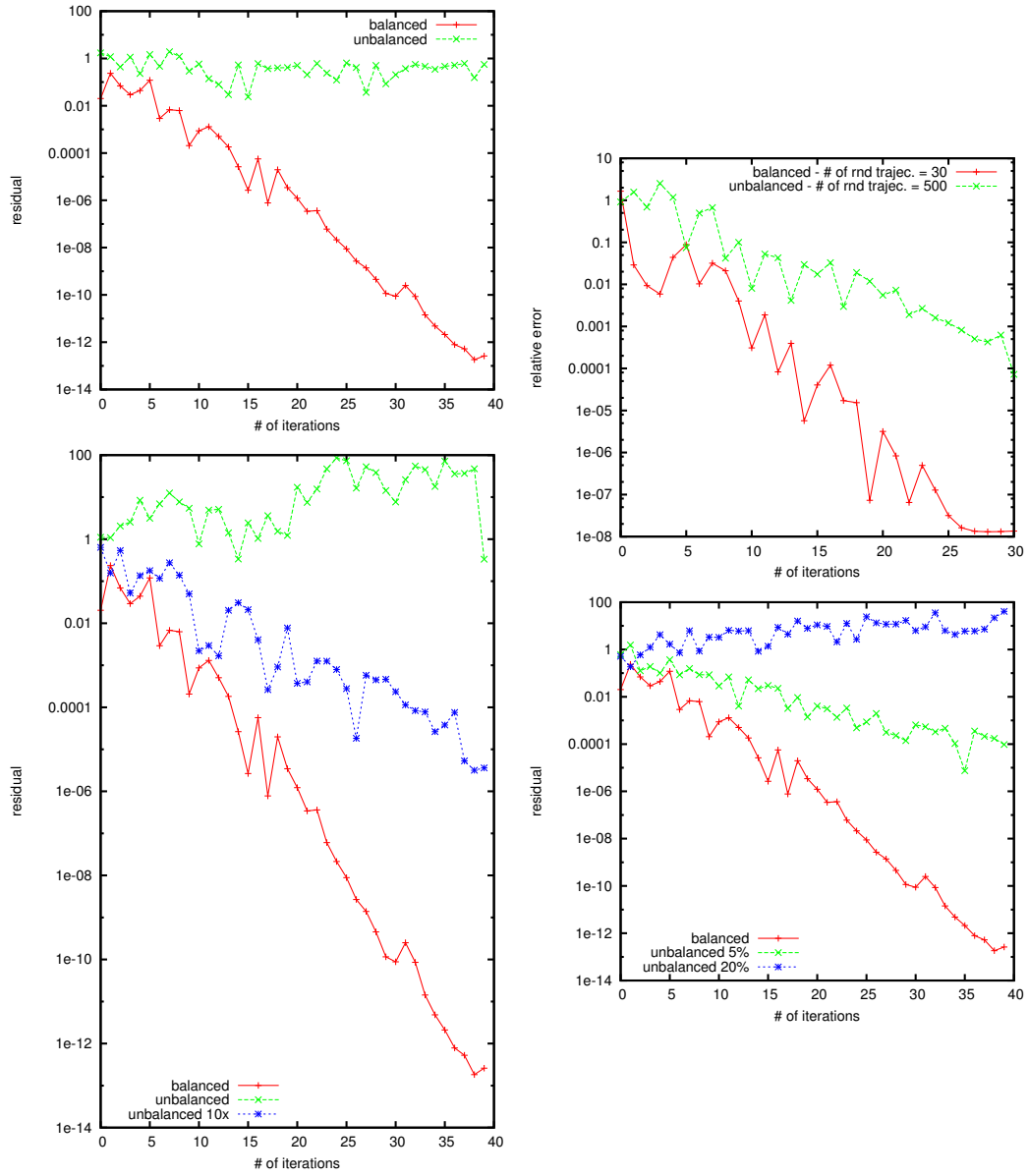



**Figure 9.** Residual for the WE MC for balanced and extremity unbalanced matrix of size $n = 10$. The number of random trajectories is $N = 5n$ (top left); Residual for the WE Monte Carlo for balanced and extremity unbalanced matrix of size $n = 10$. The number of random trajectories for the balanced matrix is $N = 5n$; for the unbalanced matrix there two cases: (i) $N = 3n$—red curve; (ii) $N = 50n$—green curve. (top right); Residual for the WE MC, the matrix of size is $5000 \times 5000$. The number of random MC trajectories is $5n$ (bottom left); Residual for the WE MC, the matrix of size is $100 \times 100$. Matrix elements are additionally perturbed by 5% and by 20% (bottom right).

For randomly generated matrices of size $n = 5000$, akin results are presented in Figure 9 (bottom left). The unbalanced matrix results from random perturbations to the elements of the balanced matrix. Both the *green* and *blue* curves pertain to the same unbalanced matrix, with the latter achieved by leveraging ten times more random trajectories. This demonstrates that increasing the number $N$ of random trajectories can enhance convergence, aligning with theoretical expectations.

In our specific numerical experiments depicted in Figure 9 (bottom left), the green curve is derived from a set of trajectories with $N = 5n$, while the blue curve is based on $N = 50n$ trajectories. It is evident that, in the former case, the algorithm fails to converge, whereas in the latter case, convergence is achieved. These experiments underscore that

the WE Monte Carlo exhibits an additional degree of freedom, absent in deterministic algorithms, allowing for enhanced convergence even with severely imbalanced matrices. However, this improvement in convergence comes at the cost of increased complexity, or computational time, due to the augmented number of simulation trajectories.

