THE PUBLISHING HOUSE
OF THE ROMANIAN ACADEMY

PROCEEDINGS OF THE ROMANIAN ACADEMY, Series A,
Volume 16, Number 1/2014, pp. 11–19

NEW ADAPTIVE MONTE CARLO ALGORITHM FOR PARALLEL SOLUTION OF LARGE LINEAR SYSTEMS WITH APPLICATIONS

Rahman FARNOOSH [1], Mahboubeh AALAEI [2]

[1] Iran University of Science and Technology, School of Mathematics, Narmak, Tehran 16844, Iran
[2] Insurance Research Center, Saadat Abad, Tehran, Iran.
E-mail: rfarnoosh@iust.ac.ir

In this paper, a new adaptive Monte Carlo algorithm is proposed for solving large linear systems. The proposed algorithm converges much faster than the conventional Monte Carlo and achieves exponential convergence. The corresponding properties of the algorithm are discussed. It has simple structure, low cost, desirable speed and accuracy. Theoretical results are established to justify the convergence of the algorithm. To confirm the accuracy and efficiency of the proposed algorithm, it is used to solve large linear systems. The adaptive Monte Carlo algorithms are implemented for parallel solution of large linear systems on parallel machine with Message Passing Interface as inter node communication. Furthermore, we provide applications of the algorithm for option pricing, where the Black Scholes formula is converted to linear systems using discretization.

Key words: adaptive Monte Carlo algorithm, large linear systems, parallel solution, option pricing.

1. INTRODUCTION

High dimensional linear systems of algebraic equations are arising from real world problems [1, 2, 13]. The large linear systems can be obtained directly or after discretization of partial differential or integral equations [3, 5, 6]. Therefore the choice of an appropriate approach for solving large sparse linear systems is a problem of unquestionable importance in many scientific and engineering applications.

One of the well known stochastic methods which is preferable for solving high dimensional linear system of algebraic equations is the Monte Carlo method. Monte Carlo algorithms 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 linear system of algebraic equations, they are more efficient than direct or iterative numerical methods [15] and they are good candidates for parallelization because of the fact that many independent sample paths are used to estimate the solution [2].

In spite of all advantages, conventional Monte Carlo method converges slowly. Halton was the first one that proposed adaptive Monte Carlo methods in [7], which improve the convergence of Monte Carlo method exponentially. These methods have been used in different areas by the researchers [12].

In this paper, we improve the adaptive Monte Carlo algorithm for linear systems. The new adaptive algorithm has faster convergence rate and needs to generate less random paths than previous 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 new adaptive Monte Carlo method achieves much faster convergence than the conventional Monte Carlo method does. Also, Monte Carlo methods have been known for their embarrassingly parallel nature [15]. The adaptive Monte Carlo algorithms are implemented in the programming language C using the Message Passing Interface (MPI).

Furthermore, to confirm the efficiency of the proposed algorithm, the adaptive Monte Carlo methods are applied to approximate the value of European options. This algorithm can be applied to options other than European style. According to our best knowledge, Monte Carlo methods have been widely applied to option pricing and other financial problems [8, 9, 11]. But evaluating the European option price based on the proposed algorithm has been investigated for the first time in this paper. Furthermore, it can be applied to solve high dimensional partial differential equations or partial integro-differential equations. These equations have applications in financial derivative pricing and risk management, see [4].

The remainder of this article is organized as follows. The conventional and adaptive Monte Carlo algorithms are described in Section 2. The new adaptive Monte Carlo algorithm is proposed in Section 3 and the convergence and its properties are discussed. Numerical test results for solving linear systems using adaptive Monte Carlo methods and parallel computing are given in Section 4 . Also, the proposed adaptive Monte Carlo algorithm is applied to value options. Our conclusions are given in Section 5.

2. MONTE CARLO METHODS

It is well known that Monte Carlo methods are more effective and more preferable than direct and iterative numerical methods for solving large sparse systems. In this section, we are going to solve the system of linear equations

$$Bx = F, \tag{1}$$

using conventional and adaptive Monte Carlo methods. Introducing $A = \{A_{ij}\}_{i,j=1}^{n} = I - B$, where I is an identity matrix, we have $x = A\,x + F$ and therefore using recursive formula

$$x^{(k+1)} = Ax^{(k)} + F, \tag{2}$$

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.

