VECTORIZED MONTE CARLO SOLUTIONS OF LINEAR EQUATIONS

by

Riyanarto Sarno, Virendra C. Bhavsar
Esam M.A. Hussein

TR92-069 July 1992

Faculty of Computer Science
University of New Brunswick
P.O. Box 4400
Fredericton, N.B. E3B 5A3

Phone: (506) 453-4566
Fax: (506) 453-3566
Email: bhavsar@unb.ca

Abstract

Monte Carlo (MC) methods for solving a system of linear equations exhibit high parallelism. The vectorization and incorporation of various sampling methods into the MC methods are investigated to speed up the computations. The weighted sampling method and an alias-like method are found to reduce the order of the time complexity of the MC methods from n^3, when using the inverse method, to n^2 for estimating n unknowns. This results into faster solutions with scalar as well as vector processing as demonstrated with computational studies on an IBM 3090-180 computer with a vector facility. The MC methods are shown to be attractive when the number of unknowns is very large and the estimation of only a very small number of unknowns is required.

Keywords: linear equations, Markov chains, Monte Carlo methods, sampling, supercomputing, vector processing.

AMS (MOS) subject classifications: 60J10, 65C05, 65C10, 65F10, 65Y20.

Carlo methods even more attractive.

In this paper, we investigate the effect of different sampling methods incorporated into the Monte Carlo methods for solving linear equations. We propose the use of vector processing to reduce the computing time and examine the vectorization of the Monte Carlo codes. For the computational studies we consider a Laplace's equation and develop scalar and the corresponding vectorized codes. The implementation of these codes is carried out on an IBM-3090 with a vector facility, and the numerical and processing time performance is analyzed.

2. Monte Carlo Solutions. Let us begin by considering a nonsingular system of equations defined by

$$\sum_{s=1}^{n} \mathbf{A}_{rs}\mathbf{x}_s = \mathbf{b}_r, \text{ where } r \in \{1, 2, \cdots, n\}, \tag{1}$$

which can also be written in the matrix equation $\mathbf{Ax}=\mathbf{b}$. This equation has a unique solution given by $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$. By introducing $\mathbf{H} = \mathbf{I}-\mathbf{A}$, where $\mathbf{I}$ is an identity matrix, Equation (1) can be rewritten as

$$\mathbf{x} = \mathbf{Hx} + \mathbf{b}. \tag{2}$$

When the spectral radius of $\mathbf{H}$ is less than one ($\rho(\mathbf{H}) < 1$), the Neumann series expansion

$$\begin{aligned} \mathbf{x} &= (\mathbf{I}-\mathbf{H})^{-1}\mathbf{b} \\ &= (\mathbf{I}+\mathbf{H}+\mathbf{H}^2+\cdots+\mathbf{H}^m+\cdots)\mathbf{b} \\ &= \sum_{m=0}^{\infty} \mathbf{H}^m\mathbf{b} \end{aligned} \tag{3}$$

is absolutely convergent and the Monte Carlo solutions can be obtained based on this equation (Halton, 1970).

2.1. Solution Using An Absorbing Markov Chain . The Monte Carlo method, suggested in (Forsythe and Leibler, 1950), employs an absorbing Markov chain to solve a system of linear equations. In a Markov chain, the random walks are governed by a probability matrix. The indices of the matrix are called states. An absorbing Markov chain has an absorbing state, at which a random walk is terminated. The procedure for obtaining the estimate of an unknown is as follows. A set of random walks is defined on the augmented index set, $S = \{0, 1, ..., n\}$, where index 0 represents the absorbing

state. Let $\mathbf{P}$ denote a $(n+1)\times(n+1)$ transition probability matrix, and $\mathbf{P}_{ij}$ denote the probability of transition from state i to state j, with $\mathbf{P}_{ij} \neq 0$ unless $\mathbf{H}_{ij} = 0$. Then, corresponding to each random walk $\Gamma = \{r, s_1, \cdots, s_m, 0\}$, with $r, s_1, \cdots, s_m \neq 0$, the primary estimator for an unknown $\mathbf{x}_r$ is given as

$$X_r(\Gamma) = \mathbf{Z}_{r,s_1}\mathbf{Z}_{s_1,s_2}\cdots\mathbf{Z}_{s_{m-1},s_m}\mathbf{Z}_{s_m,0} \tag{4}$$

where for $j \neq 0$, $\mathbf{Z}_{ij} = \mathbf{H}_{ij}/\mathbf{P}_{ij}$ and for $j = 0$, $\mathbf{Z}_{i,0} = \mathbf{b}_i/\mathbf{P}_{i0}$.

The secondary estimator for $\mathbf{x}_r$ is then expressed by the arithmetic mean of k primary estimates as follows

$$Y_r = \frac{1}{k}\sum_{i=1}^{k} X_r(\Gamma_i). \tag{5}$$

The $Y_r \rightarrow \mathbf{x}_r$ as $k \rightarrow \infty$.

Note that the Monte Carlo estimate of each of the unknowns in Equation (3) can be obtained independently of the remaining unknowns. Therefore, the computations of each of the unknowns can be carried out in parallel. Further, the computations of the k primary estimates are also independent of each other and consequently they can be performed in parallel.

2.2. Solution Using An Ergodic Markov Chain . An application of an ergodic Markov chain for solving a system of linear equations was proposed in (Wasow, 1952). The random walks in an ergodic Markov chain are not terminated by an absorbing state, but they are predetermined.

In this method, scoring for the primary estimator of an unknown is performed at every random walk step. For each random walk from an ergodic Markov chain $\Lambda = \{r, s_1, \cdots, s_{m-1}, s_m\}$, the primary estimator of an unknown $\mathbf{x}_r$ is given by

$$\begin{aligned} X_r(\Lambda) \;=\; & \mathbf{b}_r + \mathbf{b}_{s_1}\mathbf{Z}_{r,s_1} + \mathbf{b}_{s_2}\mathbf{Z}_{r,s_1}\mathbf{Z}_{s_1,s_2} + \cdots \\ & + \mathbf{b}_{s_m}\mathbf{Z}_{r,s_1}\mathbf{Z}_{s_1,s_2}\cdots\mathbf{Z}_{s_{m-1},s_m} \end{aligned} \tag{6}$$

where $\mathbf{Z}_{ij} = \mathbf{H}_{ij}/\mathbf{P}_{ij}$. Wasow has proved that the variance of this estimator is smaller than that of the estimator given in Equations (4) when the absorption probabilities in the transition probability matrix of an absorbing Markov chain ($\mathbf{P}_{i0}$ for $i = 1, 2, \cdots, n$) are relatively small. Further comparison of the variances is discussed in (Edmundson, 1952).

3. Sampling For State Transitions. The state transitions during random walks are determined by sampling from the transition probability matrix. The inverse method (Law and Kelton, 1982) is usually used for this sampling process. The scalar code of this method is given in Fig. 1. On calling this subroutine, the argument `i` contains a current state. On return, it contains the next state. Array `C` denotes a matrix for the cumulative distribution of the transition probability matrix. The RNUNF function is an IMSL (IMSL, 1989) routine, which generates uniform random numbers in the interval $(0, 1)$. This subroutine involves sequential searching to locate the interval in which a generated random number lies. Therefore, the performance of the inverse method depends on the size of the matrix.

