A Parallel Quasi-Monte Carlo Method for Computing Extremal Eigenvalues

Michael Mascagni[1] and Aneta Karaivanova[1,2]

[1] Florida State University, Department of Computer Science, Tallahassee, FL 32306-4530, USA
[2] Bulgarian Academy of Sciences, Central Laboratory for Parallel Processing, 1113 Sofia, Bulgaria

Abstract *The convergence of Monte Carlo methods for numerical integration can often be improved by replacing* **p***seudo***r***andom* **n***umbers (PRNs) with more uniformly distributed numbers known as* **q***uasi***r***andom* **n***umbers (QRNs). In this paper the convergence of a Monte Carlo method for evaluating the extremal eigenvalues of a given matrix is studied when quasirandom sequences are used. An error bound is established and numerical experiments with large sparse matrices are performed using three different QRN sequences: Sobol', Halton and Faure. The results indicate:*

- *An improvement in both the magnitude of the error and in the convergence rate that can be achieved when using QRNs in place of PRNs.*
- *The high parallel efficiency established for Monte Carlo methods is preserved for quasi-Monte Carlo methods in this case. The execution time for computing an extremal eigenvalue of a large, sparse matrix on p processors is bounded by $O(lN/p)$, where l is the length of the Markov chain in the stochastic process and N is the number of chains, both of which are independent of the matrix size.*

Keywords: Monte Carlo methods, quasi-Monte Carlo methods, eigenvalues, Markov chains, parallel computing, parallel efficiency

1 Introduction

Monte **C**arlo **m**ethods (MCMs) are based on the simulation of stochastic processes whose expected values are equal to computationally interesting quantities. Despite the universality of MCMs, a serious drawback is their slow convergence, which is based on the $O(N^{-1/2})$ behavior of the size of statistical sampling errors. This represents a great opportunity for researchers in computational science. Even modest improvements in the Monte Carlo method can have substantial impact on the efficiency and range of applicability for Monte Carlo methods. Much of the effort in the development of Monte Carlo has been in construction of variance reduction methods which speed up the computation by reducing the constant in the $O(N^{-1/2})$ expression. An alternative approach to acceleration is to change the choice of random sequence used. Quasi-Monte Carlo methods use quasirandom (also known as low-discrepancy) sequences instead of pseudorandom sequences and can achieve convergence of $O(N^{-1})$ in certain cases.

QRNs are constructed to minimize a measure of their deviation from uniformity called discrepancy. There are many different discrepancies, but let us consider the most common, the star discrepancy. Let us define the star discrepancy of a one-dimensional point set, $\{x_n\}_{n=1}^N$, by

$$D_N^* = D_N^*(x_1, \ldots, x_N) = \sup_{0 \le u \le 1} \left| \frac{1}{N} \sum_{n=1}^{N} \chi_{[0,u)}(x_n) - u \right| \tag{1}$$

where $\chi_{[0,u)}$ is the characteristic function of the half open interval $[0, u)$. The mathematical motivation for quasirandom numbers can be found in the classic Monte Carlo application of numerical integration. We detail this for the trivial example of one-dimensional integration for illustrative simplicity.

Theorem (Koksma-Hlawka, [7]): if $f(x)$ has bounded variation, $V(f)$, on $[0, 1)$, and $x_1, \ldots, x_N \in [0, 1]$ have star discrepancy D_N^*, then:

$$\left| \frac{1}{N} \sum_{n=1}^{N} f(x_n) - \int_0^1 f(x)\, dx \right| \le V(f) D_N^*, \tag{2}$$

The star discrepancy of a point set of N truly random numbers in one dimension is $O(N^{-1/2}(\log\log N)^{1/2})$, while the discrepancy of N quasirandom numbers can be as low as N^{-1}. [1] In $s > 3$ dimensions it is rigorously known that the discrepancy of a point set with N elements can be no smaller than a constant depending only on s times $N^{-1}(\log N)^{(s-1)/2}$. This remarkable result of Roth, [13], has motivated mathematicians to seek point sets and sequences with discrepancies as close to this lower bound as possible. Since Roth's remarkable results, there have been many constructions of low discrepancy point sets that have achieved star discrepancies as small as $O(N^{-1}(\log N)^{s-1})$. Most notably there are the constructions of Hammersley, Halton, [5], Sobol', [14], Faure, [4], and Niederreiter, [12].