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 vector h and $x^{(k+1)}$ obtained from Eq. (2), for an arbitrary integer $k > 0$, simulate Z independent random paths $i_0^{(s)} \to i_1^{(s)} \to \ldots i_k^{(s)}, s = 1, \ldots, Z$ of Markov chain with initial distribution $p = (p_1, \ldots, p_n)$ and transition matrix P the with following properties:

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

Define $\eta_k^{(s)}(h) = \frac{h_{i_0}}{p_{i_0}^{(s)}} \sum_{m=0}^{k} w_m^{(s)} f_{i_m}^{(s)}$, $s = 1, \ldots, Z$, where

$$w_m^{(s)} = w_{m-1}^{(s)} \frac{A_{i_{m-1}^{(s)} i_m^{(s)}}}{P_{i_{m-1}^{(s)} i_m^{(s)}}}, w_0^{(s)} \equiv 1. \tag{3}$$

We calculate $\theta_k = \frac{1}{Z} \sum_{s=1}^{Z} \eta_k^{(s)}(h)$, which is an unbiased estimator of the inner product $< h, x^{(k+1)} >$ [14].

2.2. Adaptive Monte Carlo (Halton) algorithm

Consider $F^{(0)} = F,\ \theta_k^{(0)} = 0, F^{(d)} = F^{(d-1)} - B\theta_k^{(d-1)}, d = 1, \ldots, r$, where r is the number of stages and $\theta_k^{(d)}$ is the approximate solution of

$$B\Delta^d x = F^{(d)}, \tag{4}$$

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

$$\varphi_k^{(d)}(h) = \varphi_k^{(d-1)}(h) + \theta_k^{(d)}, \quad \varphi_k^{(0)}(h) = 0,$$

is the approximated solution of linear system (1). It is shown in [7] that

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

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

3. NEW ADAPTIVE MONTE CARLO ALGORITHM

In this section, a new adaptive Monte Carlo algorithm is proposed and the pseudocode is given in Algorithm 1.

In the adaptive Monte Carlo method, the transition matrix P is fixed for all stages. Therefore we can use the same random paths generated through the transition matrix P for all of them. It means that we do not need to calculate $w_m^{(s)}$ defined in Eq. (3) for each stage because they are fixed for all stages. Here, the notation of $w(t, m, s)$ is used instead of $w_m^{(s)}$ to estimate x_t where $1 \le t \le n$, $1 \le m \le k$ and $1 \le s \le Z$.

Algorithm 1: New adaptive Monte Carlo algorithm

Step 1. Input: B, F, k, Z and r. Set $F^{(0)} = 0, \theta_k^{(0)} = 0, \varphi_k^{(0)} = 0$.

Step 2. Generating random walks:

For $t = 1, \cdots, n$ do:

For $s = 1, \cdots, Z$ do:

Generate $t \to i_1^{(s)} \to \cdots \to i_k^{(s)}$,

For $m = 1, \cdots, k$ do: $w(t, m, s) = w(t, m-1, s) \dfrac{A_{i_{m-1}^{(s)} i_m^{(s)}}}{P_{i_{m-1}^{(s)} i_m^{(s)}}}$, $w(t, 0, s) = 1$.

Step 3. Approximating the solution:

For $d = 1, \cdots, r$ do:

$F^{(d)} = F^{(d-1)} - B\theta_k^{(d-1)}$,

For $t = 1, \cdots, n$ do:

$h = (\underbrace{0, \cdots, 0, 1}_{t}, \cdots, 0)'$,

For $s = 1, \cdots, Z$ do:

$\eta_k^{(d,s)}(h) = \sum_{m=0}^{k} w(t, m, s) F_{i_m^{(s)}}^{(d)}$,

$\theta_k^{(d)} = \dfrac{1}{Z} \sum_{s=1}^{Z} \eta_k^{(d,s)}$, where $\eta_k^{(d,s)} = \{\eta_k^{(d,s)}(h)\}_{t=1}^{n}$,

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

where $\varphi_k^{(r)} = \{\varphi_k^{(r)}(h)\}_{t=1}^{n}$ is the approximated solution of linear system (1) in stage r.