Brown (Brown et al., 1981) proposed a sampling method, which is similar to the alias method (Kronmal and Peterson, 1979). In both methods, the original distribution with n mass points is transformed into an equiprobable mixture of n distributions. Each distribution has two mass points. The scalar code of Brown's method is shown in Fig. 2. Two random numbers are required to generate a sample. The first random number is used to select the distribution, and the second is used to choose the mass point. Array `id` stores the mass points of the n distributions, and array `f` stores the probabilities of the first mass points of the n distributions. Brown's method contains neither `do`-loops nor `if`-statements. However, n two-point distributions are required to represent each row of the transition probability matrix. Therefore, it is not attractive for large matrices.

We incorporate the weighted sampling method (Sarno et al., 1990) to speed up the determination of states. This method is an importance sampling method (Rubinstein, 1981) which samples from an assigned discrete distribution, different from that of the original problem. The new distribution is chosen here not only to reduce the sample variance, but also to speed up the computation. In this paper, a uniform distribution is utilized for the weighted sampling method, since it is vectorizable and requires smaller execution time both in scalar and vector processing.

Since the sampling for determining states is not carried out based on the original transition probability matrix, the estimator is multiplied by an adjustment factor in order to obtain an unbiased solution. The procedure for weighted sampling using a uniform distribution is explained as follows. Assume that s_1 is a current state, and it is located in the i-th row of the transition probability matrix. The probabilities are

$\{\mathbf{P}_{ij}, j = 1, 2, \cdots, \mathbf{l}_i\}$, where $\mathbf{l}_i$ is the number of nonzero probabilities in the i-th row. Then, we use the weighted sampling method to select the next state, s_2, according to a discrete uniform distribution in the interval $[1, \mathbf{l}_i]$. Subsequently, the primary estimator has to be multiplied by the adjustment factor $\mathbf{P}_{ij} \times \mathbf{l}_i$, where j is the selected state number.

In the Monte Carlo method employing an absorbing Markov chain, when the weighted sampling is used to carry out each random walk $\tilde{\Gamma} = \{r, s_1, \cdots, s_m, 0\}$, with $r, s_1, \cdots, s_m \neq 0$, the primary estimator for an unknown $\mathbf{x}_r$ is given as

$$\tilde{X}_r(\tilde{\Gamma}) = \tilde{\mathbf{Z}}_{r,s_1} \tilde{\mathbf{Z}}_{s_1,s_2} \cdots \tilde{\mathbf{Z}}_{s_{m-1},s_m} \tilde{\mathbf{Z}}_{s_m,0}, \tag{7}$$

for $j \neq 0, \tilde{\mathbf{Z}}_{ij} = \mathbf{Z}_{ij}[\mathbf{P}_{ij} \times \mathbf{l}_i]$ and for $j = 0, \tilde{\mathbf{Z}}_{i,0} = \mathbf{Z}_{i0}[\mathbf{P}_{i0} \times \mathbf{l}_i]$, where the quantities in the square brackets represent the adjustment factors.

According to Equation (4), $\mathbf{Z}_{ij} = \mathbf{H}_{ij}/\mathbf{P}_{ij}$ for $j \neq 0$, and $\mathbf{Z}_{i,0} = \mathbf{b}_i/\mathbf{P}_{i0}$ for $j = 0$. Therefore, $\tilde{\mathbf{Z}}_{ij} = \mathbf{H}_{ij}\mathbf{l}_i$ for $j \neq 0$, and $\tilde{\mathbf{Z}}_{i,0} = \mathbf{b}_i\mathbf{l}_i$ for $j = 0$. This means that the random walks are governed by a uniform transition probability matrix, and the probability for each row is $1/\mathbf{l}_i$ for $i = 1, 2, \cdots, n$. Similarly, this applies for the Monte Carlo method employing an ergodic Markov chain.

Thus, when the weighted sampling method is incorporated into a Monte Carlo method for solving linear equations, we basically replace the original transition probability with the probability distribution utilized by the weighted sampling method (Sarno, 1992). That is, the state transitions are sampled uniformly since the weighted sampling method utilizes a uniform distribution.

4. Vectorization . As stated earlier vector processing can be used to reduce the computing time of the Monte Carlo codes. The scalar codes when modified for enhancing vectorization on vector computers are often called as vectorized Monte Carlo codes (Martin et al., 1986). In this section, we discuss the vectorization of the Monte Carlo codes based on absorbing and ergodic Markov chains.

4.1. Absorbing Markov Chain Code. In the scalar Monte Carlo algorithm, random walks are performed one at a time. The chain is tracked from birth to absorption. In a vectorized Monte Carlo algorithm, however, a set of random walks are carried out concurrently. For this purpose, a stack (a set of vectors) is utilized to hold the attribute values of random walks (Martin et al., 1986). Since the random walk length is a random variable, the stack may contain absorbed and nonabsorbed walks. When

the stack contains relatively many absorbed walks, the utilization of vector processing hardware decreases. Therefore, it is required to remove absorbed walks and gather the nonabsorbed walks. The gather process should be done in such a way that the trade off between the vector processing efficiency and the overhead time is optimal.

A vectorized Monte Carlo algorithm using an absorbing Markov chain is given in Fig. 3. Line 1 is the loop for estimating n unknowns. Initially, $\nu = k$. Line 6 shows the loop for random walks, and limits their length to at most m. For each random walk, lines 8–13 show that ν number of samples are carried out concurrently in vector processing. Line 9 shows that the weighted sampling method is used for determining state transitions. The `if`-statements are used in lines 10–11 to select particular walks for the primary estimate calculation. The secondary estimate calculation of the absorbed walks is done in lines 14–16. Then, the gather process is carried out in lines 18–23. Line 24 shows that the value of ν is updated; i.e. the stack contains only nonabsorbed walks. This algorithm carries out the gather process in every transition of random walk since it results in minimum processing time. Finally, in line 26 the average value of the secondary estimate is calculated according to the number of absorbed walks $(k - \nu)$; i.e. all random walks of k samples do not necessarily reach the absorbing state due to the truncation of random walk length.

When the inverse method is used in the Monte Carlo code, the innermost loop will be the loop for implementing the inverse method; this loop replaces the weighted sampling method given in line 9. This innermost loop is vectorizable. However, the larger loop (lines 8–13) is not vectorizable since the innermost loop contains a `go to`-statement.

Brown's method requires four lines of code (see Fig. 2) to replace line 9. These lines contain neither loops nor vectorization inhibitors. Therefore, using the weighted sampling method or Brown's method the vectorization will be achieved for the larger segment/loop of the program (lines 8–13). Since k is usually much larger than n, the vectorization of the larger loop results in less processing time than that of the innermost loop. Thus, the weighted sampling method enhances the vectorizability of the code, and reduces the processing time.

