Research Paper
Computational Methods for Differential Equations
http://cmde.tabrizu.ac.ir
Vol. 10, No. 2, 2022, pp. 489-501
DOI:10.22034/cmde.2021.37369.1654

An adaptive Monte Carlo algorithm for European and American options

Mahboubeh Aalaei* **and Mahnaz Manteqipour**
Insurance Research Center, Saadat Abad, Tehran, Iran.

Abstract

In this paper, a new adaptive Monte Carlo algorithm is proposed to solve systems of linear algebraic equations (SLAEs). The corresponding properties of the algorithm and its advantages over the conventional and previous adaptive Monte Carlo algorithms are discussed and theoretical results are established to justify the convergence of the algorithm. Furthermore, the algorithm is used to solve the SLAEs obtained from finite difference method for the problem of European and American options pricing. Numerical tests are performed on examples with matrices of different sizes and on SLAEs coming from option pricing problems. Comparisons with standard numerical and stochastic algorithms are also done which demonstrate the computational efficiency of the proposed algorithm.

Keywords. Adaptive Monte Carlo algorithm, Finite difference method, Black Scholes model, European and American put option.
2010 Mathematics Subject Classification. 65C05, 65F10, 65N06.

1. Introduction

Systems of linear algebraic equations (SLAEs) are arisen from many scientific and engineering problems [1, 12]. High dimensional SLAEs can be obtained directly or after discretization of partial differential equations (PDEs) from real world problems [2, 3].Therefore it is a problem of unquestionable importance to choose an appropriate approach for solving SLAEs.
The problem of pricing an option as a financial derivative can be solved using the famous Black-Scholes PDE model which can be cast as a set of SLAEs using finite difference (FD) method. The Black-Scholes model is a convenient way to calculate the price of an option and the FD approach, which is a popular approach in option pricing, normally produce results of reasonable accuracy (see e.g. [13, 16] and the references cited therein).
The stochastic solution of SLAEs can be given using Monte Carlo algorithms via generating random paths where the mathematical expectation of the stochastic estimate is the desired solution [6]. Monte Carlo methods can be used in different areas and have some significant advantages. For example, they can approximate individual components of the solution without calculating the whole solution vector, [10]. Also, for a large sparse SLAEs, they are more efficient than direct or iterative numerical methods, [14] and they are good candidates for parallelization because of the fact that many independent sample paths are used to estimate the solution, [1].
In spite of all advantages, the conventional Monte Carlo method converges slowly. Halton was the first one that proposed adaptive Monte Carlo methods in [5], which improve the convergence of the Monte Carlo method exponentially and after that, some research papers worked on this problem (see e.g [6–8, 14]).
In this paper, we propose an adaptive Monte Carlo algorithm for SLAEs. The proposed adaptive Monte Carlo (PAMC) algorithm converges exponentially and improves the convergence of both the conventional and adaptive Monte Carlo methods. PAMC algorithm has a faster convergence rate and needs to generate fewer random paths than adaptive algorithm presented in [10]. Theoretical results are established to justify the convergence of the algorithm and the results of test model problems demonstrate that the PAMC algorithm achieves much faster convergence than the

Received: 16 December 2019 ; Accepted: 20 April 2021.
∗ Corresponding author. Email: aalaei@irc.ac.ir.

conventional Monte Carlo method does.
Furthermore, to confirm the efficiency of the PAMC algorithm, it is applied to approximate the value of European and American options. According to our best knowledge, Monte Carlo methods have been widely applied to option pricing and other financial problems (see e.g. [7, 11]). But evaluating the European and American options price based on the PAMC algorithm has been investigated for the first time in this paper.
The remainder of this article is organized as follows. The conventional Monte Carlo (CMC), Halton adaptive Monte Carlo and PAMC algorithms are described in section 2 and the convergence and properties of PAMC algorithm are discussed. The European and American options pricing are described in section 3. Numerical tests are performed on examples with matrices of different sizes and on systems coming from option pricing problem in section 4. Comparisons with standard numerical and stochastic algorithms are also done which demonstrate the computational efficiency of the PAMC algorithm. Our conclusions are given in section 5.

2. Adaptive monte carlo algorithm for solving SLAEs

Monte Carlo algorithms have proved to be a valuable and flexible computational tool in modern finance and have been developed within the past years to price the options. Also, It is well known that Monte Carlo methods are more effective and more preferable than direct and iterative numerical methods for solving large SLAEs. In this chapter, we present, propose and analyze adaptive Monte Carlo algorithms for solving the SLAEs which can be used to solve linear systems obtained form finite difference for option pricing. We then proceed to analyze the convergence of the proposed algorithm and discuss its corresponding properties. The algorithm has simple structure, low cost, desirable speed and accuracy. Consider we are going to solve the SLAEs

$$Bx = f, \tag{2.1}$$

using Monte Carlo algorithms. Introducing $A = \{A_{ij}\}_{i,j=1}^{n} = I - B_1$, where I is an identity matrix, $B_1 = DB$ and $F = Df$ where

$$D = diag(d_1, \dots, d_n),$$

we have $x = Ax + F$ and therefore using recursive formula

$$x^{(k+1)} = Ax^{(k)} + F, \tag{2.2}$$

we have an estimator for x under the assumption $max_i \sum_{j=1}^{n} |A_{ij}| < 1$ and the following Monte Carlo algorithms converge. In all following algorithms, independent random paths of Markov chain will be simulated with initial distribution $p = (p_1, \dots, p_n)$ and transition matrix P with following properties:

2.1. $p_i > 0$ if $h_i \neq 0$,
2.2. $P_{ij} > 0$ if $A_{ij} \neq 0, i, j = 1, \dots, n$.

