Serdica J. Computing **4** (2010), 57–72

Serdica
Journal of Computing

Bulgarian Academy of Sciences
Institute of Mathematics and Informatics

QUASI-MONTE CARLO METHODS FOR SOME LINEAR ALGEBRA PROBLEMS. CONVERGENCE AND COMPLEXITY*

Aneta Karaivanova

Abstract. We present quasi-Monte Carlo analogs of Monte Carlo methods for some linear algebra problems: solving systems of linear equations, computing extreme eigenvalues, and matrix inversion. Reformulating the problems as solving integral equations with a special kernels and domains permits us to analyze the quasi-Monte Carlo methods with bounds from numerical integration. Standard Monte Carlo methods for integration provide a convergence rate of $O(N^{-1/2})$ using N samples. Quasi-Monte Carlo methods use quasirandom sequences with the resulting convergence rate for numerical integration as good as $O((\log N)^k)N^{-1})$. We have shown theoretically and through numerical tests that the use of quasirandom sequences improves both the magnitude of the error and the convergence rate of the considered Monte Carlo methods. We also analyze the complexity of considered quasi-Monte Carlo algorithms and compare them to the complexity of the analogous Monte Carlo and deterministic algorithms.

ACM Computing Classification System (1998): F.2.1, G.1.3, G.3.

Key words: quasi-Monte Carlo methods, matrix computations, Markov chains, quasirandom sequences.

*This work is supported by the National Science Fund of Bulgaria under Grant No. D002-146/16.12.2008.

1. Introduction. In the contemporary literature, a large number of iterative methods for linear algebra problems have been described based on different principles. As a rule the computational schemes for these methods are simple and convenient for use in applied computing. However, each iterative process has a limited area of application, since in the first place an iterative process may turn out to be inapplicable for a given system and in the second place the convergence of the process may be so slow that in practice it turns out to be impossible to achieve a satisfactory approximation to the solution.

First proposed by von Neumann and Ulam, Monte Carlo methods (MCMs) for solving linear algebra problems have been known since the middle of the last century. They give statistical estimates for the elements of the inverse of a matrix or for components of the solution vector of a linear system by performing random sampling of a certain random variable whose expected value is the desired solution. Perhaps the first application of MCMs in linear algebra appeared in a paper by Forsythe and Leibler, [14] in 1950. In the following years significant contributions were made, especially by Wasow [23], Curtiss [6], Halton [17], Hammersly and Handscomb [16] and Sobol [22]. These methods were recognized as useful in the following situations, [24]: when obtaining a quick rough estimate of a solution, which will then be refined by other methods; when the problem is too large or too intricate for any other treatment; when just one component of the solution vector or one element of the inverse matrix is desired.

There has been renewed interest in MCMs in recent times, for example [18, 11, 12, 10, 7, 13, 1, 2, 3, 9, 8]. The primary reason for this is the efficiency of parallel MCMs in the presence of high communication costs. The second reason for the recent interest in MCMs is that the methods have evolved significantly since the early days. Much of the effort in the development of Monte Carlo methods has been in the construction of variance reduction techniques which speed up the computation by reducing the rate of convergence of crude MCM, which is $O(N^{-1/2})$. An alternative approach to acceleration is to change the type of random sequence, and hence improve the behavior by N. **Q**uasi-**M**onte **C**arlo **m**ethods (QMCMs) use quasirandom (also known as low-discrepancy) sequences instead of pseudorandom sequences, with the resulting convergence rate for numerical integration being as good as $O((\log N)^k)N^{-1})$. The first results of using QMCMs for linear algebra problems were presented by Mascagni and Karaivanova, see for example [19, 20].

Quasi-Monte Carlo methods often include standard approaches for variance reduction, although such techniques do not necessarily translate directly. The fundamental feature underlying all QMCMs, however, is the use of a quasi-

random sequence. In this paper the convergence and the complexity of QMCMs for estimating the solution of systems of linear algebraic equations (SLAE), inverting matrices and finding extremal eigenvalues are studied. An error bound for computing a matrix-vector product is established. Numerical experiments with sparse matrices are performed using different quasirandom number (QRN) sequences. The results indicate that for all of the considered problems improvements in both the magnitude of the error and the convergence rate can be achieved using QRNs in place of pseudorandom numbers (PRNs).