To explore the impact of matrix imbalance on the convergence of the WE Monte Carlo, we conducted a noteworthy experiment. By randomly generating matrices of varying sizes, we introduced random perturbations to the matrix elements, progressively increasing the matrix's imbalance. The results for a matrix of size $n = 100$ are illustrated in Figure 9 (bottom right). The matrix's randomly generated entries were additionally perturbed by 5% and 20%.

Matrix balancing plays a pivotal role in the performance of many computational algorithms, particularly in solving linear algebraic systems. While MC algorithms, including our WE MC algorithm, are generally less sensitive to matrix balancing compared to deterministic algorithms, the impact of balancing on convergence is non-negligible.

To provide a concrete context for the necessity of matrix balancing, let us consider a practical example from structural engineering. In finite element analysis, matrices often represent stiffness or connectivity matrices of a physical structure, like a bridge or building. These matrices can become imbalanced due to variations in material properties or non-uniform loading conditions. An imbalanced matrix in such scenarios could lead to inaccurate or inefficient computational analysis, impacting the reliability of the structural assessment.

Given the practical implications, it is vital to establish criteria for when and how to apply matrix balancing:

- *Magnitude of Element Variation*: Balancing should be considered when there is a significant disparity in the magnitude of matrix elements. A practical threshold could be set; for instance, a ratio of 5 or more between the largest and smallest elements by magnitude indicates a need for balancing.
- *Application-Specific Needs*: In fields such as structural engineering or fluid dynamics, where matrices represent physical phenomena, the need for balancing should be assessed based on the physical realism of the matrix. If imbalances in the matrix do not correspond to realistic physical conditions, balancing may be necessary.
- *Computational Efficiency*: If preliminary runs of the algorithm show a slow convergence rate, this could signal the need for balancing, especially in large matrices where computational efficiency is paramount.

### *5.4. Comparison between the Three WE Algorithms*

This subsection is very important, because it will address the issue in which cases the new algorithm is better than the old algorithm.

We consider randomly chosen matrix $100 \times 100$ and $1000 \times 1000$ with values of the dominancy parameter $\mathcal{D} = 0.95$. While in the case of lower $\mathcal{D}$ the two algorithms produce similar results with the new algorithm being better at high sequential steps, for higher $\mathcal{D}$ the advantage of the new algorithm can be clearly seen—see Table 2. The apex of our analysis is focused on the $1000 \times 1000$ matrix, the outcomes for which are illustrated in Table 2. In a mere 5 iterations, the new Monte Carlo algorithm achieved an impressively low relative error of $10^{-12}$. Comparatively, the old WE registered a relative error of $10^{-9}$. Hence, for exceedingly large-scale problems, our developed Monte Carlo approach stands out, surpassing the old algorithm by a substantial margin in terms of accuracy.

**Table 2.** Weighted residual for the matrix $B \in \mathbb{R}^{100\times 100}$ (1) with a dominancy parameter $\mathcal{D} = 0.94234$ and $B \in \mathbb{R}^{1000\times 1000}$ (2) with a dominancy parameter $\mathcal{D} = 0.947989$.

| N | New WE for (1) | Old WE for (1) | New WE for (2) | Old WE for (2) |
|---|---|---|---|---|
| 1 | 5.61119e−03 | 2.48318e−02 | 5.13837e−03 | 2.23855e−02 |
| 2 | 2.26076e−05 | 5.17592e−04 | 2.74535e−05 | 5.36106e−04 |
| 3 | 1.35103e−07 | 1.0521e−05 | 1.27667e−07 | 1.17095e−05 |
| 4 | 5.60699e−10 | 3.10678e−07 | 6.27896e−10 | 2.69074e−07 |
| 5 | 3.05923e−12 | 7.04298e−09 | 3.09402e−12 | 6.63235e−09 |

We can see the comparison between the three algorithms on Figure 10. The corresponding dominancy numbers are $\mathcal{D} = 0.466, \mathcal{D} = 0.484, \mathcal{D} = 0.484, \mathcal{D} = 0.489$. We can observe similar behavior between the three algorithms with the slight advantage to the hybrid and the new WE algorithm due to the fact that the dominancy number $\mathcal{D}$ is closer to 0.5. When we increase the dominancy number the advantage of the new algorithm versus the old algorithm become more evident—see the results on Figure 11. The corresponding dominancy numbers are $\mathcal{D} = 0.5618, \mathcal{D} = 0.5899, \mathcal{D} = 0.6033, \mathcal{D} = 0.6069$. When we augment the dominancy number, the superiority of the new algorithm over the old algorithm becomes increasingly pronounced. As the dominancy number, which characterizes the influence of the diagonal elements within a system, grows, the effectiveness and efficiency of the new algorithm become more evident in comparison to the older method. This trend suggests that the performance gap between the two algorithms widens as the system's key features gain greater prominence, underscoring the enhanced adaptability and efficacy of the new algorithm in handling scenarios with elevated dominancy numbers. This observation reinforces the notion that the advantages of the new algorithm are particularly prominent and advantageous in situations where dominant factors play a more significant role. When we decrease the dominancy number the advantage of the hybrid algorithm versus the old and new algorithm become more evident—see the results on Figure 12. The corresponding dominancy numbers are $\mathcal{D} = 0.2863, \mathcal{D} = 0.3349, \mathcal{D} = 0.3333, \mathcal{D} = 0.3468$. It is worth mentioning that in case of low dominancy numbers the difference between the old and the new algorithm is very similar. This fact could be explained by the proof of Theorem 7, but needs more careful analysis and will be an object of a future study.