In this paper, the transition matrix is computed as follows:

$$P_{ij} = \frac{|A_{ij}|}{\sum_{j=1}^{n} |A_{ij}|}, i = 1, \dots, n, j = 1, \dots, n.$$

In this section, for a better understanding the differences between algorithms, we discuss the conventional Monte Carlo method and the Halton adaptive Monte Carlo algorithm presented in [10] and then the new adaptive Monte Carlo algorithm is proposed. In the following algorithms Z, k are the number and the length of random paths respectively.

2.1. **Conventional Monte Carlo algorithm.** The base of the conventional Monte Carlo method is to express each component of the solution vector as the expectation of some random variable. To estimate the inner product of two vectors h and x obtained from Eq. (2.2), calculate θ_k via the following algorithm which is an unbiased estimator of the inner product $\langle h, x^{(k+1)} \rangle$, [15].

Algorithm 1. Conventional Monte Carlo algorithm.
Computing component x_t of the solution vector x.
1. **Input** initial data (Initialization): the matrix B, the vector f, the number of random paths N, the length of

```
random paths $k$, $h = (\underbrace{0, \cdots, 1}_{t}, 0, \cdots, 0)'$.
2. Preliminary calculations (preprocessing):
    2.1. **Compute** the matrix $A = I - DB$ where
         $$D = diag(\frac{1}{B_{11}}, \frac{1}{B_{22}}, \cdots, \frac{1}{B_{nn}}).$$
    2.3. **Compute** the transition probability matrix $P = \{p_{ij}\}_{ij}^{n}$, where
         $$p_{ij} = \frac{|A_{ij}|}{\sum_{j=1}^{n} |A_{ij}|}, i = 1, 2, \dots, n, j = 1, 2, \dots, n.$$
3. **for** $s = 1$ **to** $Z$ **do**
    3.1.Generate random paths $i_0^{(s)} \to i_1^{(s)} \to \cdots \to i_k^{(s)}$.
    3.2.**set** $w_0^{(s)} = 1$.
    3.3. **for** $m = 1$ **to** $k$ **do**
      3.3.1. **set** $w_m^{(s)} = w_{m-1}^{(s)} \frac{A_{i_{m-1}^{(s)} i_m^{(s)}}}{P_{i_{m-1}^{(s)} i_m^{(s)}}}$
      3.3.2. **set** $\eta_k^{(s)}(h) = \frac{h_{i_0^{(s)}}}{p_{i_0^{(s)}}} \sum_{m=0}^{k} w_m^{(s)} F_{i_m^{(s)}}$
      3.3.3. **set** $\theta_k(h) = \frac{1}{Z} \sum_{s=1}^{Z} \eta_k^{(s)}(h)$
    3.4. **enddo**
4. **enddo**
```

2.2. **Halton adaptive Monte Carlo algorithm.** For Halton adaptive Monte Carlo algorithm [10], Consider

$$F^{(0)} = F, \theta_k^{(0)} = 0, F^{(d)} = F^{(d-1)} - B_1 \theta_k^{(d-1)}, d = 1, \cdots, r,$$

where r is the number of stages and $\theta_k^{(d)}$ is the approximate solution of

$$B_1 \Delta^d x = F^{(d)}, \tag{2.3}$$

using described conventional Monte Carlo method which random paths are generated through a fixed transition matrix P. Then

$$\varphi_k^{(d)}(h) = \varphi_k^{(d-1)}(h) + \theta_k^{(d)}(h),$$

is the approximated solution of SLAE (2.1). It is shown in [10] that

$$\lim_{r\to\infty} F^{(r)} = 0, \lim_{r\to\infty} \theta_k^{(r)} = 0, \lim_{r\to\infty} \varphi_k^{(r)} = x_j,$$

where x_j is the j th component of the exact solution to 2.1. Note that if $r = 1$, we have the conventional Monte Carlo method.

We will skip some parts of the initialization and preprocessing steps which are the same as in the previous subsection.

Algorithm 2. Halton Adaptive Monte Carlo algorithm.

```
    **Computing all components of the solution vector** $x$.
1. Initialization. **Set** $F^{(0)} = F, \theta_k^{(0)} = 0, \varphi_k^{(0)} = 0$ and $r$ the number of stages.
2. Preprocessing.
3. **for** $d = 1$ **to** $r$ **do**
    3.1. **Set** $F^{(d)} = F^{(d-1)} - B\theta_k^{(d-1)}$.
    3.2. **for** $t = 1$ **to** $n$ **do**
    3.3. **Set** $h = (\underbrace{0, \cdots, 1}_{t}, 0, \cdots, 0)'$
      3.3.1 **for** $s = 1$ **to** $Z$ **do**
        3.3.1.1. Generate random paths $t \to i_1^{(s)} \to \cdots \to i_k^{(s)}$.
        3.3.1.2. set $w_0^{(s)} = 1$.
        3.3.1.3. for $m = 1$ to $k$ do
          3.3.1.3.1. set $w_m^{(s)} = w_{m-1}^{(s)} \frac{A_{i_{m-1}^{(s)} i_m^{(s)}}}{P_{i_{m-1}^{(s)} i_m^{(s)}}}$
        3.3.1.4. enddo
        3.3.1.5. set $\eta_k^{(d,s)}(h) = \sum_{m=0}^{k} w_m^{(s)} F_{i_m^{(s)}}^{(d)}$
      3.3.2. enddo
      3.3.3. set $\theta_k^{(d)}(h) = \frac{1}{Z} \sum_{s=1}^{Z} \eta_k^{(d,s)}(h)$
    3.4. enddo
    3.5. Update $\varphi_k^{(d)} = \varphi_k^{(d-1)} + \theta_k^{(d)}$ where $\theta_k^{(d)} = \{\theta_k^{(d)}(h)\}_{t=1}^{n}$.
4. enddo
```