2. Formulation of the Problems. Given a matrix $B = \{b_{ij}\}_{i,j=1}^{n}$, $B \in \mathbb{R}^n \times \mathbb{R}^n$ and a vector $b = (b_1, \ldots, b_n)^t \in \mathbb{R}^n$ consider the following three problems:

Problem P1. Evaluating the inner product

$$J(u) = (h, u) = \sum_{i=1}^{n} h_i u_i, \tag{1}$$

where $u \in \mathbb{R}^n$ is the unknown solution of the system $u = Au + f$ and $h \in \mathbb{R}^n$ is a given vector.

Method. First, choose a matrix $M \in \mathbb{R}^n \times \mathbb{R}^n$ such that $MB = I - A$, where $I \in \mathbb{R}^n \times \mathbb{R}^n$ is the identity matrix and $Mb = f$, $f \in \mathbb{R}^n$. Clearly, this is possible, and the choice of M depends on the properties of A, [4]. Then the matrix equation $Bu = b$ becomes

$$u = Au + f. \tag{2}$$

It will be assumed that M and A are both non-singular, and that $|\lambda(A)| < 1$ for all eigenvalues of A (all values $\lambda(A)$ for which $Au = \lambda(A)u$ is satisfied). Then a general stationary linear iteration for solving the system $Bx = b$ may be written

$$u^{(k+1)} = Au^{(k)} + f.$$

The residual corresponding to $u^{(k)}$ is $r^{(k)} = f - (I - A)x^{(k)} = u^{(k+1)} - u^{(k)}$, and assuming that $\|A\| < 1$ it can be easily shown that:

$$\|r^{(k+1)}\| \leq \|A\|^{k+1} \|r^{(0)}\|,$$

$$\|u - u^{(k)}\| \leq \frac{\|A\|^k \|r^{(0)}\|}{1 - \|A\|}.$$

The last inequality gives the error of this method.

Problem P2. Inverting matrices, i.e., computing the matrix C such that $CB = I$ where $B \in \mathbb{R}^n \times \mathbb{R}^n$ is a given real matrix.

Our approach. It will be assumed that the matrix B is non-singular, and that $||\lambda(B)| - 1| < 1$ for all eigenvalues $\lambda(B)$ of B. Construct the matrix $A = I - B$. Under the above assumptions the inverse matrix $C = B^{-1}$ can be presented as

$$C = \sum_{i=0}^{\infty} A^i, \tag{3}$$

and the desired approximation of C is the truncated series (3) with the corresponding truncation error.

Problem P3. Evaluating extremal eigenvalues, $\lambda_{\max}$ and $\lambda_{\min}$ of a matrix A, assuming A is nonsingular and $\lambda_{\min} = \lambda_n < \lambda_{n-1} \leq \cdots \leq \lambda_2 < \lambda_1 = \lambda_{\max}$.

Method. Consider the matrix A and also its *resolvent* matrix $R_q = [I - qA]^{-1} \in \mathbb{R}^{n \times n}$. The following representation

$$[I - qA]^{-m} = \sum_{i=1}^{\infty} q^i C_{m+i-1}^i A^i, \;\; |q\lambda| < 1 \tag{4}$$

is valid because of the well known behavior of the binomial expansion and the spectral theory of linear operators. The eigenvalues of the matrices R_q and A are thus connected by the equality $\mu = \dfrac{1}{1 - q\lambda}$, and the eigenvectors of the two matrices coincide. The largest eigenvalue can be obtained as follows:

- using the Power Method applied to the matrix A, [11]:

$$\lambda_{\max} = \lim_{i \to \infty} \frac{(h, A^i f)}{(h, A^{i-1} f)}, \tag{5}$$

- or using the Power Method applied to the resolvent matrix, [12], for $q > 0$:

$$\mu^{(m)} = \frac{([I - qA]^{-m} f, h)}{([I - qA]^{-(m-1)} f, h)} \;\; \underset{m \to \infty}{\longrightarrow} \;\; \mu = \frac{1}{1 - q\lambda}, \;\; f \in \mathbb{R}^n, h \in \mathbb{R}^n. \tag{6}$$

For computing the smallest eigenvalue we use the fact that for negative values of q the largest eigenvalue, $\mu_{\max}$, of R_q corresponds to the smallest eigenvalue $\lambda_{\min}$ of the matrix A, so we use (6) with $q < 1$.

The convergence rate of the power method is $O\left(\frac{\lambda_2}{\lambda_1}\right)$ and the rate of the resolvent power is $O\left(\frac{1-q\lambda_n}{1-q\lambda_{n-1}}\right)$.