In this algorithm, instead of generating random paths in each stage, we use the same random paths in all stages. It means that we do not need to generate rZ random paths with length k. The total number of random paths to estimate each component of the solution vector in our proposed algorithm is Z. Then the total number of random variables in proposed algorithm is knZ.

The algorithm presented in [10] generates the same random paths for all components of the solution vector. The length of random paths should be at least n. Therefore the total number of random variables is at least rnZ.

Therefore, if the length of random paths in our algorithm is assumed to be smaller than the number of refinement iterations, i.e. $k < r$, our algorithm needs less random variables and it can be less time consuming. This will be discussed in the numerical test results. Furthermore it has simpler structure, low cost, desirable speed and accuracy.

3.1. Convergence

To examine the convergence of Algorithm 1, we should define some notations as follows:
Consider $F^{(0)} = F, \Delta^0 x = x$ and Eq. (4) for stage r as

$$B\Delta^r x = F^{(r)}, \tag{5}$$

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

$$\Delta^r x = \Delta^{r-1}x - \Delta_k^{r-1}x, \quad ; \quad F^{(r)} = F^{(r-1)} - B\Delta_k^{r-1}x,$$

and $\Delta_k^r x$ is the approximate solution of linear system (1) obtained by using Eq. (2), k times. Considering $S_0 = \Delta_k^0 x$ and $S_r = S_{r-1} + \Delta_k^r x$, clearly we have

$$x = S_r + \Delta^{r+1}x \tag{6}$$

and the following theorem will be proven.

THEOREM 3.1. *Under the assumption* $\|A\| < 1$ *and* $\Delta_0^r x = 0$, S_r *converges to x, geometrically.*

Proof. According to Eq. (6), it is enough to prove $\lim_{r\to\infty} \Delta^r x = 0$. From Eq. (2) and (5), we have

$$\Delta^r x = A\Delta^r x + F^{(r)}, \quad ; \quad \Delta_k^r x = A\Delta_{k-1}^r x + F^{(r)}.$$

Then we can obtain

$$\Delta^{r+1}x = \Delta^r x - \Delta_k^r x = A(\Delta^r x - \Delta_{k-1}^r x) = \cdots = A^k(\Delta^r x - \Delta_0^r x) = A^k(\Delta^{r-1}x - \Delta_k^{r-1}x),$$

and then

$$\left\|\Delta^{r+1}x\right\| \le \|A\|^k . \left\|\Delta^{r-1}x - \Delta_k^{r-1}x\right\| \le \cdots \le \|A\|^{rk} . \left\|x - \Delta_k^0 x\right\| \tag{7}$$

Taking the limit of Eq. (7), the proof is completed because $\|A\| < 1$.

In the next theorem, we will prove that $\varphi_k^{(r)}$ is the approximate solution of the system (1).

THEOREM 3.2. *Az k and r tends to infinity,* $\varphi_k^{(r)}$ *converges to* x. *It means*

$$\lim_{k\to\infty} \lim_{r\to\infty} E(\varphi_k^{(r)}(h)) =< h, x >= x_t.$$

Proof. It is enough to show $E(\eta_k^{(r,s)}(h)) =< h, \Delta_k^r x >$, and therefore $E(\theta_k^{(r)}(h)) =< h, \Delta_k^r x >$ and $E(\varphi_k^{(r)}(h)) =< h, S_r >$. Since the random variable $\eta_k^{(r,s)}(h)$ is defined along the path $i_0^{(s)} \to i_1^{(s)} \to \ldots \to i_k^{(s)}$, we have