2.3. **Proposed Adaptive Monte Carlo algorithm.** In PAMC algorithm, the constant $\gamma \in (0,1]$ is used to accelerate the convergence, (see e.g. [6]). It can be optimazed using the following formula:

$$\gamma = 1 - \left(\frac{\mu}{1+\sqrt{1+\mu^2}}\right)^2, \tag{2.4}$$

where μ is the spectral radius of matrix A, [9]. Furthermore, instead of generating random paths in each stage, the same random paths can be used for all stages. It means that the transition matrix P is fixed for all stages and we do not need to generate Z random paths with length k and calculate $w_m^{(S)}$ for each stage because they are fixed for all stages.Then the total number of random paths with length k to estimate each component of the solution vector in PAMC algorithm is Z. So the total number of random variables in PAMC algorithm is nkZ. Also The presented algorithm in [14] generates the same random paths for all components of the solution vector. Therefore the total number of random variables is at least rnZ. Therefore the proposed algorithm in this paper needs less random variables if $k < r$ comparitive to algorithm presented in [14] and it can be less time consuming. Furthermore it has simple structure, low cost, desirable speed and accuracy and easy to be parallelized. The PAMC algorithm properties will be discussed in test numerical results.

Algorithm 3. Proposed Adaptive Monte Carlo algorithm (PAMC).
Computing all components of the solution vector x.

```
1. Initialization. Set $F^{(0)} = F, \theta_k^{(0)} = 0, \varphi_k^{(0)} = 0$ and $r$ the number of stages.
2. Preprocessing.
    2.1. Compute optimal $\gamma$ using Eq. (2.4), the vector $F_\gamma = Df$ and the matrix $A^\gamma = I - DB$ where $D = diag(\frac{\gamma}{B_{11}}, \frac{\gamma}{B_{22}}, \cdots, \frac{\gamma}{B_{nn}})$.
    2.3. Compute the transition probability matrix $P$.
3. for $t = 1$ to $n$ do
    3.1 for $s = 1$ to $Z$ do
      3.1.1. Generate random paths $t \to i_1^{(s)} \to \cdots \to i_k^{(s)}$.
      3.1.2. set $w_0^{(t,s)} = 1$.
      3.1.3. for $m = 1$ to $k$ do
        3.1.3.1. set $w_m^{(t,s)} = w_{m-1}^{(t,s)} \frac{A^\gamma_{i_{m-1}^{(s)} i_m^{(s)}}}{P_{i_{m-1}^{(s)} i_m^{(s)}}}$.
      3.1.4. enddo
    3.2. enddo
4. enddo
5. for $d = 1$ to $r$ do
    5.1 Set $F_\gamma^{(d)} = F_\gamma^{(d-1)} - B^\gamma \theta_k^{(d-1)}$, where $B^\gamma = I - A^\gamma$.
    5.2. for $t = 1$ to $n$ do
```

```
        5.2.1 Set $h = (\underbrace{0, \cdots, 1}_{t}, 0, \cdots, 0)'$.
        5.2.2 for $s = 1$ to $Z$ do
          5.2.2.1. set $\eta_k^{(d,s)}(h) = \frac{h_{i_0^{(s)}}}{p_{i_0^{(s)}}} \sum_{m=0}^{k} w_m^{(s)} F_{\gamma_{i_m^{(s)}}}^{(d)}$.
        5.2.3 enddo
        5.2.4. set $\theta_k^{(d)}(h) = \frac{1}{Z} \sum_{s=1}^{Z} \eta_k^{(d,s)}(h)$.
      5.3. enddo
      5.4. Update $\varphi_k^{(d)} = \varphi_k^{(d-1)} + \theta_k^{(d)}$ where $\theta_k^{(d)} = \{\theta_k^{(d)}(h)\}_{t=1}^{n}$.
6. enddo
```

2.4. **Convergence.** To examine the convergence of Algorithm 3, we should define some notations as follows: Consider $F_\gamma^{(0)} = F_\gamma, \Delta^0 x = x$ and Eq. (2.1) for stage r as

$$B^\gamma \Delta^r x = F_\gamma^{(r)}, \tag{2.5}$$

where $\Delta^r x$ and $F_\gamma^{(r)}$ are obtained by the following recursive equations

$$\Delta^r x = \Delta^{r-1} x - \Delta_k^{r-1} x,$$
$$F_\gamma^{(r)} = F_\gamma^{(r-1)} - B^\gamma \Delta_k^{r-1} x,$$

and $\Delta_k^r x$ is the approximate solution of SLAE (2.1) obtained by using Eq. (2.2), k times. Considering $S_0 = \Delta_k^0 x$ and $S_r = S_{r-1} + \Delta_k^r x$, clearly we have

$$x = S_r + \Delta^{r+1} x, \tag{2.6}$$

and the following theorem will be proven.