4.2. Ergodic Markov Chain Code. Similar to the vectorized algorithm of the solution using an absorbing Markov chain, the vectorized algorithm here also utilizes the stack processing. The vectorized Monte Carlo algorithm using an ergodic Markov

chain and incorporating the weighted sampling method is shown in Fig. 4. In this algorithm, the random walk length is predetermined; i.e. all random walks of k samples have the same length. This fact eliminates the use of `if`-statements in the primary and secondary estimate calculation, see lines 8–13 and and lines 14–16, respectively. Since the termination of random walks is the same for all samples, the gather process is not required. Note that the average value of the secondary estimate is calculated according to k samples (see line 18), since all random walks reach termination. Thus, the vectorization of this algorithm is more efficient than that of the Monte Carlo algorithm using an absorbing Markov chain.

5. Time Complexity Analysis. This section analyzes the time complexities of the Monte Carlo algorithms for solving a system of linear equations. The complexities are expressed in terms of the number of comparisons and the number of state transitions for obtaining a primary estimator. The analysis is done in two steps. First, we obtain the number of comparisons required to determine a next state for each state transition. Second, we estimate the random walk length; i.e. the number of state transitions in a random walk required for obtaining a primary estimator. Then, the complexity of the Monte Carlo algorithm is calculated by multiplying the two results.

5.1. State Determination.

5.1.1. The Inverse Method. When the inverse method is used to determine the state transitions (see Fig. 1), the algorithm complexity is as follows.

Let V_{s_1} denote a random variable representing the number of comparisons required to determine the next state s_2 from a current state s_1. We assume that the number of nonzero probabilities for each row of matrix $\mathbf{P}$ is n; i.e. $\mathrm{l}_i = n$, for $i = 1, 2, \cdots, n$. With this assumption, the results are overestimated since $\mathrm{l}_i \leq n$ is the case for a sparse system of linear equations. The expected number of comparisons and its variance can be calculated based on the following probability theory. The expected number of comparisons ($E[V_{s_1}]$) is given by

$$E[V_{s_1}] = \sum_{i=1}^{n+1} i\mathbf{P}_{s_1,(i-1)}, \tag{8}$$

while the variance is given by

$$Var[V_{s_1}] = \sum_{i=1}^{n+1} \mathbf{P}_{s_1,(i-1)} \left(i - E[V_{s_1}]\right)^2. \tag{9}$$

Equation (9) can be simplified as

$$Var[V_{s_1}] = \sum_{i=1}^{n+1} i^2 \mathbf{P}_{s_1,(i-1)} - E[V_{s_1}]^2. \tag{10}$$

Thus, the inverse method requires $\Theta(n)$ number of comparisons to determine a next state. For example, consider that $P_{s_1,j}, j = 0, 1, \cdots, n$, are uniformly distributed; i.e. $P_{s_1,j} = \frac{1}{n+1}$ for $j = 0, 1, \cdots, n$. Then the expected number of comparisons is $E[V_{s_1}] = \frac{n+2}{2}$, and the associated variance is $Var[V_{s_1}] = \frac{n(n+12)}{12}$, see (Sarno, 1992) for the details. Thus, it is seen that the expected number of comparisons and the associated variance depend on the value of n.

5.1.2. The weighted sampling method. When the weighted sampling method is used for the state determination, it does not require any comparisons and therefore it requires only a constant time. Thus, the time complexity of the sampling process is *independent* of the matrix size.

As the case of the weighted sampling method, Brown's method also does not require any comparisons. This method therefore does not depend on the matrix size.

5.2. The Monte Carlo Algorithms. This subsection discusses the time complexities of the Monte Carlo algorithms for solving linear equations by employing ergodic and absorbing Markov chains.

5.2.1. Algorithm Using An Ergodic Markov Chain. In the algorithm using an ergodic Markov chain, the random walk length for each sample is predetermined and is the same, i.e. the value of s_m in Equation (6) is given to be the same for all samples. For solving n number of linear equations, if we carry out k number of samples for each of the unknowns, then the total number of state transitions is equal to ks_mn. If we determine $s_m = n$, the total number of state transitions equals kn^2.

When the inverse method is used for determining state transitions, each state transition requires $\Theta(n)$ number of comparisons. Thus, the time complexity of the Monte Carlo algorithm is $\Theta(n^3)$.

However, when the weighted sampling method is used to determine state transitions, no comparisons are required. Consequently, the time complexity of the Monte Carlo algorithm reduces to $\Theta(n^2)$.

5.2.2. Algorithm Using An Absorbing Markov Chain. In this case, the random walk length (number of state transitions) for each sample is a random variable.

The expectation of this random variable is found out as follows.

The substochastic matrix $\mathbf{Q}$, corresponding to the transition probability matrix $\mathbf{P}$, is given as (Kemeny and Snell, 1960)

$$\mathbf{Q} = \mathbf{P}_{ij}, \quad i, j \neq 0, \tag{11}$$

and the fundamental matrix $\mathbf{F}$ is given by

$$\mathbf{F} = (\mathbf{I} - \mathbf{Q})^{-1}, \tag{12}$$

where $\mathbf{I}$ is the identity matrix.

For each sample utilized in estimating an unknown x_i, the length of random walks from state i to the absorbing state 0 is a discrete random variable with the mean given by (Kemeny and Snell, 1960)

$$\mu_i = \sum_{j=1}^{n} \mathbf{F}_{ij}. \tag{13}$$

Note that an element $\mathbf{F}_{ij}$ of the fundamental matrix $\mathbf{F}$ describes the total number of walks visiting state j, if the chain started from state i, until it reaches an absorbing state.

In order to estimate n unknowns, the expectation of the total number of state transitions is expressed by

$$E[W] = k \sum_{i=1}^{n} \mu_i, \tag{14}$$

where k is the number of samples for each unknown. This equation implies that the time complexity for estimating n unknowns is $\Theta(n^2)$. For a transition probability matrix of size $(n+1) \times (n+1)$ wherein the nonzero probabilities in a row are equal, the expected number of state transitions for estimating n unknowns will be $E[W] = kn(n+1)$.

When the inverse method is used for determining state transitions, each state transition requires $\Theta(n)$ number of comparisons. Thus, the time complexity of the Monte Carlo algorithm is $\Theta(n^3)$.

In contrast with the inverse method, the weighted sampling method does not require any comparisons for determining state transitions. Thus, the time complexity of the Monte Carlo algorithm is only $\Theta(n^2)$.

Since Brown's method does not need any comparisons and requires a constant time for determining state transitions, the time complexity of the Monte Carlo algorithms

incorporating Brown's method is the same as that of Monte Carlo algorithms implementing the weighted sampling method. Note that Gauss elimination method (a direct inversion method) has a time complexity of $\Theta(n^3)$ (Kronsjo, 1987), while that of the Gauss-Seidel iteration method is $\Theta(n^2)$ (Varga, 1962).

6. Results and Discussion. For the computational study, we examine a Laplace's equation which is often used in mathematical physics and engineering (Cheney and Kincaid, 1985). The objective is to investigate the performance of the Monte Carlo solutions using different sampling methods on scalar and vector processing. We chose the inverse method as a reference for comparison, since this method is commonly used. The inverse method is expected to perform well due to only a few numbers of mass points involved in this problem. Brown's method is fully vectorizable and is therefore examined here as well. Several aspects are investigated, including the solution and its error, the processing time, the vectorization speedup, and the efficiency.