$$E[\eta_k^{(r,s)}(h)] = \sum_{i_0^{(s)}}^{n} \cdots \sum_{i_k^{(s)}}^{n} \eta_k^{(r,s)}(h) p_{i_0^{(s)}} P_{i_0^{(s)} i_1^{(s)}} \cdots P_{i_{k-1}^{(s)} i_k^{(s)}},$$

which, together with the formulas in Algorithm 1, gives

$$E[\eta_k^{(r,s)}(h)] = E[\sum_{m=0}^{k} w(t,m,s) F_{i_m^{(s)}}^{(r)}] = \sum_{i_0^{(s)}}^{n} \cdots \sum_{i_k^{(s)}}^{n} \sum_{m=0}^{k} A_{i_0^{(s)} i_1^{(s)}} A_{i_1^{(s)} i_2^{(s)}} \cdots A_{i_{m-1}^{(s)} i_m^{(s)}} F_{i_m^{(s)}}^{(r)} P_{i_m^{(s)} i_{m+1}^{(s)}} \cdots P_{i_{k-1}^{(s)} i_k^{(s)}}. \quad (8)$$

Using the property $\sum_{j=1}^{n} P_{ij} = 1$, Eq. (8) can be written as

$$= \sum_{m=0}^{k} \sum_{i_0^{(s)}}^{n} \cdots \sum_{i_k^{(s)}}^{n} A_{i_0^{(s)} i_1^{(s)}} A_{i_1^{(s)} i_2^{(s)}} \cdots A_{i_{m-1}^{(s)} i_m^{(s)}} F_{i_m^{(s)}}^{(r)},$$

and we immediately obtain

$$E[\eta_k^{(r,s)}(h)] =< h, \sum_{m=0}^{k} A^m F^{(r)} >=< h, \Delta_k^r x > .$$

4. NUMERICAL TEST RESULTS

4.1. High dimensional linear systems

In this subsection, we report numerical results of solving linear equations using new adaptive Monte Carlo algorithm. If x is the exact solution of the linear system and $x^{(r)}$ is the approximate solution using the adaptive Monte Carlo method at stage r, the L_2 absolute estimate will be

$$\left\| x - x^{(r)} \right\| = (\sum_{i=1}^{n} (x_i - x_i^{(r)})^2)^{1/2}.$$

But the exact solution x is not known, therefore we use the following formula as the absolute error,

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

Example 1. Suppose a linear system with

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

where $r_i = c + \rho_i (d - c), c = \min_i \sum_{j=1}^{n} A_{ij}, d = \max_i \sum_{j=1}^{n} A_{ij} = \|A\|$ and ρ_i, ρ_{ij} are pseudo-random numbers uniformly distributed in (0, 1), and $F_i = i$, [10]. We consider $c = 0.25$, $d = 0.75$.

The algorithm is implemented to matrices with size 1000 and 3000. The results are shown in Table 1. It is clear that the algorithm converges exponentially. Considering the approximate absolute errors for this example, the presented algorithm for adaptive Monte Carlo method in [10] does not converge, while the proposed adaptive algorithm converges exponentially.

The proposed adaptive algorithms are using independent random paths. Therefore, the calculations on these paths can be done in parallel. They are implemented in the programming language C using the MPI for communications between parallel processors. In parallelization of these algorithms, coefficient matrix and right hand side vector are given to all of CPUs such that probability matrix is constructed in them. All the computations are done in different CPUs because the stochastic paths are independent. The Communication between CPUs is essential for obtaining a solution vector in each stage.

Here, all tests were executed over a cluster of workstation environment, with 8 nodes Intel Xeon 2.8 GHz, 4GB RAM. Time (in seconds) statistics of Halton and the proposed algorithms for large linear

systems of different orders with different number of processors are given in Table 2. It is obvious that the proposed algorithm is less time consuming than Halton algorithm for the same order of accuracy.