Theorem 2.1. *Under the assumption $\|A^\gamma\| < 1$ and $\Delta_0^r x = 0$, $\lim_{r\to\infty} \Delta^r x = 0$.*

Proof. From Eq. (2.2) and (2.3), we have

$$\Delta^r x = A^\gamma \Delta^r x + F_\gamma^{(r)}$$
$$\Delta_k^r x = A^\gamma \Delta_(k-1)^r x + F_\gamma^{(r)}.$$

Then we can obtain

$$\Delta^r x = \Delta^{r-1} x - \Delta_{k-1}^{r-1} x = A^\gamma (\Delta^{r-1} x - \Delta_{k-1}^{r-1} x) = \cdots = (A^\gamma)^k \Delta^{r-1} x,$$

and

$$\Delta^r x = (A^\gamma)^k \Delta^{r-1} x = (A^\gamma)^{2k} \Delta^{r-2} x = (A^\gamma)^{3k} \Delta^{r-3} x = \cdots = (A^\gamma)^{(r-1)k} x.$$

Therefore

$$\|\Delta^r x\| \leq \|(A^\gamma)^{(r-1)k}\|.\|x\|, \tag{2.7}$$

since $\|A^\gamma\| < 1$, then $B^\gamma = I - A^\gamma$ is invertible and has a unique solution. Therefore $\|x\|$ is finite. Taking the limit of Eq. (2.7), the proof is completed. □

Theorem 2.2. *Under the assumptions $\|A\| < 1$ and $\Delta_0^r x = 0$, as r tends to infinity S_r converges to x, geometrically, [7].*

Theorem 2.2 shows that the numerical method which the proposed adaptive Monte Carlo method is based on, is converged to the solution vector x.

Theorem 2.3. *Az Z tends to infinity, $\theta_k^{(r)}$ converges to $\Delta_k^r x$.*

Proof. Since the random variable $\eta_k^{(r,s)}(h)$ is defined along the path $t \to i_1^{(s)} \to \ldots \to i_k^{(s)}$ we have

$$E[\eta_k^{(r,s)}(h)] = \sum_{i_k^{(s)}}^{n} \cdots \sum_{t}^{n} \eta_t^{(r,s)}(h) P_{ti_1^{(s)}} \cdots P_{i_{k-1}^{(s)} i_k^{(s)}},$$

which, together with the formulas in Algorithm 3, gives

$$\begin{aligned} E[\eta_k^{(r,s)}(h)] &= E[\sum_{m=t}^{k} w_m^{(s)} F_{\gamma_{i_m^{(s)}}}^{(r)}] \\ &= \sum_{i_0=t}^{n} \cdots \sum_{i_k^{(s)}=1}^{n} A_{ti_1^{(s)}}^{\gamma} \cdots A_{i_{m-1}^{(s)} i_m^{(s)}}^{\gamma} F_{\gamma_{i_m^{(s)}}}^{(r)} P_{i_m^{(s)} i_{m+1}^{(s)}} \cdots P_{i_{k-1}^{(s)} i_k^{(s)}} \\ &= \sum_{m=t}^{k} \sum_{i_1^{(s)}=1}^{n} \cdots \sum_{i_m^{(s)}=1}^{n} A_{i_0^{(s)} i_1^{(s)}}^{\gamma} \cdots A_{i_{m-1}^{(s)} i_m^{(s)}}^{\gamma} F_{\gamma_{i_m^{(s)}}}^{(r)}. \end{aligned}$$

The last equation is obtained using the property $\sum_{j=1}^{n} P_{ij} = 1$ and we immediately obtain

$$E[\eta_k^{(r,s)}(h)] = \langle h, \sum_{m=0}^{k} (A^{\gamma})^m F_{\gamma}^{(r)} \rangle = \langle h, \Delta_t^r x \rangle$$

and therefore as Z tends to infinity, $\theta_k^{(r)} = \frac{1}{Z} \sum_{s=1}^{Z} \eta_k^{(r,s)}$ converges to $\Delta_k^r x$.

Theorem 2.4. *As k and r tend to infinity, $\varphi_k^{(r)}$ converges to x.*

Proof. Consider $S_r = \sum_{d=1}^{r} \Delta_k^d x$. From equation (8) we have

$$x = \sum_{d=1}^{r} \Delta_k^d x + \Delta^{r+1} x.$$

From theorem 2.3, we have $\lim_{Z\to\infty} \theta_k^{(d)} = \Delta_k^d x$, for $d = 1, \ldots, r$. It will be concluded that $\varphi_k^{(r)} = \sum_{d=1}^{r} \theta_k^{(d)}$ converges to x as k and r tend to infinity from theorem 2.1. □

3. OPTION PRICING

An option is a financial instrument that gives one the right, but not obligation, to buy or sell underlying asset at a specified price (the strike) by a predetermined date which is known as the maturity.
An option to buy some security is called a call option, while an option to sell is put option. A simply distinction for options can be made between American options, which policyholders have the right to alter the contract before its natural termination and European options, which exercise is admitted only at contract expiration.