3. Monte Carlo Methods for Linear Algebra. To solve these problems via MCMs (see, for example, [16, 22]) one has to construct for each problem a random process with mean equal to the solution of the desired problem. All of these methods are based on computing a matrix-vector product.

3.1. Computing Matrix-Vector Product. Given a matrix A and vectors $f, h \in R^n$, we want to compute $h^T A^i f$ for some i using a Monte Carlo method. Construct a Markov chain: $k_0 \to k_1 \to \ldots \to k_i$, where $k_j = 1, 2, \ldots, n$ for $j = 1, \ldots, i$ are natural numbers. The rules for constructing the chain are: $P(k_0 = \alpha) = p_\alpha$, $P(k_j = \beta | k_{j-1} = \alpha) = p_{\alpha\beta}$, where p_α is the probability that the chain starts in state α and $p_{\alpha\beta}$ is the transition probability from state α to state β. The probabilities $p_{\alpha\beta}$ define a transition matrix P. We require that $\sum\limits_{\alpha=1}^{n} p_\alpha = 1$, $\sum\limits_{\beta=1}^{n} p_{\alpha\beta} = 1$ for any $\alpha = 1, 2, \ldots, n$, and that the distribution $(p_1, \ldots, p_n)^T$ is permissible to the vector h and similarly the distribution $p_{\alpha\beta}$ is permissible to A [22]. Common constructions are to choose $p_\alpha = 1/n$, $p_{\alpha\beta} = 1/n$, which corresponds to *crude* Monte Carlo, or to choose

$$p_\alpha = \frac{|h_\alpha|}{\sum\limits_{\alpha=1}^{n} |h_\alpha|}; \;\; p_{\alpha\beta} = \frac{a_{\alpha\beta}}{\sum\limits_{\beta=1}^{n} |a_{\alpha\beta}|}, \;\; \alpha = 1, \ldots, n, \tag{7}$$

which corresponds to *importance sampling* for matrix computations — the zero elements will never be visited and the elements with larger magnitude will be visited more often during the random walks on the elements of the matrix.

Now define weights for our Markov chain using the following recursion formula:

$$W_0 = 1, \;\; W_j = W_{j-1} \frac{a_{k_{j-1}k_j}}{p_{k_{j-1}k_j}}, \;\; j = 1, \ldots, i, \tag{8}$$

and the random variable

$$\theta = \frac{h_{k_0}}{p_{k_0}} W_i f_{k_i}. \tag{9}$$

Following [22], it is easy to show that

$$E[\theta] = (h, A^i f), \;\; i = 1, 2, \ldots. \tag{10}$$

Convergence. The Monte Carlo error obtained when computing a matrix-vector product is well known to be [22]:

$$|h^T A^i f - \frac{1}{N} \sum_{s=1}^{N} (\theta)_s| \approx Var(\theta)^{1/2} N^{-1/2},$$

where $Var(\theta) = \{(E[\theta])^2 - E[\theta^2]\}$. If the row sums of A are a constant, $\sum_{j=1}^{n} a_{ij} = a$, and if all the elements of the vector f are constant, and if we furthermore define the initial and transition densities as in (7), then $Var[\theta] = 0$, [20]. For the common case we have $Var[\theta] = (E[h_{k_0} W_m f_{k_m}])^2 - E[(h_{k_0} W_m f_{k_m})^2] \leq (E[h_{k_0} W_m f_{k_m}])^2 \leq \sum_{i=1}^{n} |a_{k_0 i}| . \sum_{i=1}^{n} |a_{k_1 i}| \ldots \sum_{i=1}^{n} |a_{k_{m-1} i}|$, for f and h normalized.

3.2. Monte Carlo estimations. To solve **Problem 1**, define the random variables $\theta[h]$:

$$\theta[h] = \frac{h_{k_0}}{p_{k_0}} \sum_{j=0}^{\infty} W_j f_{k_j}. \tag{11}$$

It is known [22] that the mathematical expectation of this random variable is $E[\theta[h]] = (h, u)$. The partial sum corresponding to (11) is defined as $\theta_i[h] = \frac{h_{k_0}}{p_{k_0}} \sum_{j=0}^{i} W_j f_{k_j}$. Thus the Monte Carlo estimate for (h, x) is

$$(h, x) \approx \frac{1}{N} \sum_{s=1}^{N} \sum_{i=0}^{l_s} \left(\frac{h_{k_0}}{p_{k_0}} W_i f_{k_i} \right)_s,$$

where N is the number of chains and $\theta_i[h]_s$ is the value of $\theta_i[h]$ taken over the s-th chain. This estimate has a statistical error of size $O(Var(\theta_i)^{1/2} N^{-1/2})$.