Here we describe the significance of the dominancy number $\mathcal{D}$ and its impact on the performance of our new Monte Carlo algorithm compared to traditional methods. To provide a clearer understanding of how the dominancy number relates to real-world systems, we will establish a link between the bands of this parameter and their applications in various scenarios.

#### 5.4.1. Dominancy Number in Real Systems

1. *Structural Engineering and Mechanics*: In structural engineering, matrices derived from finite element analysis often have dominancy numbers reflecting the stiffness of the structure. A higher dominancy number (closer to 1) might indicate a structure with more uniform material properties or loading conditions, whereas a lower dominancy number (closer to 0) could represent a structure with highly variable stiffness or irregular load distribution. For instance, in a uniformly loaded beam, the stiffness matrix would likely exhibit a higher dominancy number, indicating the potential effectiveness of our new algorithm in such scenarios.

2. *Electrical Networks and Circuit Analysis*: In electrical engineering, matrices representing circuit networks often have dominancy numbers that reflect the distribution of resistances or impedances. A circuit with evenly distributed resistances across components may present a higher dominancy number, suggesting a system where our new algorithm could be particularly efficient.

3. *Fluid Dynamics and Heat Transfer*: Systems modeled for fluid dynamics or heat transfer can have matrices with varying dominancy numbers depending on the homogeneity of the medium or the boundary conditions. For example, a heat transfer problem in a homogenous medium might have a higher dominancy number, indicating the suitability of our new algorithm for such cases.

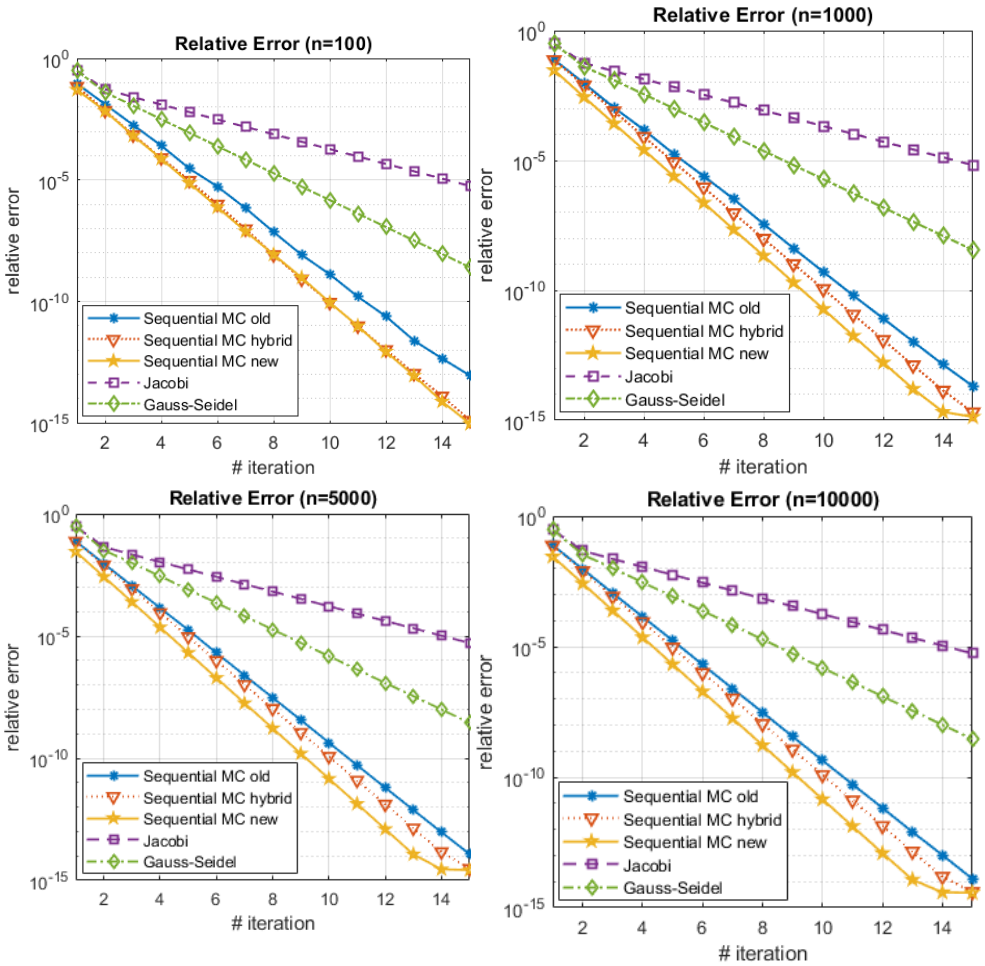



**Figure 10.** Relative error of random dense matrices $n \times n$ for $n = 100$ (top left), $n = 1000$ (top right), $n = 5000$ (bottom left) and $n$ = 10,000 (bottom right).

5.4.2. Practical Implications of Dominancy Number Bands

- *High Dominancy Band* ($\mathcal{D} \approx 0.6$–$1.0$): Systems with high dominancy numbers are likely to benefit more from our new algorithm. These systems typically exhibit more uniform properties or conditions, making our algorithm more efficient in handling them.
- *Medium Dominancy Band* ($\mathcal{D} \approx 0.4$–$0.6$): In this range, the performance of our new algorithm is comparable to traditional methods, with slight advantages in certain scenarios. Systems in this band often exhibit moderate variability in their properties.
- *Low Dominancy Band* ($\mathcal{D} \approx 0.0$–$0.4$): Systems with low dominancy numbers might not exhibit a significant performance difference between our new algorithm and traditional methods. These systems often represent highly variable or irregular conditions.