The well known Black Scholes model for an European put option can be described by formula [7]:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + (r-q)S\frac{\partial V}{\partial S} - rV = 0, \tag{3.1}$$

with final condition $V(S,T) = max(E - S, 0)$ and boundary conditions $V(0,t) = Ee^{-r(T-t)}$and $V(S,t) \approx 0$ as $S \to \infty$, where S, E, T, r are the current price of asset, the strike price, the expiry time and the risk free interest rate, respectively. Also, S is assumed to behave $dS = (r-q)Sdt + \sigma S dW$, where dW is a Wiener process, r and σ are the drift rate and the volatility of the asset, respectively. In this case, there is the closed form solution. But, for more styles of options, there are not the closed form solutions. Stochastic methods can be used to price these options. In this regard, the adaptive Monte Carlo algorithm can be used to value European options and pricing formulas for these

options can be checked using this method.
The finite difference (FD) method can be used to approximate the solution of (3.1) where $\theta \in (0,1)$ is the parameter of discretization. Considering $V_{ij} = V(i\Delta_S, j\Delta_t), 0 < i < N, 0 \leq j \leq M$, the Black Scholes model can be formulated as the following SLAEs:

$$CV^{j+1} = DV^j + b^j, \tag{3.2}$$

where

$$C = \begin{bmatrix} 1-\theta m_1 & -\theta u_1 & 0 & \cdots & 0 \\ -\theta d_2 & 1-\theta m_2 & -\theta u_2 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & -\theta d_{N-1} & 1-\theta m_{N-1} \end{bmatrix},$$

$$b^j = \begin{bmatrix} \theta d_1 V_{0j} + (1-\theta) d_1 V_{0j+1} \\ 0 \\ \vdots \\ 0 \\ \theta u_{N-1} V_{Nj} + (1-\theta) u_{N-1} V_{Nj+1} \end{bmatrix},$$

$$D = \begin{bmatrix} 1+(1-\theta)m_1 & (1-\theta)u_1 & 0 & \cdots & 0 \\ (1-\theta)d_2 & 1+(1-\theta)m_2 & (1-\theta)u_2 & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & (1-\theta)d_{N-1} & 1+(1-\theta)m_{N-1} \end{bmatrix},$$

where $d_i = \frac{\Delta_t \sigma^2 S_i^2}{2\Delta_S^2} - \frac{\Delta_t (r-q) S_i}{2\Delta_S}, m_i = -\frac{\Delta_t \sigma^2 {S_i}^2}{\Delta_S^2} - \Delta_t r, u_i = \frac{{\Delta_t}^2 {S_i}^2}{2\Delta_S^2} + \frac{\Delta_t (r-q) S_i}{2\Delta_S}$. The linear system obtained in each time step will be approximated using the PAMC algorithm and at the end, the solution vector will be the price of the European option.
To describe an American put option using the Black Scholes model, we assume $V(S,t) \geq max(E-S,0)$, final condition $V(S,T) = max(E-S,0)$ and boundary conditions $V(0,t) = Ee^{-r(T-t)}$and $V(S,t) \approx 0$ as $S \to \infty$.
For American option pricing, the SLAE (3.2) should be solved in each time step and the solution vector should be compared with the final condition and the result will be the solution vector in that time step. The solution vector will be calculated for $j = M, \ldots, 0$ and at the end, the solution vector will be the price of the American option. The SLAE (3.2) will be solved using the PAMC algorithm.

4. Numerical Test Results

In this section, we report numerical results using PAMC algorithm to solve the SLAEs, European and American put options. We note that in all examples, $\theta = \frac{1}{2}$. Also, in all examples, the starting stage of PAMC algorithm is the solution obtained by using CMC algorithm and the results of these two algorithms can be compared together.

4.1. **The solution of SLAEs.**

In this subsection, we obtain solutions of linear equations using the proposed adaptive Monte Carlo algorithm and compare the results with [7]. If x is the exact solution of SLAE and $x^{(r)}$ is the approximate solution using the adaptive Monte Carlo method at stage r, the L_2 absolute estimate will be

$$\|x - x^{(r)}\| = (\sum_{i=1}^{n} (x_i - {x_i}^{(r)})^2)^{\frac{1}{2}}$$

when the exact solution x is not known, then we use the following formula as the absolute error of estimation,

$$(\sum_{i=1}^{n}(x_i{}^{(r)} - \sum_{j=1}^{n} A_{ij}x_j{}^{(r)} - F_i)^2)^{\frac{1}{2}}.$$

Example 4.1. Consider a dense diagonally dominant SLAE with random components

$$B(i,j) = \begin{cases} \rho_{ij} & \text{if } i \neq j \\ \sum_{j=1, j\neq i}^{n} \rho_{ij} + 20 \times r_i & \text{if } i = j \end{cases}$$

where ρ_{ij} and r_i are random numbers uniformly distributed in $(0,1)$ and $x_i = \frac{i}{n}$. we assume $k = 10, Z = 100$.
The absolute errors are calculated for different γs. The results are shown on Figure 1. One can observe that for PAMC algorithm with optimal γ, 0.8511, the convergence is much faster and after 35 iterations, the absolute error is about 10^{-14}. At the same time, the convergence of algorithm with $\gamma = 0.5$ and $\gamma = 0.8$ are slower and after 35 iterations, the absolute errors are about 10^{-8} and 10^{-12} respectively. It is clear that PAMC give better results because the norm of matrix A is reduced using optimal γ. The norm will be 0.9301, 0.8881 and 0.8810 for γ parameters 0.5, 0.8 and optimal one, respectively. Furthermore, to observe the convergence behavior of PAMC algorithm, the logarithm