Table 1

The absolute error of estimation for $k = 20$, $Z = 100$

Stage	n=1000		n = 3000	
	Proposed algorithm	Algorithm in [10]	Proposed algorithm	Algorithm in [10]
1	4.9526×10^{3}	4.995×10^{4}	2.9863×10^{4}	1.4247×10^{5}
3	3.8111×10^{1}	1.325×10^{5}	6.8797×10^{2}	2.2974×10^{5}
5	2.2295×10^{-1}	2.381×10^{5}	1.4266×10^{1}	3.2243×10^{5}
7	1.0542×10^{-3}	3.592×10^{5}	2.9225×10^{-1}	4.2148×10^{5}
9	4.8934×10^{-6}	5.047×10^{5}	6.0000×10^{-3}	5.2573×10^{5}
11	2.2530×10^{-8}	6.705×10^{5}	1.2324×10^{-4}	6.3526×10^{5}
13	1.0559×10^{-10}	8.502×10^{5}	2.5320×10^{-6}	7.4809×10^{5}
15	1.9701×10^{-11}	1.023×10^{6}	5.2027×10^{-8}	8.6135×10^{5}
17	1.9700×10^{-11}	1.233×10^{6}	1.0993×10^{-9}	9.7569×10^{5}
19	1.9700×10^{-11}	1.441×10^{6}	2.5570×10^{-10}	1.0900×10^{6}

Table 2

Time (in s) taken by Halton and proposed algorithm for the solution of large linear systems of different orders

Processors 'p'	Proposed algorithm			Halton algorithm		
	Order 'n'					
	1024	2048	4098	1024	2048	4096
1	0.6460	2.9123	11.6020	9.6429	49.9074	207.5782
2	0.4116	1.7583	6.6427	4.9696	25.2157	105.1958
3	0.3008	1.2814	4.7079	3.5176	17.8535	75.0621
4	0.3209	1.1519	4.2257	2.6768	13.0481	55.4293
5	0.2750	1.1654	3.8061	2.1599	10.6765	44.2075
6	0.2943	1.0043	3.6901	2.1051	10.4263	43.7587
7	0.2802	0.9277	3.5417	1.6795	8.1631	34.2888
8	0.2276	0.9101	3.4961	1.6817	8.2299	34.0587

Speed up against the different number of processors for large linear systems of different orders using the proposed algorithms is plotted in Fig. 1.

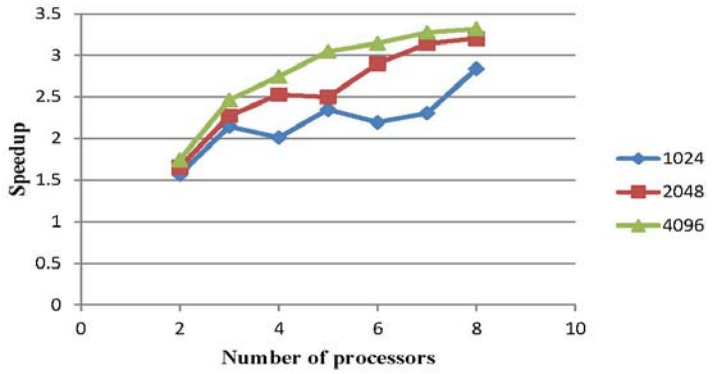

Fig. 1 – Speedup obtained against different number of processors for solving linear systems of various orders.

Example 2. Consider the linear system (1) where

$$B = \begin{bmatrix} 2\rho+1 & -\rho & & & \\ -\rho & 2\rho+1 & -\rho & & \\ & \cdots & & & \\ & & -\rho & 2\rho+1 & -\rho \\ & & & -\rho & 2\rho+1 \end{bmatrix},$$

$\rho = 0.5, F_i = \dfrac{i}{n}$ and $n = 1\,000,\ \ 2\,000$. The calculated error for matrices with sizes 1 000 and 2 000 is of order 10^{-16} for stages greater than 27. The results are shown in Table 3.