Using the cartesian coordinate system, the problem can be defined as

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0. \tag{15}$$

The boundary conditions for the region $0 \leq x \leq 10$, and $0 \leq y \leq 10$ are chosen arbitrarily as $u(x,0) = u(x,10) = u(10,y) = -10$ and $u(0,y) = 10$.

The finite difference method using the five point formula (Cheney and Kincaid, 1985, p. 444) is used to approximate the original problem. The x as well as y axis is discretized into 33 intervals. This discretization results in 1024 internal points. Each point represents an unknown in the linear equation. Thus, the linear system contains 1024 linear equations. The linear equations can then be solved using Gauss-Seidel iterative method (Cheney and Kincaid, 1985, p. 450) or the Monte Carlo methods.

When the linear system is represented as matrix equation **Ax=b**, the resulting matrix **A** is a sparse matrix with a regular nonzero structure (a band matrix). The iterative method is known to perform well for linear equations involving band matrices (Dongarra et al., 1991). Therefore, the Monte Carlo estimates are compared with the results obtained using the iterative method. The numerical solutions are also compared with the analytical solution of the Laplace's equation obtained using some results from reference (Berg and McGregor, 1966, p. 312).

In the ensuing subsections, we use the following names for both scalar and vector codes of Monte Carlo algorithms:

1. SEIDEL: code for the Gauss-Seidel iterative method,
2. ERGIN: code for the ergodic algorithm using the inverse sampling method,
3. ERGDS: code for the ergodic algorithm using Brown's sampling method,
4. ERGWS: code for the ergodic algorithm using the weighted sampling method,
5. ABSIN: code for the absorbing algorithm using the inverse sampling method,
6. ABSDS: code for the absorbing algorithm using Brown's method, and
7. ABSWS: code for the absorbing algorithm using the weighted sampling method.

6.1. Results. This subsection reports the solutions of the Laplace's equation for all unknowns and a particular unknown. We use the solution of the iterative method as a reference. Figure 5 shows the solution obtained using the iterative method (SEIDEL). The number of iterations for the iterative method, which results in a convergent solution, is found to be equal to 80.

For the Monte Carlo solutions employing ergodic Markov chains, the transition probabilities are distributed uniformly. Therefore, the solutions obtained using the inverse method, Brown's method and the weighted sampling method are the same. Figure 6 depicts the solutions obtained by ERGWS code with the random walk length equals 125. Each unknown employs 1000 samples.

In the Monte Carlo solutions employing absorbing Markov chains, the Monte Carlo solutions obtained using ABSIN and ABSIN are very close since their transition probability matrices are the same. In these probability matrices, the transition probabilities are uniformly distributed on all points except for those of the boundary points. Since the number of boundary points are much less than that of the internal points, the uniform distribution is more dominant than the nonuniform distribution. Consequently, the solutions of ABSIN and ABSDS are close to that of ABSWS which utilizes uniform distributions for all points. Figure 7 shows the solution obtained by ABSWS.

The error for all unknowns of the Monte Carlo solution is calculated relative to the solution of the iterative method. We define this error as

$$\text{Error } = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}, \tag{16}$$

where $\hat{y}_i$ is a solution for an unknown obtained by the iterative method, while y_i is the solution for the same unknown obtained using the Monte Carlo code. The number of unknowns is denoted by n.

The average values of the solution for all unknowns and the corresponding errors

are presented in Table 1. As shown in this table, the Monte Carlo codes employing an absorbing Markov chain result in smaller errors. For each sample, the random walk length in the ergodic Markov chain is chosen to be equal to 125, which also represents the maximum random walk length in the absorbing Markov chain.

As shown in Table 1, the ergodic algorithms result in larger errors than the absorbing ones. This is caused by the fixed lengths of the random walks for estimating the solutions of all unknowns. When we choose a random walk of length 50, good solutions of the boundary points are obtained, while poor solutions calculated for the central points. However, if the random walk of length is 125, the solutions of the central points improve, but the solutions of the boundary points become inacceptable. In conclusion, the absorbing codes are more suitable for estimating all unknowns.

In order to investigate the convergence of the Monte Carlo solutions we examine the solution at point (2.4242, 2.4242), which is located in the middle of the lower left corner grid point and the centre grid point. This point is chosen, since it is between two extreme points. Figure 8 demonstrates the solutions at this point We chose 80 as the random walk lengths for the ergodic codes, and 125 as the maximum random walk lengths for the absorbing codes. This figure shows that all Monte Carlo solutions converge to the SEIDEL's solutions. Note that the Monte Carlo solutions obtained using the ergodic codes (ERGIN, ERGDS and ERGWS) are the same, since they have the same transition probabilities which are distributed uniformly.

6.2. Processing Time and Speedup. In this subsection the scalar and vector processing time of the solutions for all unknowns are first discussed. Then, the increase in processing time as a function of the number of unknowns is demonstrated.

The scalar and vector processing time of Monte Carlo solutions for all unknowns are depicted in Table 2. Each unknown utilizes 2000 samples. In scalar processing, the ergodic and absorbing codes incorporating the weighted sampling method (ERGWS and ABSWS) require about 20 % less computing time than those of the codes implementing the inverse method (ERGIN and ABSIN). The ergodic and absorbing codes implementing Brown's method (ERGDS and ABSDS) require larger processing time than those incorporating the inverse method. The performance of the inverse method is comparatively good since the maximum number of comparisons is only four. In the vector processing, Table 2 shows that ERGIN requires 3.5 times more computing time than that of ERGWS. This means that the weighted sampling can enhance the vectorization

of the ergodic algorithm; however, it is less successful in the absorbing algorithm. The maximum speedup is achieved by using the vector ERGWS over the scalar one. The ERGWS can achieve the best speedup, because the uniform sampling contains no loop and the random walk length is predetermined.

We now compare the processing time for all unknowns required by the iterative method (SEIDEL) and the Monte Carlo method (which requires the least processing time). For this purpose we choose a sample size of 2000, in which the solutions obtained by SEIDEL and ERGWS are relatively equal. Table 2 demonstrates that ERGWS requires the smallest processing time in the scalar and vector processors. These times are 485.23 and 86.03 seconds respectively. The scalar and vector processing of SEIDEL require 616 milliseconds and 456 milliseconds, respectively, to complete.

Therefore, the scalar ERGWS requires about 788 times longer processing time than SEIDEL; whereas, the vector ERGWS requires about 188 times longer processing time than the vector SEIDEL. Assuming that the errors are relatively equal, the scalar ERGWS is beneficial if it computes less than 0.13 % of the total unknowns. Moreover, the vector ERGWS still gives benefit when it estimates less than 0.53 % of the total unknowns. Thus, for the estimation of a single unknown, the speedup of the vector ERGWS over the scalar SEIDEL is 7.1.

Table 3 demonstrates that the scalar and vector processing time of the iterative method increase by a factor of about 8. This is caused by the increase in problem size by a factor of 4 and the increase in maximum iteration by a factor of 2.

The time complexity of the Monte Carlo solution for solving n unknowns is $\Theta(n^2)$. If the problem size increases by a factor of 4, the processing time should increase 16 times. However, Table 3 shows that the scalar and vector processing time of Monte Carlo solutions increase by a factor of about 8. This is caused by the increase in problem size by a factor of 4 and the increase in random walk length by a factor of 2. We only increased the random walk length twice, since the number of nonzero elements for each row remains the same with the increase in problem size. The number of samples for each unknown utilized by the Monte Carlo codes is 2000.