While QRNs do improve the convergence of applications like numerical integration, it is by no means trivial to enhance the convergence of all MCMs. In fact, even with numerical integration, enhanced convergence is by no means assured in all situations with the näive use of quasirandom numbers, [2,11].

In this paper we study the applicability of quasirandom sequences for solving the eigenvalue problem. The paper is organized as follows: §2 briefly explains QRN generation. §3 we present two MCMs for computing extremal eigenvalues. Both algorithms are based a stochastic application of the power method. One uses MCMs to compute high powers of the given matrix, while the other, high powers of the related resolvent matrix. Then in §4 we describe how to modify these MCMs by the careful use of QRNs. In §5 we present some numerical results that confirm the efficacy of the proposed quasi-MCMs

[1] Of course, the N optimal quasirandom points in $[0, 1)$ are the obvious: $\frac{1}{(N+1)}, \frac{2}{(N+1)}, \cdots \frac{N}{(N+1)}$.

and that they retain the parallel efficiency of the analogous MCMs. Finally, in §6 we present some brief conclusions and comment on future work.

2 Quasirandom Number Generation

Perhaps the best way to illustrate the difference between QRNs and PRNs is with a picture. Thus in *Figure 1* we plot 4096 tuples produced by successive elements from a 64-bit PRN generator from the `SPRNG` library, [10] developed by one of the the authors (MM). These tuples are distributed in a manner consistent with real random tuples. In *Figure 2* we see 4096 quasirandom tuples formed by taking the 2nd and 3rd dimensions from the Soboĺ sequence. It is clear that the two figures look very different and that *Figure 2* is much more uniformly distributed. Both plots have the same number of points, and the largest "hole' in *Figure 1* is much larger than in *Figure 2*. This illustrates quite effectively the qualitative meaning of low discrepancy.

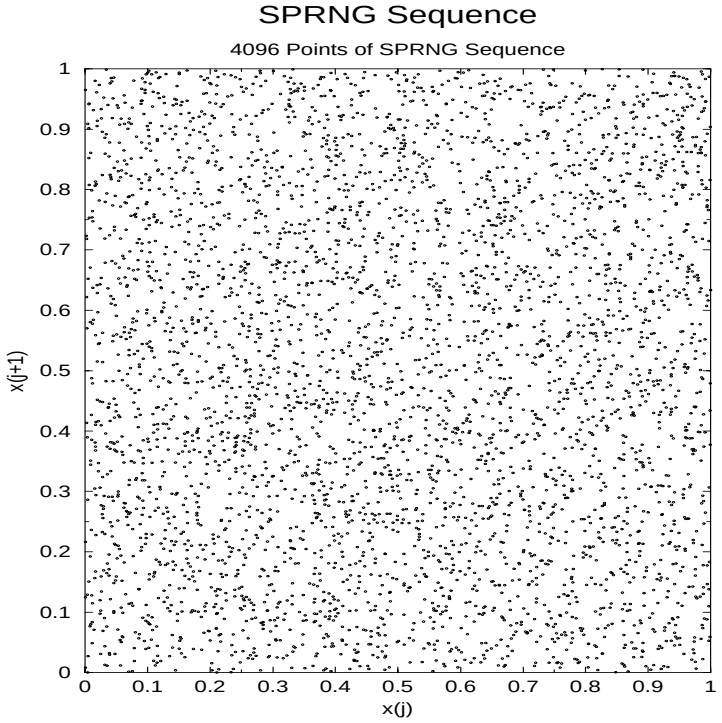

Fig. 1. Tuples produced by successive elements from the `SPRNG` pseudorandom number generator, `lfib`.