If we are interested in one component of the solution, x_r, then we choose the vector h with coordinates $h(i) = 0$ for $i \neq r$, and $h(r) = 1$, then $x_r = E[\theta[h]]$ for the above defined random variable θ.

To solve **Problem 2**, i.e., to compute the element $c_{rr'}$ of the matrix inverse to A, we use the following equality, [22]:

$$c_{rr'} = E\left\{\sum_{i|k_i=r'} W_i\right\}, \tag{12}$$

where $(i|k_i = r')$ means summation only for weights W_i for which $k_i = r'$ and $C = \{c_{rr'}\}_{r,r'=1}^{n}$. The Monte Carlo estimation is then:

$$c_{rr'} \approx \frac{1}{N}\sum_{s=1}^{N}\left[\sum_{(j|k_j=r')} W_j\right]_s.$$

To solve **Problem 3**, we use the equality (10) and also the following one, ([12]):

$$E\left[\sum_{i=0}^{\infty} q^i C_{i+m-1}^{i}\frac{h_{k_0}}{p_{k_0}} W_i f(x_i)\right] = (h, [I - qA]^{-m} f), \;\; m = 1, 2, \ldots,$$

which allow us to express the estimates (5) and (6) as:

$$\lambda_{\max} \approx \frac{E[W_i f_{k_i}]}{E[W_{i-1} f_{k_{i-1}}]}. \tag{13}$$

and

$$\lambda \approx \frac{1}{q}\left(1 - \frac{1}{\mu^{(m)}}\right) = \frac{E\left[\sum\limits_{i=1}^{\infty} q^{i-1} C_{i+m-2}^{i-1} W_i f(x_i)\right]}{E\left[\sum\limits_{i=0}^{\infty} q^i C_{i+m-1}^{i} W_i f(x_i)\right]}. \tag{14}$$

We use MCM for an approximate calculation of these expected values:

$$\lambda_{\max} \approx \frac{\sum\limits_{s=1}^{N}(W_m f_{k_m})_s}{\sum\limits_{s=1}^{N}(W_{m-1} f_{k_{m-1}})_s}$$

$$\lambda_{\min} =\approx \frac{\sum\limits_{s=1}^{N}\left(\left[\sum\limits_{i=0}^{l} q^i C_{i+m-1}^{i-1} W_{i+1} f(x_{i+1})\right]\right)_s}{\sum\limits_{s=1}^{N}\left(\left[\sum\limits_{i=0}^{l} q^i C_{i+m-1}^{i} W_i f(x_i)\right]\right)_s}.$$

We note that in (13) the length of the Markov chain, l, is equal to the number of iterations, i, in the power method. However in (14) the length of the Markov chain is equal to the number of terms in the truncated series for the resolvent matrix. In this second case the parameter m corresponds to the number of power iterations.

4. Quasi-Monte Carlo Methods for Matrix Computations. Recall that all presented methods are based on computing $h^T A^i f$ (see (5) and (6)). But computing $h^T A^i f$ is equivalent to computing an $(i+1)$-dimensional integral. Thus we may analyze using QRNs with bounds from numerical integration. We do not know A^i explicitly, but we do know A and we perform random walks on the elements of the matrix to compute approximately $h^T A^i f$.

If we define $G = [0, n)$, $G_i = [i-1, i)$, $f(x) = f_i$ for $x \in G_i$, $a(x, y) = a_{ij}$ for $x \in G_i, y \in G_j$, $h(x) = h_i$ for $x \in G_i$, where $i, j = 1, \ldots, n$, we can consider computing $h^T A^i u$ to be equivalent to computing a $(i+1)$-dimensional integral. Now, first consider the scalar product $h^T A f$ bearing in mind that the vectors h, f and the matrix A are normalized with factors of $1/\sqrt{n}$ and $1/n$ respectively and denoted by h_N, A_N and f_N. In this case

$$h_N^T A_N f_N = \int_0^1 \int_0^1 h(x)a(x,y)f(y)dxdy = \sum_{i=1}^{n}\sum_{j=1}^{n} \int_{\frac{i-1}{n}}^{\frac{i}{n}} \int_{\frac{j-1}{n}}^{\frac{j}{n}} h_i a_{ij} f_j dxdy =$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} h_i a_{ij} f_j v_{ij},$$

where $v_{ij} = \dfrac{1}{n^2}$ is the volume of the $Box(ij) = \left[\dfrac{i-1}{n}, \dfrac{i}{n}\right) \times \left[\dfrac{j-1}{n}, \dfrac{j}{n}\right)$.