Table 4 shows the vectorization speedups for different number of unknowns. The speedups for the iterative method and the ergodic codes are relatively constant. The absorbing codes attain the highest speedups when the number of unknowns is equal to 256. It is due to the optimality of vector processing for this size.

6.3. Efficiency. The efficiency is defined as a quantity inversely proportional to the product of the sample variance and the processing time (Rubinstein, 1981). The relative efficiency of a Monte Carlo method with respect to another Monte Carlo method is given by the ratio of their efficiencies. Thus, the relative efficiency compares both computational and statistical performance of Monte Carlo methods.

The relative efficiencies of solutions at point (2.4242, 2.4242) obtained by various vector codes with respect to those of the scalar ABSIN are presented in Table 5. The efficiency obtained by ABSIN was chosen as a reference, since it is often used in the literature. It is seen that the weighted sampling method results in the highest relative efficiencies for both ergodic and absorbing codes. The relative efficiencies are higher than their vectorization speedups. This indicates that the weighted sampling method entails lower variances than those of the ABSIN.

7. Conclusions. The vectorized Monte Carlo solutions of linear equations employing absorbing and ergodic Markov chains have been investigated. Brown's and the weighted sampling methods enhance the vectorizability of the Monte Carlo codes. Further, these sampling methods also reduce the time complexity of the Monte Carlo algorithms from $\Theta(n^3)$ to $\Theta(n^2)$, where n is the number of the unknowns. This results into faster solutions with both scalar and vector processing as compared to the solutions using the inverse sampling method.

Based on the computational studies, the Monte Carlo methods become attractive when the estimation of at most 0.1 % and 0.5 % of the unknowns is required in scalar and vector processing, respectively.

We expect that the combined use of vector and parallel processing available on recent high performance computers will make the Monte Carlo methods for solving linear equations computationally more attractive in the near future.

Acknowledgements. The first author would like to thank the Government of Indonesia for the financial support for studies at the University of New Brunswick. This research was partially supported by the Natural Sciences and Engineering Research Council of Canada grant no. OGP0089.

REFERENCES

[1] Berg P.W. and McGregor J.L. 1966. *Elementary Partial Differential Equations*. San Francisco: Holden–Day Inc.

[2] Bhavsar, V.C. 1981. *Parallel Algorithms For Monte Carlo Solutions of Some Linear Operator Problems*. Ph.D. Thesis. Bombay: Indian Institute of Technology, Department of Electrical Engineering.

[3] Bhavsar, V.C. and Isaac, J.R. 1987. Design And Analysis of Parallel Monte Carlo Algorithms. *SIAM J. Sci. Stat. Comput.* 8(1):73–95.

[4] Brown, F.B., Martin, W.R and Calahan, D.A. 1981. A Discrete Sampling Method for Vectorized Monte Carlo Calculations. *Trans. Am. Nucl. Soc.* 38:354–355.

[5] Cheney, W. and Kincaid, D. 1985. *Numerical Mathematics and Computing*. 2nd edition. California: Brooks / Cole Publishing Co.

[6] Edmundson, H.P. 1952. Monte Carlo Matrix Inversion and Recurrent Events. *Math. Tables Other Aids Comput.* 7:18–21.

[7] Forsythe G.E. and Leibler, R.A. 1950. Matrix Inversion by a Monte Carlo Method. *Math. Tables Other Aids Comput.* 4:127–129.

[8] Halton, J.H. 1970. A Retrospective and Prospective Survey of The Monte Carlo Methods. *SIAM Review.* 12(1):1–63.

[9] IBM. 1989. *VS FORTRAN Version 2 Release 4, Language and Library Reference.* 5th edition. New York: IBM Corp.

[10] IMSL. 1989 *Fortran Subroutines for Statistical Analysis.* Version 1.0. Texas: IMSL Inc.

[11] Dongarra, J.J., Duff, L.S., Sorensen, D.C. and Van Der Vorst, H.A. 1991. *Solving Linear Systems on Vector and Shared Memory Computers.* Philadelphia: SIAM.

[12] Kemeny, J.G. and Snell, J.L. 1960. *Finite Markov Chains.* New Jersey: D. Van Nostrand Company.

[13] Kronmal, R.A. and Peterson Jr., A.V. 1979. On the Alias Method for Generating Random Variables from a Discrete Distribution. *The American Statistician.* 33(4):214–218.

[14] Kronsjo, L. 1987. *Algorithms: Their Complexity and Efficiency.* 2nd edition. Chichester: John Wiley & Sons Inc.

[15] Law, A.M. and Kelton, W.D. 1982. *Simulation Modeling and Analysis.* New York: McGraw-Hill Book Company.

[16] Martin, W.R., Nowak, P.F. and Rathkopf, J.A. 1986. Monte Carlo Photon Transport On A Vector Supercomputer. *IBM Journal of Research and Development.* 30(2):193–201.

[17] Rubinstein, R.Y. 1981. *Simulation and The Monte Carlo Method.* New York: John Wiley & Sons,Inc.

[18] Sarno, R., Bhavsar, V.C. and Banerjee, P.K. 1989. Design and Analysis of Vector Processing in Monte Carlo Particle Transport Codes. *Proceedings of the 1989 International Conference on Parallel Processing.* The Pennsylvania State University Press. III.80–III.87.

[19] Sarno, R., Bhavsar, V.C. and Hussein, E.M.A. 1990. Generation of Discrete Random Variables on Vector Computers For Monte Carlo Simulations. *International Journal of High Speed Computing.* 2(4):335–350.

[20] Sarno, R., Bhavsar, V.C. and Hussein, E.M.A. 1991. Monte Carlo Solutions of Linear Equations Using Weighted Sampling. *Proceedings of Supercomputing Symposium '91.* edited by V.C. Bhavsar and U.G. Gujar. Canada: University of New Brunswick, Faculty of Computer Science, pp. 151–163.

[21] Sarno, R. 1992. *Discrete Sampling Methods For Vector Monte Carlo Codes.* PhD Thesis, submitted to the Faculty of Computer Science, University of New Brunswick, N.B., Canada.

[22] Stinson, D.R. 1987. *An Introduction to The Design and Analysis of Algorithms.* 2nd edition (Revised). Winnipeg: The Charles Babbage Research Centre.

[23] Varga, R.S. 1962. *Matrix Iterative Analysis.* New Jersey: Prentice-Hall.

[24] Wasow, W.R. 1952. A Note on the Inversion of Matrices by Random Walks. *Math. Tables Other Aids Comput.* 6:78–81.

TABLE 1

The average values of solutions for all unknowns and their errors.

Average	Iterative solution	Monte Carlo Solutions					
		Ergodic			Absorbing		
		ERGIN	ERGDS	ERGWS	ABSIN	ABSDS	ABSWS
Solution	-2.8664	-2.3394	-2.3394	-2.3394	-2.7563	-2.9611	-2.7583
Error	0.0000	1.3967	1.3967	1.3967	0.5505	0.5855	0.5587

TABLE 2

Processing time and speedup for estimating all unknowns.