The first high-dimensional QRN sequence was proposed by Halton, [5], and is based on the Van der Corput sequence with different prime bases

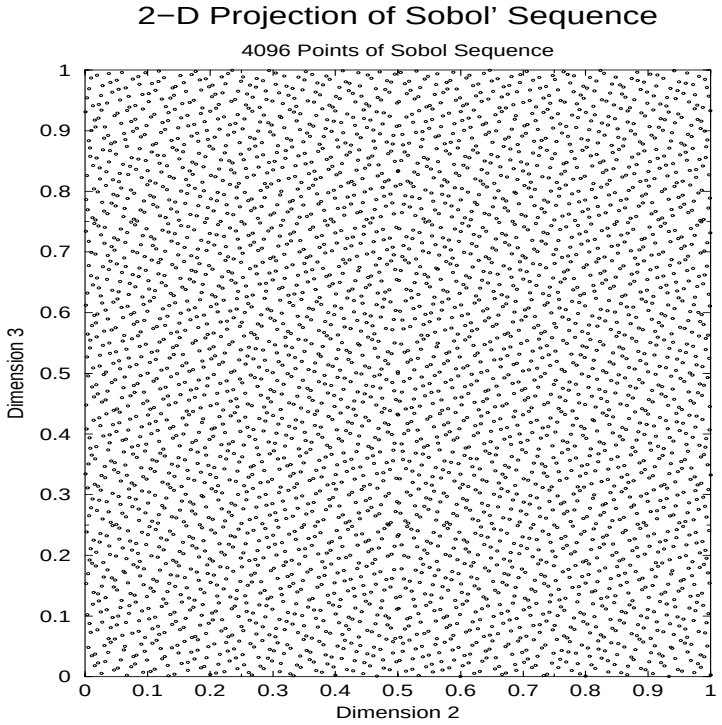

Fig. 2. Tuples produced by the 2nd and 3rd dimension of the Soboĺ sequence.

for each dimension. The jth element of Van der Corput sequence base b is defined as $\phi_b(j-1)$ where $\phi_b(\cdot)$ is the radical inverse function and is computed by writing $j-1$ as an integer in base b, and then flipping the digits about the ordinal (decimal) point. Thus if $j-1 = a_n \dots a_0$ in base b, then $\phi_b(j-1) = 0.a_0 \dots a_n$. As an illustration, in base $b = 2$, the first elements of the Van der Corput sequence are $\frac{1}{2}, \frac{1}{4}, \frac{3}{4}, \frac{1}{8}, \frac{5}{8}, \frac{3}{8}, \frac{7}{8}$, while with $b = 3$, the sequence begins with $\frac{1}{3}, \frac{2}{3}, \frac{1}{9}, \frac{4}{9}, \frac{7}{9}, \frac{2}{9}, \frac{5}{9}, \frac{8}{9}$. With $b = 2$, the Van der Corput sequence methodically breaks the unit interval into halves in a manner that never leaves a gap that is too big. With $b = 3$, the Van der Corput sequence continues with its methodical ways, but instead recursively divides intervals into thirds.

Another way to think of the Van der Corput sequence (with $b = 2$) is to think of taking the bits in $j-1$, and associating with the ith bit the number v_i. Every time the ith bit is one, you exclusive-or in v_i, what we will call the ith direction number. For the Van der Corput sequence, v_i is just a bit sequence with all zeros and a one in the ith location counting from the left. Perhaps the most popular QRN sequence, the Soboĺ sequence, can be thought of in these terms. Soboĺ, [14], found a clever way to define more complicated direction numbers than the "unit vectors" which define the

Van der Corput sequence. Besides producing very good quality QRNs, the reliance on direction numbers means that the Soboĺ sequence is both easy to implement and very computationally efficient.