Table 3

The absolute error of proposed algorithm for $k = 10$, $Z = 100$

Stage	$n = 1000$	$n = 2000$
1	2.6790×10^{-1}	2.6702×10^{-1}
5	1.4019×10^{-3}	1.3702×10^{-3}
9	7.3432×10^{-6}	7.0362×10^{-6}
13	3.8493×10^{-8}	3.6132×10^{-8}
17	2.0191×10^{-10}	1.8552×10^{-10}
21	1.0593×10^{-12}	9.5290×10^{-13}
25	5.7731×10^{-15}	5.3290×10^{-15}
29	4.4408×10^{-16}	5.5511×10^{-16}

4.2. Option pricing

The well known Black Scholes model for a European put option can be described by formula:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0, \tag{9}$$

with final condition $V(S,T) = \max(E - S, 0)$ and boundary conditions $V(0,t) = E\mathrm{e}^{-r(T-t)}$ 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, the risk free interest rate, respectively. Also, S is assumed to behave $\mathrm{d}S = \mu S\mathrm{d}t + \sigma S\mathrm{d}W$, where $\mathrm{d}W$ is a Wiener process, μ and σ are the drift rate and the volatility of the asset, respectively. In this case, there exists a closed form solution. But, for more styles of options, closed form solutions do not exist. Stochastic methods can be used to price these options. In this regard, the adaptive Monte Carlo method can be used to value European options and pricing formulas for these options can be checked using this method.

The general discretization method can be used to approximate the solution of Eq.(9), where $\theta \in (0,1)$ is the parameter of discretization [16]. Assume $V_{ij} = V(i\Delta_S, j\Delta_t),\ \ 0 < i < N, 0 \le j \le M$. The Black Scholes model can be formulated as the following linear systems:

$$CV^{j+1} = DV^j + b^j, \tag{10}$$

where :

$$C = \begin{pmatrix} 1-\theta m_1 & -\theta u_1 & 0 & \cdots & 0 \\ -\theta d_2 & 1-\theta m_2 & -\theta u_2 & \cdots & 0 \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \\ 0 & \cdots & 0 & -\theta d_{N-1} & 1-\theta m_{N-1} \end{pmatrix}, b^j = \begin{pmatrix} \theta d_1 V_{0j} + (1-\theta) d_1 V_{0j+1} \\ 0 \\ \vdots \\ 0 \\ \theta u_{N-1} V_{Nj} + (1-\theta) u_{N-1} V_{Nj+1} \end{pmatrix},$$

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

and $d_i = \frac{\Delta_t \sigma^2 S_i^2}{2\Delta_S^2} - \frac{\Delta_t (r-q) S_i}{2\Delta_S}$, $m_i = -\frac{\Delta_t \sigma^2 S_i^2}{\Delta_S^2} - \Delta_t r$, $u_i = \frac{\Delta_t \sigma^2 S_i^2}{2\Delta_S^2} + \frac{\Delta_t (r-q) S_i}{2\Delta_S}$.

We solve the system of linear equations (10) using the proposed adaptive algorithm.

Example 3. Consider $E = 10$, $T = 0.5$, $\sigma = 0.2$, $r = 0.05$, $N = 300$, $M = 100$. We obtained the following numerical results. The values obtained by the proposed algorithm and those obtained by the Black Scholes formula and the difference between them as the error are shown in Table 4.

5. CONCLUSIONS

To solve large linear systems of algebraic equations, we improved the adaptive Monte Carlo algorithm. We analyzed the convergence, speed up and efficiency of the algorithm in the case of dealing with random matrices with sizes 1 000 and 3 000 and tridiagonal sparse matrices with sizes 1 000 and 2 000 which are arising from discretization of parabolic partial differential equation. From numerical results for random matrices, the proposed adaptive algorithm is less time consuming and therefore more efficient than the adaptive Monte Carlo algorithm which has been introduced before in [10]. The previous adaptive algorithm converges slow or diverge for the same number of random paths while the proposed algorithm converges exponentially and improves the convergence of both the conventional and adaptive Monte Carlo methods.

Also, Halton and the proposed algorithms are implemented to be parallelized using the MPI. This parallel computing allows us to decrease the time for solving systems. The results in Table 2 and Figure 1 show that the time taken by these algorithms is reduced using parallel computing and therefore they are good candidates for parallelization.

Furthermore, the proposed algorithm is implemented for option pricing using Crank-Nicolson ($\theta = 0.5$) to discritize the Black-Scholes formula. The results in Table 4 show the efficiency and accuracy of the proposed adaptive Monte Carlo algorithm for option pricing.