On the other hand, consider a two-dimensional sequence of N points (x_s, y_s), then

$$\frac{1}{N}\sum_{s=1}^{N} h(x_s)a(x_s,y_s)f(y_s) = \frac{1}{N}\sum_{\#\text{ of boxes}} \left(\sum_{(x_s,y_s)\in Box(ij)} h(x_s)a(x_s,y_s)f(y_s) \right) =$$

$$\frac{1}{N}\left(\sum_{i=1}^{n}\sum_{j=1}^{n} h_i a_{ij} f_j [\#\text{ of points in } Box(ij)] \right).$$

Thus, the difference between the scalar product and its estimated value becomes:

$$|h_N^T A_N f_N - \frac{1}{N}\sum_{s=1}^{N} h(x_s)a(x_s,y_s)f(y_s)| \le \sum_{i=1}^{n}\sum_{j=1}^{n}|h_i a_{ij} f_j| D_N^* = |h|^T|A||f|.D_N^*,$$

where $|h| = \{|h_i|\}_{i=1}^n$, $A = \{|a_{ij}|\}_{i,j=1}^n$ and $|f| = \{|f_i|\}_{i=1}^n$.
Analogously, considering $h^T A^l f$ and an $l+1$-dimensional sequence we have

$$|h_N^T A_N^l f_N - \frac{1}{N}\sum_{s=1}^{N} h(x_s)a(x_s,y_s)\dots a(z_s,w_s)f(w_s)| \le |h|^T|A|^l|f|.D_N^*. \tag{15}$$

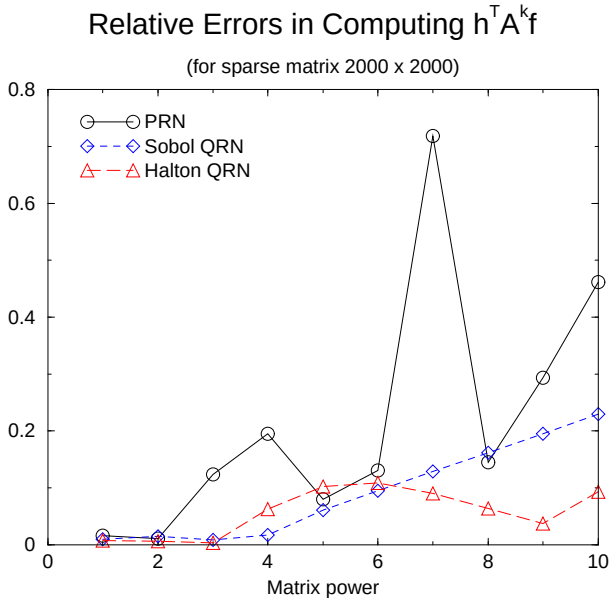

Fig. 1. Relative errors in computing $h^T A^k f$ for $k = 1, 2, \dots, 10$ for a sparse matrix 2000 $\times$ 2000. The corresponding Markov chains are realized using PRN, Sobol and Halton sequences

Let A be a general sparse matrix with d_i nonzero elements per row. The following mapping procedure corresponds to the importance sampling approach:

$$G = [0, 1)$$

$$G_i = \left[\frac{\sum_{k'=1}^{i-1} |a_{ik'}|}{\sum_{k'=1}^{n} |a_{ik'}|}, \frac{\sum_{k'=1}^{i} |a_{ik'}|}{\sum_{k'=1}^{n} |a_{ik'}|} \right), \quad i = 1, \ldots, n$$

and summation on k' means summation only on nonzero elements.

$$a(x, y) = a_{ij}, x \in G_i, \ y \in G_j, \ \ i = 1, \ldots, n, \ j = 1, \ldots, d.$$

Often, the vectors f and h are chosen to be $(1, 1, \ldots, 1)$, so $h(x) = 1, x \in G$, $f(x) = 1$, $x \in G$.

In this case after similar calculation we prove that the bound on the error (for non-normalized matrix) is given by:

$$|h^T A f - \frac{1}{N} \sum_{s=1}^{N} h(x_s) a(x_s, y_s) f(y_s)| \leq (d\|A\|)^l D_N^*,$$

where d is the mean value of the nonzero elements per row, l is the length of the Markov chain, D_N^* is the star discrepancy of the sequence used, and $\|A\| < 1$.