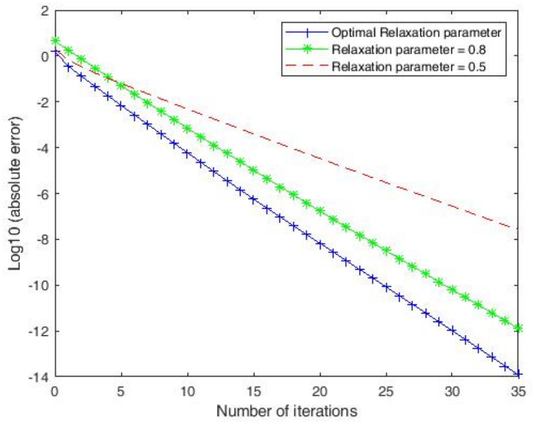


FIGURE 1. Comparison of PAMC algorithm for different relaxation parameters.

of absolute error for matrices with sizes 100×100 and 300×300 with respect to the number of simulated random paths are shown on Figure 2. One can see that increasing the number of simulated random paths can improve the convergence of PAMC algorithm, as it is clear from the theory.

Example 4.2. Consider a linear system with

$$A_{ij} = \frac{\rho_{ij} r_i}{\sum_{k=1}^{n} \rho_{ik}}$$

where $r_i = c + \rho_i(d-c)$ and $c = min_i \sum_{j=1}^{n} A_{ij}$, $d = max_i \sum_{j=1}^{n} A_{ij} = \|A\|$ and ρ_i and ρ_{ij} are pseudo-random numbers uniformly distributed in (0, 1) and $F_i = i$, [14]. We consider c = 0.25, d = 0.75, $k = 10$ and $Z = 20$.
A convergence comparison of the PAMC and conjugate gradient (CG) iterative method is presented in the Figures 2 and 3. In CG method, the relative tolerance for the residual error is assumed 10^{-20} and the maximum allowable number of iterations is assumed 50 and the starting point is assumed $x_i{}^0 = 10$, for $i = 1, \ldots, n$. The results presented in Figures 3 and 4 show that the convergence for PAMC is faster than CG and for the smaller number of iterations, the error of PAMC algorithm is significently less than CG method. This will be more visible when the size of the matrix is increased. This example is verfired the obtained results in [6].

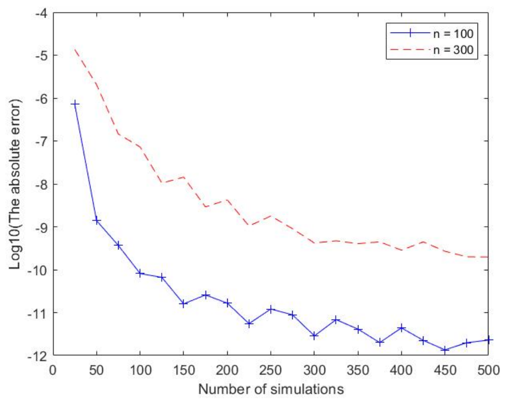


FIGURE 2. The convergence behavior of PAMC algorithm with respect to the number of simulated paths.

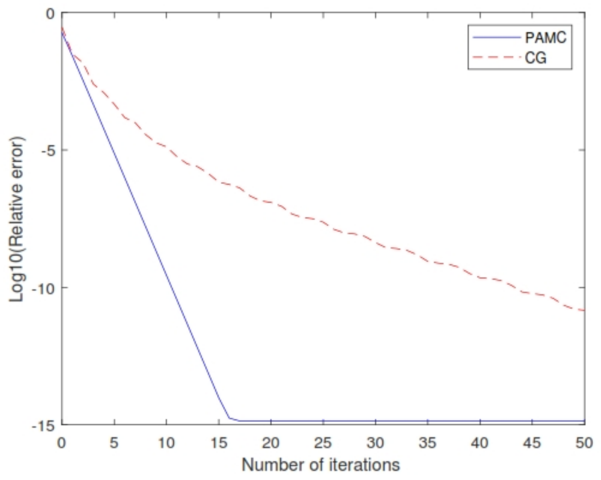


FIGURE 3. The logarithm of relative error for PAMC algorithm with matrix size 1000 $\times$ 1000.

4.2. European option pricing.

Example 4.3. Consider an European put option with $E = 1, T = 1, r = 0.1, \sigma = 0.1, q = 0, S_{min} = 0, S_{max} = 4, N = 100, M = 100$, [13].
Figures 5 and 6 illustrate the exact and the PAMC solution of European put options choosing a fine grid for T and S_{max} with $\Delta T = \frac{T}{M}$ and $\Delta_S = \frac{S_{max}-S_{min}}{N}$. Results manifest that the price obtained by the PAMC algorithm has acceptable accuracy in the whole of the domain.
Also, the comparisons are furnished in Table 1 It is observed from numerical results that the proposed algorithm is competitive with the ones proposed in [13].