The dominancy number is a critical parameter in determining the suitability and efficiency of our new Monte Carlo algorithm for various real-world systems. By understanding the implications of different dominancy number bands, practitioners can better assess the applicability of our algorithm to specific scenarios, ranging from structural engineering to electrical networks.

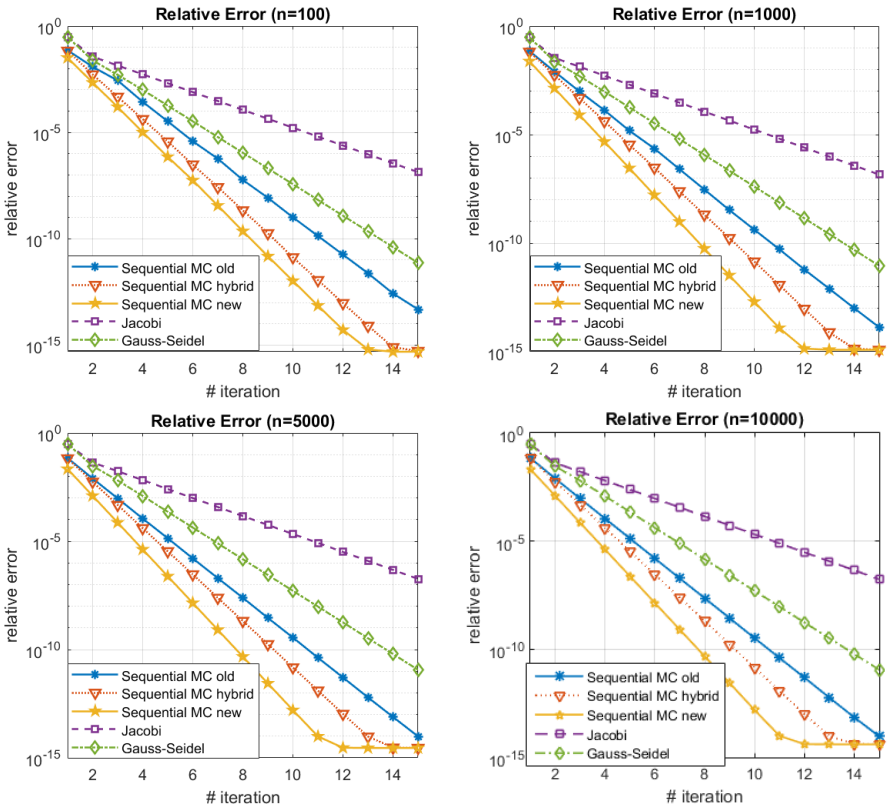



**Figure 11.** Relative error of random dense matrices $n \times n$ for $n = 100$ (top left), $n = 1000$ (top right), $n = 5000$ (bottom left) and $n$ = 10,000 (bottom right).

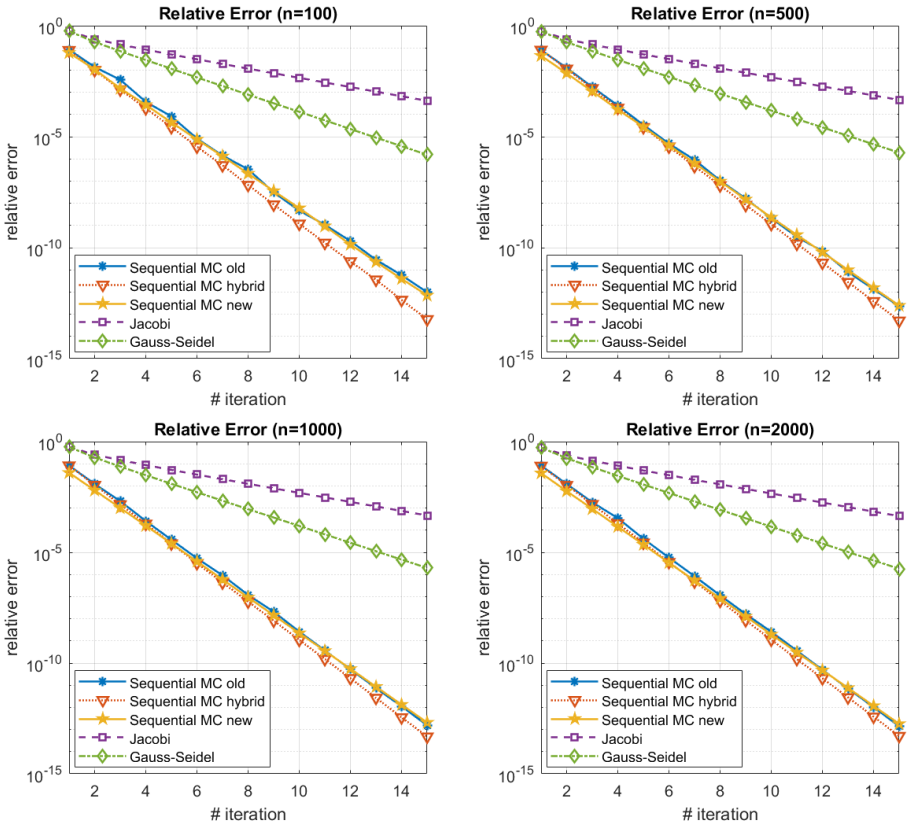



**Figure 12.** Relative error of random dense matrices $n \times n$ for $n = 100$ (top left), $n = 500$ (top right), $n = 1000$ (bottom left) and $n = 2000$ (bottom right).