Let us recall that usually the average number d of nonzero entries per row is much smaller than the size of the matrix n, $d \ll n$. Thus the order of the above estimation is the order of D_N^* which is $O((\log^l N) N^{-1})$.

Convergence and Complexity. In the Monte Carlo methods there are two kinds of errors that control the convergence: systematic, which comes from the method, and stochastic, which comes from the approximation of the mean value with an averaged sum. The complexity of a Monte Carlo method is the product of the expected value of the length of the corresponding walk (Markov chain), and a number of walks (chains). For the problems that we considered in this paper, we have:

- Scalar product of the solution ($x = Ax + f$) and an element of the inverse matrix $A^{-1} = C = \{c_{rr'}\}$:

  The computational complexity is lN, where l is the length of the performed walks (Markov chain) which for MCM is $l = E[l_s]$, and for QMCM l the dimension of the quasirandom sequence; l depends on the spectrum of the matrix A. Let us note that the first few steps of a random (quasirandom) walk tend to improve results greatly, whereas many additional steps would be necessary to refine the result to sufficient accuracy. We suggest using these methods with a relatively small l for a quick rough estimation.

The convergence for MCM and QMCM in this case is
$O\left(\frac{\|A\|^l\|r^{(0)}\|}{1-\|A\|}+\sigma N^{-1/2}\right)$ and $O\left(\frac{\|A\|^l\|r^{(0)}\|}{1-\|A\|}+(\log^l N)N^{-1}\right)$
correspondingly.

- Largest eigenvalue:

  Here the computational complexity is mN where m is the power in the power method, it is independent of the size of the matrix n.

  The convergence for MCM and QMCM in this case is $O\left(\left\|\frac{\lambda_2}{\lambda_1}\right\|^m+\sigma N^{-1/2}\right)$ and $O\left(\left\|\frac{\lambda_2}{\lambda_1}\right\|^m+(\log^m N)N^{-1}\right)$ correspondingly.

- Smallest eigenvalue:

  Computational complexity: $4lN$ where $l=E[l_s]$ for MCM, and l the dimension of the quasirandom sequence for QMCM; here l depends on again on the spectrum of the matrix A.

  The convergence for MCM and QMCM in this case is $O\left(\left\|\frac{\mu_2}{\mu_1}\right\|^m+\sigma N^{-1/2}\right)$ and $O\left(\left\|\frac{\mu_2}{\mu_1}\right\|^m+(\log^l N)N^{-1}\right)$ correspondingly.

Thus, the corresponding MCM and QMCM have the same computational complexity, while their errors from approximate calculation of mean values are:

- Products of two factors (first depends on A, second on the sequence). The order is $N^{-1/2}$ for MCM and $(\log^{m+1} N)N^{-1}$ for QMCM.
- Probabilistic error bound for MCM, worst-case bound (inequality) for QMCM.

5. Numerical Results. Why are we interested in quasi-MCMs for the eigenvalue problem? Because the computational complexity of QMCMs is bounded by $O(lN)$ where N is the number of chains, and l is the mathematical expectation of the length of the Markov chains, both of which are independent of matrix size n. This makes QMCMs very efficient for large, sparse problems, for which deterministic methods are not computationally efficient.

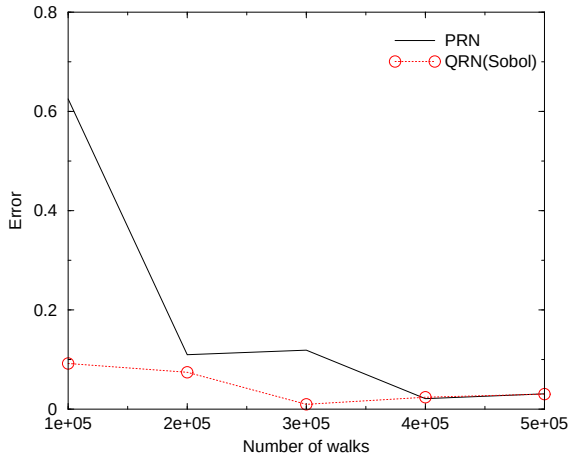

Fig. 2. Accuracy versus number of walks for computing (h, x), where x is the solution of a system with 2000 equations