Example 4.4. Consider an European put option with $E = 10, T = 0.5, r = 0.05, \sigma = 0.2, q = 0, S_{min} = 0, S_{max} = 20, N = 200, M = 500$, where $k = 15, Z = 20$ and $r = 20$, [7].
We obtained the following numerical results. The values obtained by the PAMC algorithm and those obtained by the Black Scholes formula and the difference between them as the absolute error are shown in Table 2. Furthermore, the results have been compared with adaptive Monte Carlo (AMC) algorithm in [7], which the results obtained by PAMC

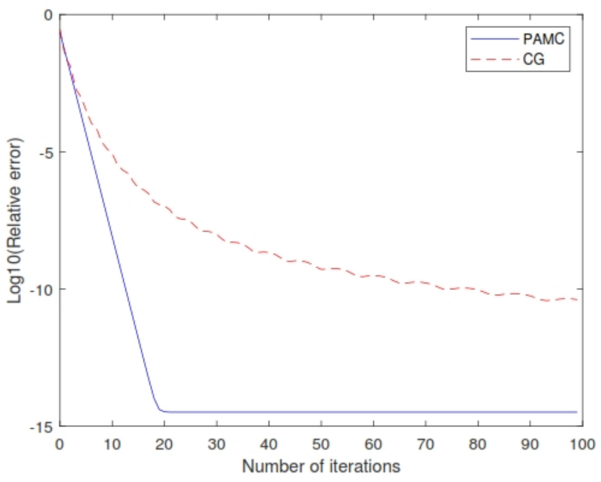
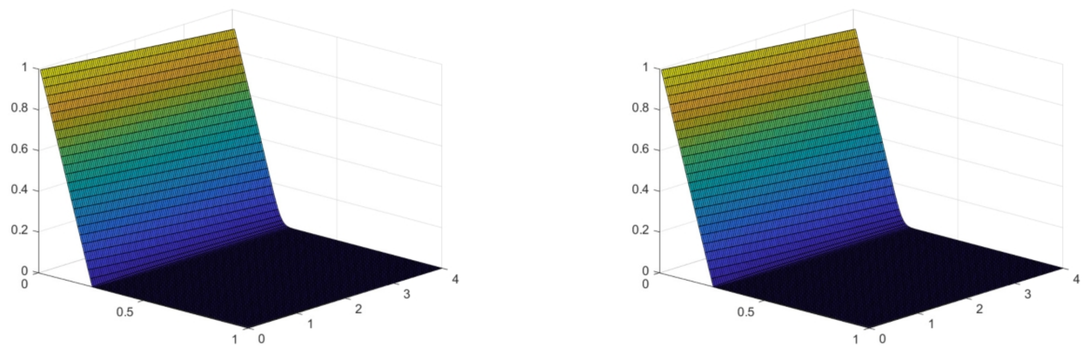


FIGURE 4. The logarithm of relative error for PAMC algorithm with matrix size 5000 × 5000.

TABLE 1. Results of comparison for different time steps

Numerical parameters	M=200	M=500	M=1000
The absolute error	8.5891×10^{-4}	2.2854×10^{-5}	7.2425×10^{-6}



(A) PAMC solution (B) Black Scholes solution

FIGURE 5. Price surface via maturity and stock price using the PAMC algorithm (left) and the Black Scholes model (right)

are better than AMC.

4.3. **American option pricing.**

Example 4.5. Consider an European put option with $E = 100, T = 3, r = 0.08, \sigma = 0.2, S_{min} = 0, S_{max} = 2 \times S, N = 300, M = 100$.

We obtained the following numerical results. The values obtained by the PAMC algorithm and the Black Scholes formula are shown in Table 3. Also, RMSE denotes the root mean squared absolute error and the true value is computed

TABLE 2. Camparisons of Adaptive Monte Carlo algorithms for European Options

Asset	Black Scholes	AMC in [7]	Error in [7]	PAMC	Error
2	7.753099120	7.753099119	1.2699×10^{-9}	7.753099120	5.0595×10^{-11}
3	6.753099120	6.753099119	1.2370×10^{-9}	6.753099120	5.0651×10^{-11}
4	5.753099120	5.753099369	2.4941×10^{-7}	5.753099120	2.3100×10^{-11}
5	4.753099342	4.753102615	3.2730×10^{-6}	4.753099402	5.9078×10^{-8}
6	3.753180620	3.753302476	1.2185×10^{-4}	3.753185327	4.7073×10^{-6}
7	2.756835269	2.757692583	8.5731×10^{-4}	2.756871741	3.6479×10^{-5}
8	1.798714599	1.798755250	4.0650×10^{-5}	1.798710805	3.7938×10^{-6}

by the binomial tree method where the length of each time step is 0.0001 years, [4].

TABLE 3. Results of comparison for American put option

q	S	True	PAMC (M=500)	PAMC (M=1000)	PAMC (M=2000)
	80	20.000	20.000	20.000	20.000
	90	11.697	11.685	11.692	11.697
0.00	100	6.932	6.917	6.925	6.931
	110	4.155	4.143	4.151	4.155
	120	2.510	2.499	2.506	2.510
	80	20.350	20.343	20.347	20.350
	90	13.497	13.491	13.494	13.497
0.04	100	8.944	8.939	8.941	8.944
	110	5.912	5.904	5.909	5.912
	120	3.898	3.890	3.895	3.898
	80	22.205	22.195	22.200	22.205
	90	16.207	16.198	16.203	16.207
0.08	100	11.704	11.698	11.701	11.704
	110	8.367	8.360	8364	8.367
	120	5.930	5.922	5.927	5.930
	80	25.658	25.651	25.655	25.658
	90	20.083	20.075	20.079	20.083
0.12	100	15.498	15.492	15.496	15.498
	110	11.803	11.795	11.800	11.803
	120	8.886	8.878	8.882	8.886
RMSE			0.008	0.003	0.000

Example 4.6. Consider an American put option with $E = 50, T = 1, r = 0.1, \sigma = 0.3, q = 0, S_{min} = 0, S_{max} = 2 \times S, N = 200, M = 500$.