## 6. Discussion

In this section we will discuss the most important features of the new algorithm. We will discuss its advantages over the original algorithm. As can be seen the advantage of the new algorithm versus the old one depends on the newly presented parameter in this paper—the so called dominancy number.

We may observe the following key elements of the presented algorithms.

- Performance enhancement through strategic relaxation parameter selection, leading to reduced computation time and relative errors within a fixed number of iterations.
- Optimization potential by balancing the iteration matrix, demonstrating improvements in the algorithm's original form.
- The introduced random variable serves as an unbiased estimator for the solution vector. The stochastic error's structure is analyzed in detail.
- The analysis of the stochastic error structure reveals that the convergence rate of the proposed WE Monte Carlo algorithm is influenced by matrix balancing. Although this influence is less pronounced compared to deterministic algorithms, it is still a factor to consider.
- Application of a sequential Monte Carlo method by John Halton, based on the control variate method, for further optimization .
- Analysis of the relative stochastic error structure, revealing a dependency on matrix balancing for the convergence rate of the WE algorithm.
- Comparative analysis of numerical outcomes with the most optimal deterministic method, the Preconditioned Conjugate Gradient (PCG), indicating superior results for the WE Monte Carlo method in certain cases.
- The proposed WE Monte Carlo algorithm, in conjunction with the sequential Monte Carlo for computing all solution components, demonstrates significantly faster convergence compared to certain well-known deterministic iterative methods, such as the Jacobi and Gauss–Seidel methods.
- The proposed WE Monte Carlo algorithm, in conjunction with the sequential Monte Carlo for computing all solution components, shows faster convergence compared to the old algorithm when the dominancy number is close to 1. This trend is consistent across matrices of various sizes, with a more pronounced effect observed for larger matrices.

Now, when we consider the impact of the dominancy number on the performance of algorithms, particularly the comparison between a new and an old algorithm, the following dynamics come into play:

The dominancy number reflects the diagonally dominance in the system. As this number increases, it implies that specific elements or features have a more substantial influence on the overall system behavior.

The old algorithm might have been designed or optimized based on certain assumptions or characteristics of the problem. However, if the dominancy number is low, the impact of the dominance may not have been fully exploited or addressed by the old algorithm.

The new algorithm, on the other hand, may have been specifically developed or adapted to handle scenarios with higher dominancy numbers. It could incorporate techniques that capitalize on the influence of diagonally dominance. As the dominancy number increases, the advantages of these tailored approaches become more apparent.

The observed performance difference between the old and new algorithms is indicative of the latter's adaptability and efficacy in scenarios where certain factors play a more critical role. The new algorithm may dynamically adjust or scale its strategies based on the prominence of diagonally dominance, leading to improved outcomes in these situations.

In essence, the correlation between the dominancy number and algorithmic performance emphasizes the importance of algorithmic design that considers and leverages the influential factors within a given problem. As the dominancy number grows, the advantages of the new and the hybrid algorithm specifically crafted to handle such conditions

become increasingly evident, showcasing the importance of adaptability and specialization in algorithmic solutions.

However, there is another issue to be addressed. The need for increased simulation trajectories to achieve convergence, especially when dealing with imbalanced matrices, raises questions about efficiency and practicality, particularly in comparison to canonical methods.

To address this critical point, it is possible to apply a number of strategies to achieve a balance between convergence speed and the number of simulation trajectories:

- *Adaptive Trajectory Sampling*: By implementing an adaptive sampling technique, the algorithm can dynamically adjust the number of trajectories based on the convergence rate. When the algorithm is far from convergence, fewer trajectories can be used. As it approaches the solution, the algorithm can increase the number of trajectories to fine-tune the accuracy. This adaptive approach helps in reducing unnecessary computational overhead in the early stages of the algorithm.
- *Parallel Computing and Optimization*: The inherently parallel nature of Monte Carlo simulations makes our WE algorithm particularly amenable to parallel computing techniques. By distributing the simulation trajectories across multiple processors or a GPU, we can significantly reduce the overall CPU time. Furthermore, optimizing the code for specific hardware architectures can also lead to substantial improvements in computational efficiency.
- *Hybrid Methodologies*: Combining the WE algorithm with deterministic methods can offer a compromise between the robustness of Monte Carlo approaches and the speed of traditional methods. Starting with a deterministic method to reach a closer approximation to the solution and then switching to the WE algorithm for fine-tuning could reduce the total number of simulation trajectories required.
- *Improving the Dominancy Number Calculation*: As our research highlights the importance of the dominancy number in the algorithm's performance, refining the calculation or estimation of this number can lead to a more efficient trajectory planning. This could involve developing predictive models that estimate the optimal dominancy number based on matrix characteristics.
- *Advanced Stochastic Techniques*: Incorporating advanced stochastic techniques, such as importance sampling or variance reduction strategies, could improve the efficiency of the simulation trajectories. These techniques aim to focus computational effort on the most critical parts of the solution space, thereby reducing the number of trajectories required for a given level of accuracy.

To summarize, while the increase in CPU time due to additional simulation trajectories is a valid concern, especially for imbalanced matrices, the strategies outlined above provide viable pathways to optimize the trade-off between convergence speed and computational effort. Our future work will focus on further investigating and implementing these strategies to enhance the practical applicability of the WE algorithm.