Quantity	Ergodic codes			Absorbing codes		
	ERGIN	ERGDS	ERGWS	ABSIN	ABSDS	ABSWS
Scalar time	618.42	621.02	485.23	644.35	778.78	601.56
Vector time	329.25	141.43	86.03	163.40	178.48	147.21
Speedup	1.88	4.39	5.64	3.94	4.36	4.09

Time is in seconds.

TABLE 3

Processing time for different number of unknowns.

Number of unknowns	Mode	Processing time in seconds						Iterative (milli-seconds)
		Ergodic			Absorbing			
		ERGIN	ERGDS	ERGWS	ABSIN	ABSDS	ABSWS	
64	S	9.97	10.32	7.93	7.06	7.81	6.96	7.83
	V	5.49	2.04	1.35	1.77	2.24	1.61	5.72
256	S	77.02	80.63	60.54	69.05	79.86	64.81	61.49
	V	42.21	16.20	10.50	15.51	18.27	13.95	44.57
1024	S	618.42	621.02	485.23	644.35	778.78	601.56	483.56
	V	329.25	141.43	86.03	163.40	178.48	147.21	363.34

S: scalar, and V: vector.

TABLE 4

Vectorization speedups for different number of unknowns.

Unknowns	Ergodic codes			Absorbing codes			Iterative method
	ERGIN	ERGDS	ERGWS	ABSIN	ABSDS	ABSWS	
64	1.82	5.06	5.87	3.99	3.49	4.32	1.37
256	1.82	4.98	5.77	4.45	4.37	4.65	1.38
1024	1.88	4.39	5.64	3.94	4.36	4.09	1.33

TABLE 5

Efficiencies of different vector codes relative to those of scalar ABSIN.

Ergodic codes			Absorbing codes		
ERGIN	ERGDS	ERGWS	ABSIN	ABSDS	ABSWS
2.066	3.580	8.201	4.780	0.912	5.023

```
      Subroutine Statei(i)
        r1 = RNUNF()
        do 20 j = 1,n
          if(C(i,j) .ge. r1) then
            i = j
            go to 10
          endif
20      continue
10    Return
      End
```

FIG. 1. *Scalar Code of the inverse method for state determination.*

```
      Subroutine Stateb(i)
         r1   = RNUNF()
         r    = r1 * n + 1.0
         ir   = int(r)
         j    = ir + r - f(ir)
         i    = id(j)
      Return
      End
```

FIG. 2. *Scalar Code of Brown's method for state determination.*

```
BEGIN
1.    FOR i = 1 TO n STEP 1 DO /* Estimate n unknowns */
2.        Y_i = 0; ν = k
3.        FOR j = 1 TO ν STEP 1 DO
4.            Z_j = 1; s_{1_j} = i  /* Initialization */
5.        END DO
6.        FOR τ = 1 TO m STEP 1 DO /* Random walk loop */
7.            Generate ν random numbers ξ_j, j = 1, ··· , ν
8.            FOR j = 1 TO ν STEP 1 DO /* Carry out ν samples */
9.                s_{2_j} = ⌊ξ_j × l_{s_{1_j}}⌋ + 1  /* The weighted sampling method */
10.               IF (s_{2_j} ≠ 0) THEN Z_j = Z_j * H_{s_{1_j}, s_{2_j}} * l_{s_{1_j}}  /* Scoring */
11.               IF (s_{2_j} = 0) THEN Z_j = Z_j * b_{s_{1_j}} * l_{s_{1_j}}  /* Scoring */
12.               s_{1_j} = s_{2_j}
13.           END DO
14.           FOR j = 1 TO ν STEP 1 DO /* Calculate the secondary estimate */
15.               IF (s_{2_j} = 0) THEN Y_i = Y_i + Z_j
16.           END DO
17.           λ=0
18.           FOR j = 1 TO ν STEP 1 DO
19.               IF (s_{2_j} ≠ 0) THEN /* Gather nonabsorbed walks */
20.                   λ = λ + 1
21.                   Z_λ = Z_{s_{2_j}}
22.               END IF
23.           END DO
24.           ν = λ
25.       END DO
26.       Y_i = Y_i/(k − ν)
27.   END DO
END
```

FIG. 3. *A vectorized Monte Carlo algorithm employing an absorbing Markov chain and the weighted sampling method.*

```
BEGIN
1.    FOR i = 1 TO n STEP 1 DO /* Estimate n unknowns */
2.        Y_i = 0
3.        FOR j = 1 TO k STEP 1 DO
4.            Z_j = 1; s_{1_j} = i; T_j = b_i /* Initialization */
5.        END DO
6.        FOR τ = 1 TO m STEP 1 DO /* Random walk loop */
7.            Generate k random numbers ξ_j, j = 1, ···, k
8.            FOR j = 1 TO k STEP 1 DO /* Carry out k samples */
9.                s_{2_j} = ⌊ξ_j × l_{s_{1_j}}⌋ + 1 /* The weighted sampling method */
10.               Z_j = Z_j * H_{s_{1_j}, s_{2_j}} * l_{s_{1_j}} /* Scoring */
11.               T_j = T_j + Z_j * b_{s_{2_j}} /* Scoring */
12.               s_{1_j} = s_{2_j}
13.           END DO
14.           FOR j = 1 TO k STEP 1 DO /* Calculate the secondary estimate */
15.               Y_i = Y_i + T_j
16.           END DO
17.       END DO
18.       Y_i = Y_i/k
19.   END DO
END
```

FIG. 4. *A vectorized Monte Carlo algorithm employing an ergodic Markov chain and the weighted sampling method.*