The values obtained by the proposed algorithm and those obtained by Binomial tree, Projective Successive Over-relaxation (PSOR) and Explicit methods considered in [17] are shown in Table 4.

5. CONCLUSIONS

In this paper, we have proposed an adaptive Monte Carlo algorithm to solve SLAE with less random number generation and therefore more efficient than adaptive Monte Carlo algorithms which have been introduced before in

TABLE 4. Comparison of our algorithm with other methods in [17] for an American put option price.

Stock Price	Binomial	PSOR	Explicit	PAMC
25	25.0001	25.00000	25.00000	25.00000
30	20.0001	20.00000	20.00000	20.00000
35	15.00011	15.00000	15.00000	15.00000
40	10.13439	10.13345	10.13400	10.13342
45	6.56042	6.55925	6.56000	6.55810
50	4.16877	4.16781	4.16849	4.16860
55	2.60434	2.60345	2.60407	2.60422
60	1.60399	1.60305	1.60356	1.60314
65	0.97647	0.97574	0.97617	0.97638
70	0.58955	0.58846	0.58885	0.58931
75	0.35371	0.35192	0.35234	0.35319

[7, 8, 14]. We have analyzed the convergence, speed up and efficiency of the algorithm in the case of dealing with matrices with different sizes. It is clear that the PAMC algorithm and algorithm in [7] converges exponentially, but the number of generated random numbers and therefore the corresponding calculations are reduced in PAMC algorithm. Also, the PAMC algorithm is compared with conjugate gradient (CG) method and the obtained results show that the convergence of PAMC algorithm is faster than CG method and for the smaller number of iterations, the use of PAMC algorithm is strongly advised.
Furthermore, the proposed algorithm has been implemented to solve sparse matrices which are arising from the discretization of parabolic partial differential equation arising from option pricing. The results show the efficiency and accuracy of the PAMC algorithm for pricing European and American options. All the examples have been compared with other methods and demonstrate the computational efficiency of the PAMC algorithm.
Numerical results showed a stable and efficient way for valuing put options. Moreover, the results led us to the extension of the PAMC algorithm to several other financial models such as pricing other options, high dimensional partial differential equations (PDEs) and partial integro-differential equations that have applications in financial derivative pricing and risk management in the forthcoming works. Also, in order to improve the accuracy of the solution, instead of using conventional finite difference method, high order semi-discretization schemes (see e.g. [13]) can be used to discretize the financial problems and the obtained SLAEs can be solved with PAMC algorithm.

REFERENCES

[1] V. N. Alexandrov, C. G. Martel, and J. Straßburg, *Monte Carlo scalable algorithms for computational finance*, Procedia Computer Science, *4* (2011), 1708–1715.

[2] A. R. Bacinello, *Regression-based algorithms for life insurance contracts with surrender guarantees*, Quantitative Finance, *10*(9) (2010), 1077–1090.

[3] D. Bauer, A. Kling, and J. Rub, *A universal pricing framework for guaranteed minimum benefits in variable annuities*, ASTIN Bulletin, *38* (2008), 621–651.

[4] M. H. Chiang, H. H. Fu, Y. T. Huang, C. L. Lo, and P. T. Shih, *Analytical Approximations for American Options: The Binary Power Option Approach*, Journal of Financial Studies, *26*(3) (2018).

[5] M. Dehghan and M. Hajarian, *SSHI methods for solving general linear matrix equations*, Engineering Computations, 2012.

[6] I. Dimov, S. Maire, and J. M. Sellier, *A new walk on equations Monte Carlo method for solving systems of linear algebraic equations*, Appl. Math. Model, *39* (2015), 4494–4510

[7] R. Farnoosh and M. Aalaei, *New adaptive Monte Carlo algorithm for parallel solution of large linear systems with applications*, Proceedings of the Romanian Academy Series A, *16*(1) (2015), 11–19.

[8] R. Farnoosh, M. Aalaei, and M. Ebrahimi, *Combined probabilistic algorithm for solving high dimensional problems*, Stochastics An International Journal of Probability and Stochastic Processes, *87*(1) (2015), 30–47.

[9] W. Hackbusch, *Iterative Solution of Large Sparse Systems of Equations*, Applied Mathematical Sciences, *95* (2016).

[10] J. Halton, *Sequential Monte Carlo*, Proceedings of the Cambridge Philosophical Society, *58* (1962), 57–78.

[11] C. H. Han and Y. Lai, *A smooth estimator for MC/QMC methods in finance*, Mathematics and Computers in simulation, *81*(3) (2010), 536–550.

[12] A. Jasra and P. D. Moral, *Sequential Monte Carlo methods for option pricing*, Stochastic analysis and applications, *29* (2011), 292–316.

[13] F. Kiyoumarsi, *European and American put valuation via a high-order semi-discretization scheme*, Computational Methods for Differential Equations, *6*(1) (2018), 63–79.

[14] Y. Lai, *Adaptive Monte Carlo methods for matrix equations with applications*, Journal of Computational and Applied Mathematics, *231* (2009), 705–714.

[15] R. Y. Rubinstein, *Simulation and the Monte Carlo method*, Wiley, New York, 1981.

[16] D.Y. Tangman, A. Gopaul, and M. Bhuruth, *A fast high-order finite difference algorithm for pricing American options*, J. Comput. Appl. Math., *222* (2008), 17–29.

[17] I. Vidid, *Numerical methods for option pricing*, Master Thesis, Master in Advanced Computing for Science and Engineering, Politecnica University, 2012.