## 7. Conclusions

Linear systems hold significance not just for their mathematical elegance but for their pervasive influence across various disciplines, serving as a foundational language to articulate and comprehend the complexities of the world. Linear Algebraic Equations are more than mere problems to solve; they act as a vital link between abstract theories and practical realities in scientific and engineering applications. Whether navigating simulations in computational fluid dynamics or optimizing structures in engineering, they offer a mathematical framework to perceive, understand, and tackle intricate challenges. As computational capabilities expand, addressing large-scale linear algebraic problems has become a focal point, pushing the boundaries of both analytical techniques and computational hardware.

In our research, we have introduced and thoroughly investigated a novel Monte Carlo method designed explicitly for large systems of algebraic linear equations. This innovative approach is based on the "Walk on Equations" Monte Carlo model, a recent development attributed to Ivan Dimov, Sylvain Maire, and Jean Michel Sellier. Additionally, we present

a new hybrid algorithm, a combination of the two methods, introducing a crucial element—the dominancy number—for algorithmic performance. Notably, our study breaks new ground by providing a comparative analysis with the Jacobi and Gauss–Seidel algorithms, focusing on matrices of considerable dimensions—an aspect largely unexplored in existing scholarly discussions.

To validate our approach, we conduct numerical experiments, including a significant case involving a large system from finite element approximation in structural mechanics, specifically addressing a beam structure problem. Our work contributes to advancing the understanding and effective solution of large-scale linear algebraic problems, showcasing the relevance and potential applications of our proposed Monte Carlo method in practical, real-world scenarios.

Looking ahead to the future landscape of scholarly endeavors in this field, we anticipate a comprehensive and rigorous evaluation that juxtaposes traditional computational methods with our cutting-edge Monte Carlo technique. Moreover, we need to investigate how the proposed method change to nonlinearities, and what possible criteria can be done in this case. Additionally, we aim to broaden the scope of our innovative algorithm by applying it to various matrices drawn from the well-known Harwell-Boeing collection. This initiative seeks to underscore the adaptability and effectiveness of our approach in solving linear algebraic equations crucial to a variety of practical, real-world scenarios.

**Author Contributions:** Conceptualization, V.T. and I.D.; methodology, V.T. and I.D.; software, V.T. and I.D.; validation, V.T.; formal analysis, V.T. and I.D.; investigation, V.T.; resources, V.T.; data curation, V.T.; writing—original draft preparation, V.T.; writing—review and editing, V.T.; visualization, V.T; supervision, V.T. and I.D.; project administration, V.T. and I.D.; funding acquisition, V.T. and I.D. All authors have read and agreed to the published version of the manuscript.

**Funding:** The methodology for investigation is supported by the Project BG05M2OP001-1.001-0004 UNITe, funded by the Operational Programme "Science and Education for Smart Growth", co-funded by the European Union trough the European Structural and Investment Funds. The development of the stochastic methods is supported by the Bulgarian National Science Fund (BNSF) under Project KP-06-N52/5 "Efficient methods for modeling, optimization and decision making". The computational experiments are supported by the BNSF under Project KP-06-N52/2 "Perspective Methods for Quality Prediction in the Next Generation Smart Informational Service Networks". The simulations are supported by the BNSF under Project KP-06-N62/6 "Machine learning through physics-informed neural networks".

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Data are contained within the article.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Dimov, I.T.; Maire, S.; Sellier, J.M. A New Walk on Equations Monte Carlo Method for Linear Algebraic Problems. *Appl. Math. Model.* **2015**, *39*, 4494–4510. [CrossRef]
2. Zheng, Z.C.; Ren, G.X.; Wang, W.J. A reduction method for large scale unsymmetric eigenvalue problems in structural dynamics. *J. Sound Vib.* **1997**, *199*, 253–268. [CrossRef]
3. Stüben, K. A review of algebraic multigrid. *J. Comput. Appl. Math.* **2001**, *128*, 281–309. [CrossRef]
4. Audu, K.J.; Essien, J.N. An Accelerated Iterative Technique: Third Refinement of Gauss–Seidel Algorithm for Linear Systems. *Comput. Sci. Math. Forum* **2023**, *7*, 7. [CrossRef]
5. Lukyanenko, D. Parallel Algorithm for Solving Overdetermined Systems of Linear Equations, Taking into Account Round-Off Errors. *Algorithms* **2023**, *16*, 242. [CrossRef]
6. Ishii, G.; Yamamoto, Y.; Takaishi, T. Acceleration and Parallelization of a Linear Equation Solver for Crack Growth Simulation Based on the Phase Field Model. *Mathematics* **2021**, *9*, 2248. [CrossRef]
7. Georgiev, S.; Vulkov, L. Parameters Identification and Numerical Simulation for a Fractional Model of Honeybee Population Dynamics. *Fractal Fract.* **2023**, *7*, 311. [CrossRef]
8. Georgiev, S.G.; Vulkov, L.G. Coefficient identification in a SIS fractional-order modelling of economic losses in the propagation of COVID-19. *J. Comput. Sci.* **2023**, *69*, 102007. [CrossRef] [PubMed]