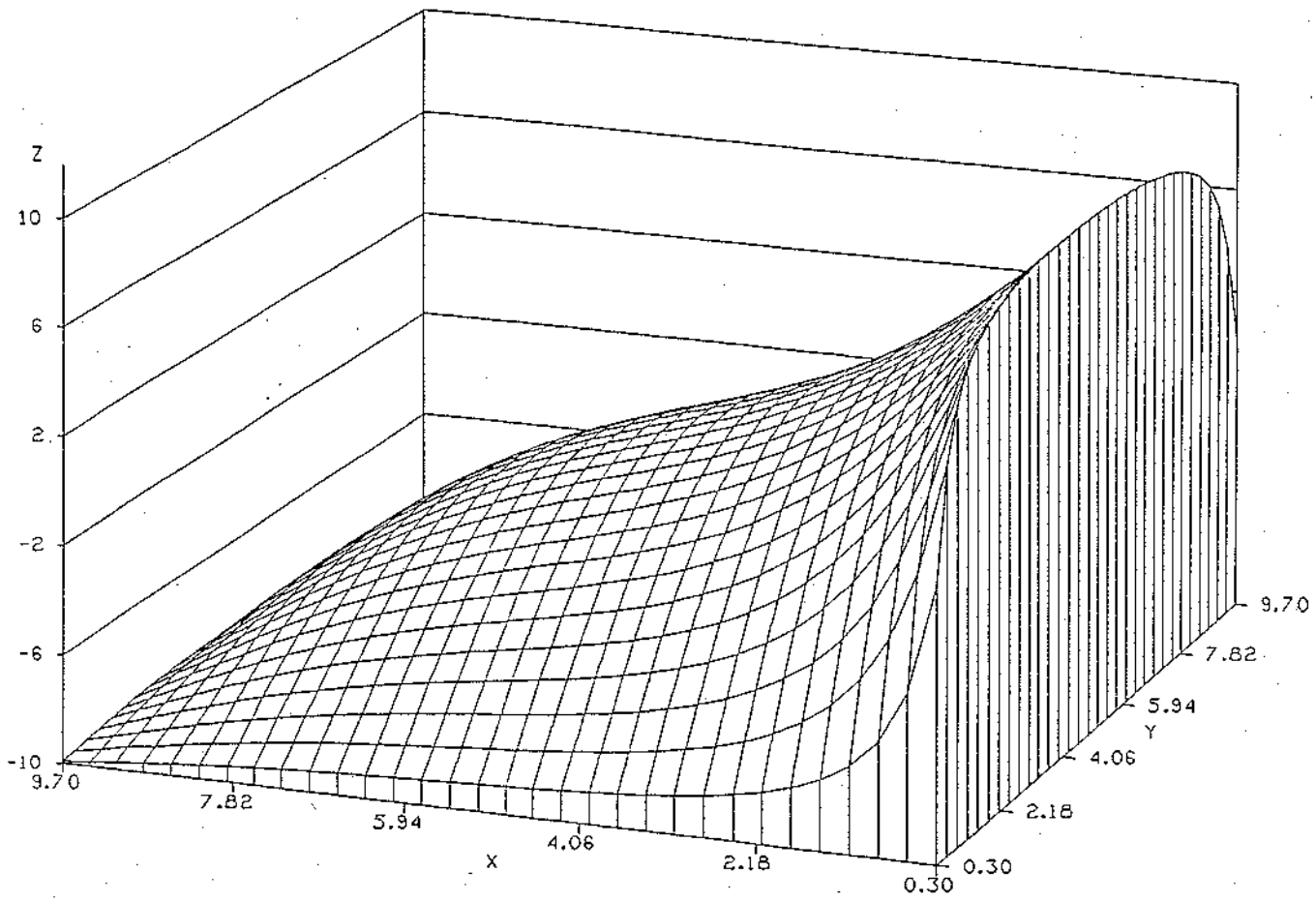

FIG. 5. *Solution obtained using the iterative method (SEIDEL).*

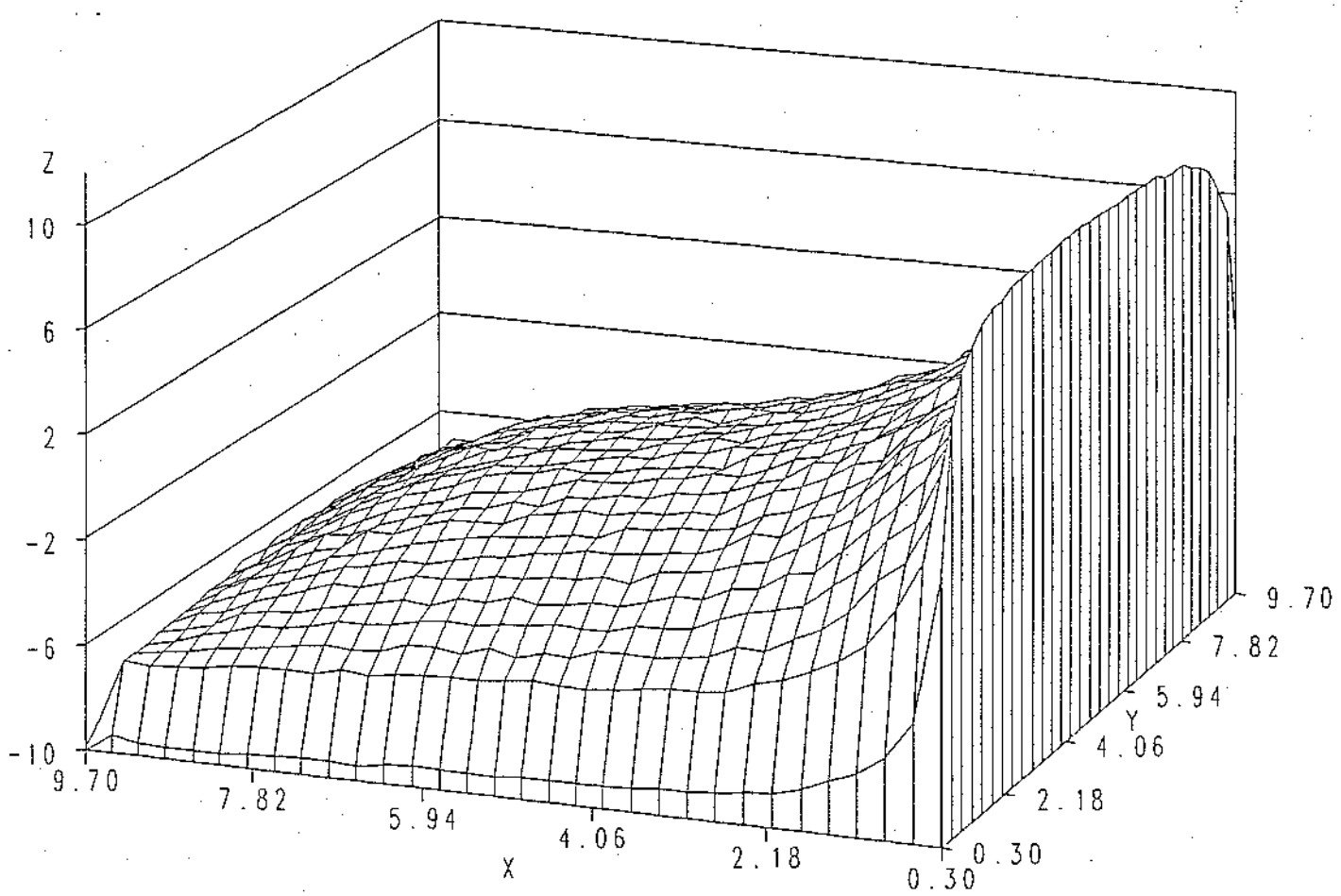

FIG. 6. *Solution obtained using ERGWS, sample size=1000, random walk length=125.*