Table 4

Option value for $\theta = \frac{1}{2}$, $k = 15$, $Z = 20$, $r = 20$

Asset price	Proposed algorithm	Black-Scholes	Error
2	7.753099119	7.753099120	1.2699×10^{-9}
3	6.753099119	6.753099120	1.2370×10^{-9}
4	5.753099369	5.753099120	2.4941×10^{-7}
5	4.753102615	4.753099342	3.2730×10^{-6}
6	3.753302476	3.753180620	1.2185×10^{-4}
7	2.757692583	2.756835269	8.5731×10^{-4}
8	1.798755250	1.798714599	4.0650×10^{-5}

ACKNOWLEDGEMENTS

The authors would like to acknowledge the support of parallel computing research laboratory of IUST School of mathematics for the research.

REFERENCES

1. ALEXANDROV, V. N., DIMOV, I. T., KARAIVANOVA, A., TAN, C.J.K., *Parallel Monte Carlo algorithms for information retrieval*, Mathematics and Computers in Simulation, **62**, pp. 289–295, 2003.
2. ALEXANDROV, V. N., MARTEL, C. G., STRAßBURG, J., *Monte Carlo scalable algorithms for computational finance*, Procedia Computer Science, **4**, pp. 1708–1715, 2011.
3. BAYKUS, N., SEZER, M., *Solution of high-order linear Fredholm integro-differential equations with piecewise intervals*, Numerical Methods for Partial Differential Equations, **27**, *5*, pp. 1327–1339, 2010.
4. CONT, R., TANKOV, P., *Financial modelling with jump processes*, Chapman and Hall/CRC Press, 2003.
5. DEHGHAN, M., HAJARIAN, M., *SSHI methods for solving general linear matrix equations*, Engineering Computations, **28**, *8*, pp. 1028–1043, 2011.
6. FARNOOSH, R., AALAEI, M., EBRAHIMI, M., *Combined probabilistic algorithm for solving high dimensional problems*, Stochastics: An International Journal of Probability and Stochastic Processes, 2014.
7. HALTON, J., *Sequential Monte Carlo*, Proceedings of the Cambridge Philosophical Society, **58**, pp. 57–78, 1962.
8. HAN, C. H., LAI, Y., *A smooth estimator for MC/QMC methods in finance*, Mathematics and Computers in simulation, **81**, *3*, pp. 536–550, 2010.
9. JASRA, A., MORAL, P.D., *Sequential Monte Carlo methods for option pricing*, Stochastic analysis and applications, **29**, pp. 292–316, 2011.
10. LAI, Y., *Adaptive Monte Carlo methods for matrix equations with applications*, Journal of Computational and Applied Mathematics, **231**, pp. 705–714, 2009.
11. LAI, Y., SPANIER, J., *Applications of Monte Carlo/Quasi-Monte Carlo methods in finance: option pricing*, in: *Monte-Carlo and Quasi-Monte Carlo Methods 1998*, Proceedings of a Conference at the Claremont Graduate University, Claremont, California, USA, June 22–26, 1998.
12. LI, L., SPANIER, J., *Approximation of transport equations by matrix equations and sequential sampling*, Monte Carlo Methods and Applications, **3**, pp.171–198, 1997.
13. MUTHUVALU, M. S., SULAIMAN, J., *The arithmetic mean iterative methods for solving dense linear systems arise from first kind linear Fredholm integral equations*, Proceedings of the Romanian Academy, Series A, **13**, *3*, pp. 198–206, 2012.
14. RUBINSTEIN, R. Y., *Simulation and the Monte Carlo method*, Wiley, New York, 1981.
15. TAN, C.J.K., *Solving systems of linear equations with relaxed Monte Carlo method*, The Journal of Supercomputing, **22**, pp. 113–123, 2002.
16. WILMOTT, P., DEWYNNE, J., HOWISON, S., *Option pricing: mathematical models and computation*, Oxford University Press, 1995.

Received December 11, 2012