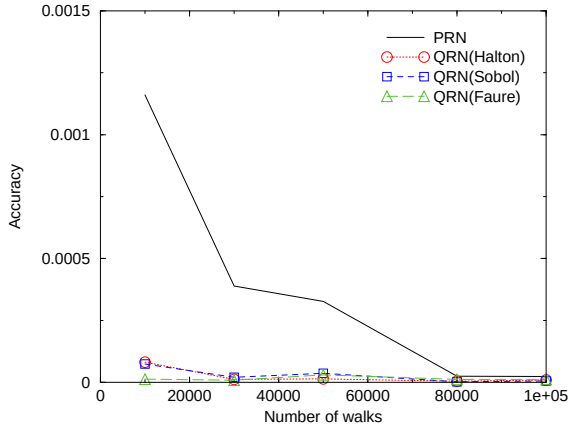

Fig. 3. Accuracy versus number of walks for computing one component, x_{64}, of the solution for a system with 1024 equations2

Numerical tests were performed on general sparse matrices using PRNs and Sobol, Halton and Faure QRNs. The relative errors in computing $h^T A^k f$ with A a sparse matrix of order 2000 and $h = f = (1, 1, \ldots, 1)$ are presented in Figure 1. The results confirm that the QRNs produce higher precision results than PRNs. The more important fact is the smoothness of the quasirandom "iterations" with k. This is important because these eigenvalue algorithms com-

pute a Raleigh quotient which requires the division of values from consecutive iterations.

The accuracy when computing a scalar product of a given vector h and the solution of a system of size 2000 is presented in Figure 2. The accuracy in computing x_{64} is presented in Figure 3 The results confirm that using QRNs we obtain much higher accuracy than using PRNs.

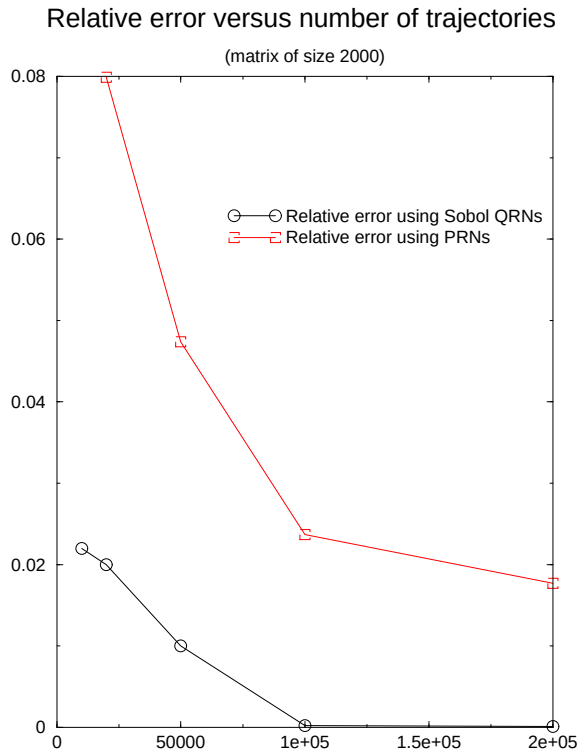

Fig. 4. Relative errors in computing $\lambda_{\max}$ using power MCM and quasi-MCM with different length of Markov chains for a sparse matrix 2000 × 2000

Figure 4 graphs the relative errors of the power Monte Carlo algorithm and power quasi-Monte Carlo algorithm (using the Sobol sequence) for computing the dominant eigenvalue for a sparse matrix of size 2000. Note that with 20000 points our Sobol sequence achieves about the same accuracy as when 100000 or more PRNs are used. The fact that similar accuracy with these kinds of calculations can be achieved with QRNs in a fraction of the time required with PRNs is very significant. This is the major reason for using QRNs over PRNs: an overall decreased time to solution.

6. Conclusions. Quasi-Monte Carlo methods and QRNs are powerful tools for accelerating the convergence of ubiquitous MCMs. For computing the solution of a system of linear equations, scalar products of the solution and any given vector, elements of the inverse matrix, and extremal eigenvalues of a matrix, it is possible to accelerate the convergence of well-known Monte Carlo methods with QRNs. Moreover, we have seen smoother convergence with the increasing length of the walks which is very important for computing the eigenvalues. In the same time MCMs and QMCMs have the same computational complexity.

REFERENCES

[1] Alexandrov V., A. Karaivanova. Parallel Monte Carlo Algorithms for Sparse SLAE using MPI. Lecture Notes in Computer Science, Springer, **1697** (1999), 283–290.