Since this initial work, Faure, Niederreiter and Soboĺ, [4], [12], [14], chose alternate methods based on another sort of finite field arithmetic that utilizes primitive polynomials with coefficients in some prime Galois field. All of these constructions of quasirandom sequences have discrepancies that are $O(N^{-1}(\log N)^s)$. What distinguishes them is the asymptotic constant in the discrepancy, and the computational requirements for implementation. However, practice has shown that the provable size of the asymptotic constant in the discrepancy is a poor predictor of the actual computational discrepancy displayed by a concrete implementation of any of these QRN generators. There are existing implementations of the Halton, Faure, Niederreiter and Soboĺ sequences, [1], that are computationally efficient. Each of these sequences is initialized to produce quasirandom s-tuples and each one of these requires the initialization of s one-dimensional quasirandom streams.

3 Computing Extremal Eigenvalues

Let A be a large $n \times n$ matrix. In most cases we also assume that A is sparse. Consider the problem of computing one or more eigenvalues of A, i.e., the values λ that satisfy the equation

$$Au = \lambda u. \tag{3}$$

Suppose the n eigenvalues of A are ordered as follows $|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_{n-1}| > |\lambda_n|$.

Consider also computing the eigenvalues of the *resolvent* matrix of a given matrix: $R_q = [I - qA]^{-1} \in \mathbb{R}^{n \times n}$. If $|q\lambda| < 1$, then the following representation holds: $[I - qA]^{-m} = \sum_{i=0}^{\infty} q^i C_{m+i-1}^i A^i$. Here the coefficients C_{m+i-1} are binomial coefficients and the previous expression is merely an application of the binomial formula. The eigenvalues of the matrices R_q and A are related to one another through the equality $\mu = \frac{1}{1-q\lambda}$, and the eigenvectors of the two matrices coincide[2].

Let $f \in \mathbb{R}^n, h \in \mathbb{R}^n$ be given, n-dimensional vectors. We use them to apply the power method, ([3]), to approximately compute the desired eigenvalues via following iterative process for both A and R_q:

$$\lambda^{(m)} = \frac{(h, A^m f)}{(h, A^{m-1} f)} \underset{m \to \infty}{\longrightarrow} \lambda_{max} \tag{4}$$

[2] If $q > 0$ the largest eigenvalue μ_{max} of the resolvent matrix corresponds to the largest eigenvalue λ_{max} of the matrix A, but if $q < 0$, then μ_{max}, corresponds to the smallest eigenvalue λ_{min} of the matrix A.

$$\mu^{(m)} = \frac{(h,[I-qA]^{-m}f)}{(h,[I-qA]^{-(m-1)}f)} \underset{m\to\infty}{\longrightarrow} \mu_{max} = \frac{1}{1-q\lambda}. \tag{5}$$

Given that both these equations require on the computation of matrix-vector products, we may apply the well known matrix-vector multiplication, [6] Monte Carlo method to obtain a stochastic estimate of the desired eigenvalues. To derive this desired MCM, we begin with a Markov chain $k_0 \to k_1 \to \ldots \to k_i$, on the natural numbers, $k_j = 1, 2, \ldots, n$ for $j = 1, \ldots, i$. We then define an initial density vector, $p = \{p_\alpha\}_{\alpha=1}^n$, to be permissible to the vector h and a transition density matrix, $P = \{p_{\alpha\beta}\}_{\alpha\beta=1}^n$, to be permissible to A, [3].[3] We then define the following random variable on the given Markov chain:

$$W_0 = \frac{h_{k_0}}{p_{k_0}}, \;\; W_j = W_{j-1}\frac{a_{k_{j-1}k_j}}{p_{k_{j-1}k_j}}, \;\; j = 1, \ldots, i. \tag{6}$$