9. Atanasov, A.Z.; Georgiev, S.G.; Vulkov, L.G. Reconstruction analysis of honeybee colony collapse disorder modeling. *Optim. Eng.* **2021**, *22*, 2481–2503. [CrossRef]
10. Georgiev, I.K.; Kandilarov, J. An immersed interface FEM for elliptic problems with local own sources. *AIP Conf. Proc.* **2009**, *1186*, 335–342. [CrossRef]
11. Veleva, E.; Georgiev, I.R. Seasonality of the levels of particulate matter PM10 air pollutant in the city of Ruse, Bulgaria. *AIP Conf. Proc.* **2020**, *2302*, 030006. [CrossRef]
12. Boutchaktchiev, V. Inferred Rate of Default as a Credit Risk Indicator in the Bulgarian Bank System. *Entropy* **2023**, *25*, 1608. [CrossRef]
13. Traneva, V.; Tranev, S. Intuitionistic Fuzzy Model for Franchisee Selection. In *Intelligent and Fuzzy Systems. INFUS 2022*; Kahraman, C., Tolga, A.C., Cevik Onar, S., Cebi, S., Oztaysi, B., Sari, I.U., Eds.; Lecture Notes in Networks and Systems; Springer: Cham, Switzerland, 2022; Volume 504, pp. 632–640. [CrossRef]
14. Traneva, V.; Tranev, S. Intuitionistic Fuzzy Two-factor Variance Analysis of Movie Ticket Sales. *J. Intell. Fuzzy Syst.* **2021**, *42*, 563–573. [CrossRef]
15. Dimov, I.; Tonev, O. Performance Analysis of Monte Carlo Algorithms for Some Models of Computer Architectures. In *International Youth Workshop on Monte Carlo Methods and Parallel Algorithms—Primorsko*; Sendov, B., Dimov, I., Eds.; World Scientific: Singapore, 1990; pp. 91–95.
16. Spanier, J.; Gelbard, E. *Monte Carlo Principles and Neutron Transport Problem*; Addison-Wesley: Boston, MA, USA, 1969.
17. Curtiss, J.H. Monte Carlo methods for the iteration of linear operators. *J. Math Phys.* **1954**, *32*, 209–232. [CrossRef]
18. Curtiss, J.H. *A Theoretical Comparison of the Efficiencies of Two Classical Methods and a Monte Carlo Method for Computing One Component of the Solution of a Set of Linear Algebraic Equations*; John Wiley and Sons: Hoboken, NJ, USA, 1956; pp.191–233.
19. Arafat, A.; El-Mikkawy, M. A Fast Novel Recursive Algorithm for Computing the Inverse of a Generalized Vandermonde Matrix. *Axioms* **2023**, *12*, 27. [CrossRef]
20. Alexandrov, V.; Atanassov, E.; Dimov, I. Parallel Quasi-Monte Carlo Methods for Linear Algebra Problems. *Monte Carlo Methods Appl.* **2004**, *10*, 213–219.
21. Dimov, I. Monte Carlo Algorithms for Linear Problems. *Pliska (Studia Math. Bulg.)* **2000**, *13*, 57–77.
22. Dimov, I.; Tonev, O. Monte Carlo algorithms: Performance analysis for some computer architectures. *J. Comput. Appl. Math.* **1993**, *48*, 253–277. [CrossRef]
23. Dimov, I.; Alexandrov, V.; Karaivanova, A. Parallel resolvent Monte Carlo algorithms for linear algebra problems. *J. Math. Comput. Simul.* **2001**, *55*, 25–35. [CrossRef]
24. IDimov, T.; Karaivanova, A.N. Iterative Monte Carlo algorithms for linear algebra problems. In *Numerical Analysis and Its Applications, Proceedings of the First Workshop on Numerical Analysis and Applications, Rousse, Bulgaria, 24–27 June 1996*; Springer Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1997; Volume 1196, pp. 150–160.
25. Dimov, I.T.; Alexandrov, V. A New Highly Convergent Monte Carlo Method for Matrix Computations. *Math. Comput. Simul.* **1998**, *47*, 165–181. [CrossRef]
26. Hammersley, J.M.; Handscomb, D.C. *Monte Carlo Methods*; John Wiley & Sons, Inc.: New York, NY, USA; London, UK; Sydney, Australia; Methuen, MA, USA, 1964.
27. Shyamkumar, N.; Banerjee, S.; Lofgren, P. Sublinear estimation of a single element in sparse linear systems. In Proceedings of the 2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton), Monticello, IL, USA, 27–30 September 2016.
28. Choudalakis, S.; Mitrouli, M.; Polychronou, A.; Roupa, P. Solving High-Dimensional Problems in Statistical Modelling: A Comparative Study. *Mathematics* **2021**, *9*, 1806. [CrossRef]
29. Hached, M.; Jbilou, K.; Koukouvinos, C.; Mitrouli, M. A Multidimensional Principal Component Analysis via the C-Product Golub–Kahan–SVD for Classification and Face Recognition. *Mathematics* **2021**, *9*, 1249. [CrossRef]
30. Angelova, V.; Hached, M.; Jbilou, K. Sensitivity of the Solution to Nonsymmetric Differential Matrix Riccati Equation. *Mathematics* **2021**, *9*, 855. [CrossRef]
31. Delgado, J.; Peña, G.; Peña, J.M. Green Matrices, Minors and Hadamard Products. *Axioms* **2023**, *12*, 774. [CrossRef]
32. Zhou, W.; Xiong, Z.; Qin, Y. Forward Order Law for the Reflexive Inner Inverse of Multiple Matrix Products. *Axioms* **2022**, *11*, 123. [CrossRef]
33. Li, M.; Feng, Y.; Wang, G. Estimating Failure Probability with Neural Operator Hybrid Approach. *Mathematics* **2023**, *11*, 2762. [CrossRef]
34. Abud, T.P.; Augusto, A.A.; Fortes, M.Z.; Maciel, R.S.; Borba, B.S.M.C. State of the Art Monte Carlo Method Applied to Power System Analysis with Distributed Generation. *Energies* **2023**, *16*, 394. [CrossRef]
35. Tomczyk, K. Monte Carlo-Based Procedure for Determining the Maximum Energy at the Output of Accelerometers. *Energies* **2020**, *13*, 1552. [CrossRef]
36. Dimov, I.T. *Monte Carlo Methods for Applied Scientists*; World Scientific: Hackensack, NJ, USA; London, UK; Singapore, 2008; 291p, ISBN-10 981-02-2329-3.
37. Golub, G.V.; Loon, C.F.V. *Matrix Computations*, 3rd ed.; Johns Hopkins Univ. Press: Baltimore, MD, USA, 1996.
38. Dimov, I.T.; Philippe, B.; Karaivanova, A.; Weihrauch, C. Robustness and applicability of Markov chain Monte Carlo algorithms for eigenvalue problems. *Appl. Math. Model.* **2008**, *32*, 1511–1529. [CrossRef]