[2] Alexandrov V., I. Dimov, A. Karaivanova, C. J. Tan. Parallel Monte Carlo ALgorithms for Information retrieval. *Mathematics and Computers in Simulation*, Elsevier, **62** (2003), 289–295.

[3] Alexandrov V., A. Karaivanova. Finding the smallest eigenvalue by the Inverse Monte Carlo Method with Refinement. Lecture Notes in Computer Science, **3516** (2005), Springer, 766–774.

[4] Burden R. L., J. D. Faires. Numerical Analysis. Fifth Edition, Brooks/Cole Publishing Company, Pacific Grove, California, 1996.

[5] Caflisch R. E. Monte Carlo and quasi-Monte Carlo methods. *Acta Numerica*, **7** (1998), 1–49.

[6] Curtiss J. H. Monte Carlo methods for the iteration of linear operators. *Journal of Mathematical Physics*, **32** (1954), 209–323.

[7] Danilov D., S. Ermakov, J. Halton. Asymptotic complexity of Monte Carlo methods for solving linear systems. *Journal of Statistical Planning and Inference*, **85** (2000.), 5–18.

[8] Dimov I. T., B. Philippe, A. Karaivanova, C. Weinbrauch. Robustness and Applicability of Markov Chain Monte Carlo Algorithms for Eigenvalue Problems. *Journal of Applied Mathematical Modelling* **32** (2008), 1511–1529.

[9] Dimov I., V. Alexandrov, R. Papancheva, C. Weihrauch. Monte Carlo Numerical Treatment of Large Linear Algebra Problems. Lecture Notes in Computer Science, **4487** (2007), Springer, 747–754.

[10] Dimov I., V. Alexandrov, A. Karaivanova. Resolvent Monte Carlo Methods for Linear Algebra Problems. *Mathematics and Computers in Simulations*, **55** (2001), 25–36.

[11] Dimov I., A. Karaivanova. Iterative Monte Carlo algorithms for linear algebra problems, Lecture Notes in Computer Science, **1196** (1996), Springer, 66–77.

[12] Dimov I., A. Karaivanova. Parallel computations of eigenvalues based on a Monte Carlo approach. *Monte Carlo Methods and Applications*. **4** (1998), No 1, 33–52.

[13] Fathi B., B. Liu, V. Alexandrov. Mixed Monte Carlo Parallel Algorithms for Matrix Computation. Lecture Notes in Computer Science, **2330** (2002), part II, Springer, 609–618.

[14] Forsythe G., R. Leibler. Matrix Inversion by a Monte Carlo Method. *Math. Tables and Other Aids to Computation*, **4** (1950), 127–147.

[15] Golub G. H., C. F. Van Loon. Matrix computations. The Johns Hopkins Univ. Press, Baltimore, 1996.

[16] Hammersley J. M. , D. C. Handscomb. Monte Carlo methods, John Wiley & Sons, inc., New York, London, Sydney, Methuen, 1964.

[17] Halton J. Sequential Monte Carlo. In: Proceedings of the Cambridge Philosophical Society, 58 part, **1** (1962), 57–78.

[18] Halton J. H. Sequential Monte Carlo Techniques for the Solution of Linear Systems. *SIAM Journal of SC*, **9** (1994), 213–257.

[19] Mascagni M., A. Karaivanova. Matrix Computations Using Quasirandom Sequences. Lecture Notes in Computer Science, **1988** (2001), Springer, 552–559.

[20] Mascagni M., A. Karaivanova, A Parallel Quasi-Monte Carlo Method for Computing Extremal Eigenvalues. Lecture Notes in Statistics, (Eds K.-T. Fang, F. J. Hickernell, H. Niederreiter), Springer, 2002, 369–380.

[21] Niederreiter H. Random number generation and quasi-Monte Carlo methods. SIAM, Philadelphia, 1992.

[22] Sobol, I. M. Monte Carlo numerical methods. Nauka, Moscow, 1973 (in Russian).

[23] Wasow W. A note on the inversion of matrices by random walks. *Math. Tables and Other Aids to Computation*, **6** (1952), 78–78.

[24] Westlake J. A Handbook of Numerical Matrix Inversion and Solution of Linear Equations, J. Wiley & Sons, New York, London, Sydney, 1968.

Institute for Parallel Processing
Bulgarian Academy of Sciences
Acad. G. Bonchev Str. Bl. 25 A
1113 Sofia, Bulgaria
e-mail: anet@parallel.bas.bg

Received November 21, 2009
Final Accepted February 4, 2010