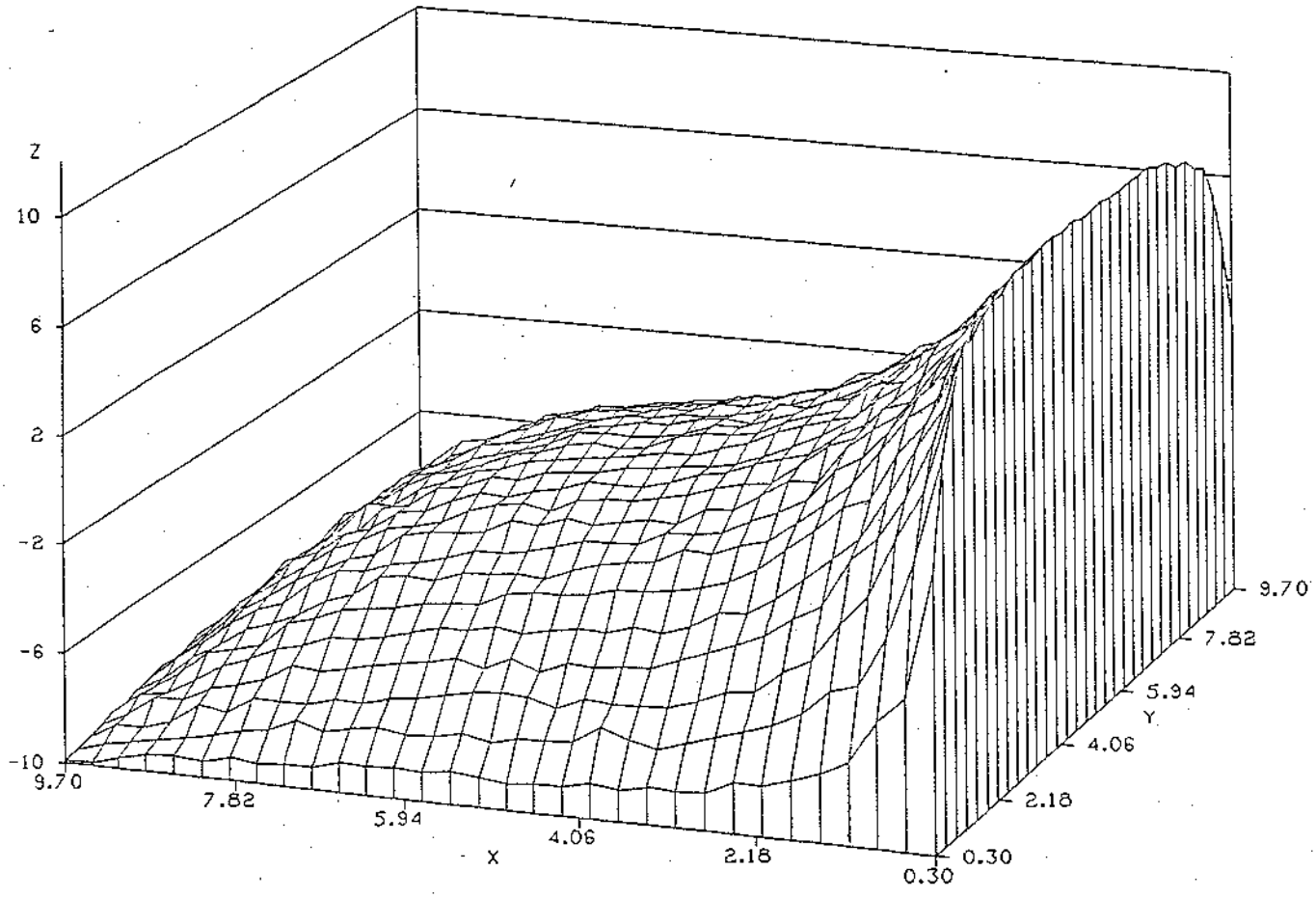

FIG. 7. *Solution obtained using ABSWS, sample size=1000, maximum random walk length=100.*

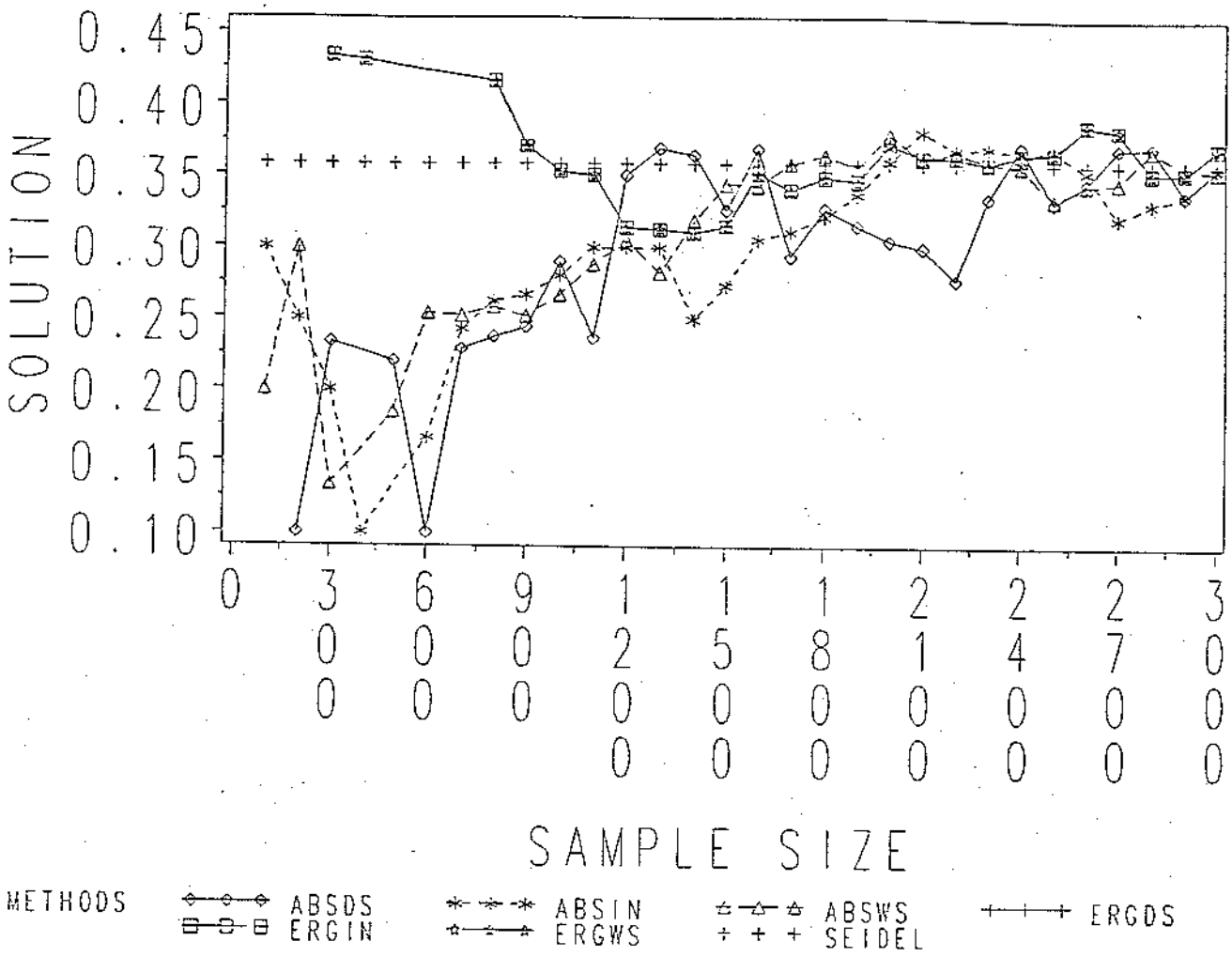

FIG. 8. *Solution of the Laplace's equation at point (2.4242, 2.4242), ERGIN, ERGDS and ERGWS use random walk lengths=80, ABSIN, ABSDS and ABSWS utilize maximum random walk lengths=125.*