39. Maire, S. Reducing variance using iterated control variates. *J. Stat. Comput. Simul.* **2003**, *73*, 1–30. [CrossRef]
40. Straßburg, J.; Alexandrov, V.N. A Monte Carlo Approach to Sparse Approximate Inverse Matrix Computations. *Procedia Comput. Sci.* **2013**, *18*, 2307–2316. [CrossRef]
41. Sobol, I.M. *Monte Carlo Numerical Methods*; Nauka: Moscow, Russia, 1973.
42. Dimov, I.; Karaivanova, A. Parallel computations of eigenvalues based on a Monte Carlo approach. *J. Monte Carlo Method Appl.* **1998**, *4*, 33–52. [CrossRef]
43. Dimov, I.; Karaivanova, A. *A Power Method with Monte Carlo Iterations, Recent Advances in Numerical Methods and Applications*; Iliev, O., Kaschiev, M., Sendov, B., Vassilevski, P., Eds.; World Scientific: Singapore, 1999; pp. 239–247.
44. Kantorovich, L.W.; Krylov, V.I. *Approximate Methods of Higher Analysis*; Interscience: New York, NY, USA, 1964.
45. Densmore, J.D.; Larsen, E.W. Variational variance reduction for particle transport eigenvalue calculations using Monte Carlo adjoint simulation. *J. Comput. Phys.* **2003**, *192*, 387–405. [CrossRef]
46. Rosca, N. A New Monte Carlo Estimator for Systems of Linear Equations. *Stud. Univ. Babes-Bolyai Math.* **2006**, *51*, 97–107.
47. Forsythe, G.E.; Leibler, R.A. Matrix Inversion by a Monte Carlo Method. *Math. Comput.* **1950**, *4*, 127–129. [CrossRef]
48. Dimov, I. Minimization of the Probable Error for Some Monte Carlo methods. In Proceedings of the International Conference on Mathematical Modeling and Scientific Computation, Albena, Bulgaria, 23–28 September 1990; Publishing House of the Bulgarian Academy of Sciences: Sofia, Bulgaria, 1991; pp. 159–170.
49. Halton, J. Sequential Monte Carlo. *Math. Proc. Camb. Philos. Soc.* **1962**, *58*, 57–78. [CrossRef]
50. Halton, J. *Sequential Monte Carlo*; University of Wisconsin: Madison, WI, USA, 1967; Mathematics Research Center Technical Summary Report No. 816, 38p.
51. Halton, J.; Zeidman, E.A. *Monte Carlo Integration with Sequential Stratification*; University of Wisconsin: Madison, WI, USA, 1969; Computer Sciences Department Technical Report No. 61, 31p.
52. Halton, J. Sequential Monte Carlo for linear systems—A practical summary. *Monte Carlo Methods Appl.* **2008**, *14*, 1–27. [CrossRef]
53. Gobet, E.; Maire, S. Sequential control variates for functionals of Markov processes. *SIAM J. Numer. Anal.* **2005**, *43*, 1256–1275 . [CrossRef]
54. Kollman, C.; Baggerly, K.; Cox, D.; Picard, R. Adaptive importance sampling on discrete Markov chains. *Ann. Appl. Probab.* **1999**, *9*, 391–412.
55. L'Ecuyer, P.; Blanchet, J.H.; Tuffin, B.; Glynn, P.W. Asymptotic robustness of estima- tors in rare-event simulation. *ACM Trans. Model. Comput. Simul.* **2010**, *20*, 6.
56. L'Ecuyer, P.; Tuffin, B. Approximate zero-variance simulation. In Proceedings of the 2008 Winter Simulation Conference, Miami, FL, USA, 7–10 December 2008; pp. 170–181.
57. Saad, Y. *Iterative Methods for Sparse Linear Systems*, 2nd ed.; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2003; p. 195, ISBN 978-0-89871-534-7.
58. Westlake, J.R. *A Handbook of Numerical Matrix Inversion and Solution of Linear Equations*; John Wiley & Sons, Inc.: New York NY, USA; London, UK; Sydney, Australia, 1968.
59. Golub, G.H.; Ye, Q. Inexact Preconditioned Conjugate Gradient Method with Inner-Outer Iteration. *SIAM J. Sci. Comput.* **1999**, *21*, 1305. [CrossRef]
60. Knyazev, A.V.; Lashuk, I. Steepest Descent and Conjugate Gradient Methods with Variable Preconditioning. *SIAM J. Matrix Anal. Appl.* **2008**, *29*, 1267. [CrossRef]
61. Website: Matrix Market. NOS4: Lanczos with Partial Reorthogonalization. Finite Element Approximation to a Beam Structure. Available online: http://math.nist.gov/MatrixMarket/data/Harwell-Boeing/lanpro/nos4.html (accessed on 1 January 2004).