Monte Carlo methods for computing the extremal eigenvalues are based on the following equalities ([3]):

$$(h, A^i f) = E[W_i f_{k_i}], \;\; i = 1, 2, \ldots,$$

and

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

This gives us the corresponding estimates for the desired eigenvalues as:

$$\lambda_{max} \approx \frac{E[W_i f_{k_i}]}{E[W_{i-1} f_{k_{i-1}}]}, \tag{7}$$

and

$$\lambda \approx \frac{1}{q}\left(1 - \frac{1}{\mu^{(m)}}\right) = \frac{E[\sum_{i=1}^{\infty} q^{i-1} C_{i+m-2}^{i-1} W_i f(x_i)]}{E[\sum_{i=0}^{\infty} q^i C_{i+m-1}^i W_i f(x_i)]}. \tag{8}$$

Since, the coefficients C_{n+m}^n are binomial coefficients, they may be calculated using the recurrence $C_{i+m}^i = C_{i+m-1}^i + C_{i+m-1}^{i-1}$.

Monte Carlo Error

The Monte Carlo error obtained when computing a matrix-vector product is well known to be:

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

[3] The initial density vector $p = \{p_i\}_{i=1}^n$ is called *permissible* to the vector $h = \{h_i\}_{i=1}^n \in \mathbb{R}^n$, if $p_i > 0$ when $h_i \neq 0$ and $p_i = 0$ when $h_i = 0$. The transition density matrix $P = \{p_{ij}\}_{i,j=1}^n$ is called *permissible* to the matrix $A = \{a_{ij}\}_{i,j=1}^n$, if $p_{ij} > 0$ when $a_{ij} \neq 0$ and $p_{ij} = 0$ when $a_{ij} = 0$, $i, j = 1, \ldots, n$.

where $Var(\theta) = \{(E[\theta])^2 - E[\theta^2]\}$ and

$$E[\theta] = E[\frac{h_{k_0}}{p_{k_0}} W_i f_{k_i}] =$$

$$\sum_{k_0=1}^{n} \frac{h_{k_0}}{p_{k_0}} p_{k_0} \sum_{k_1=1}^{n} \cdots \sum_{k_i=1}^{n} \frac{a_{k_0 k_1} \ldots a_{k_{m-1} k_m}}{p_{k_0 k_1} \ldots p_{k_{m-1} k_m}} p_{k_0 k_1} \ldots p_{k_{m-1} k_m}$$

An Optimal Case If the row sums of A are a constant, a, i.e. $\sum_{j=1}^{n} a_{ij} = a$, and if all the elements of the vector f are constant, and if we furthermore define the initial and transition densities as follows: $i = 1, 2, \ldots n$ and $p_\alpha = \frac{|h_\alpha|}{\sum_{\alpha=1}^{n} |h_\alpha|}$; $p_{\alpha\beta} = \frac{|a_{\alpha\beta}|}{\sum_{\beta=1}^{n} |a_{\alpha\beta}|}$, $\alpha = 1, \ldots n$ (the case of using importance sampling), then $Var[\theta] = 0$.

Proof: Direct calculations gives us: $E[f\frac{h_{k_0}}{p_{k_0}} W_i] = (h, e)(-1)^j a^i f$, and $E[(f\frac{h_{k_0}}{p_{k_0}} W_i)^2] = (h, e)^2 a^{2i} f^2$, and so $Var[f\frac{h_{k_0}}{p_{k_0}} W_i] = 0$.

The Common Case

$Var[\theta] = (E[h_{k_0} W_m f_{k_m}])^2 - E[(h_{k_0} W_m f_{k_m})^2] \leq (E[h_{k_0} W_m f_{k_m}])^2 \leq \sum_{i=1}^{n} |a_{k_0 i}| . \sum_{i=1}^{n} |a_{k_1 i}| \ldots \sum_{i=1}^{n} |a_{k_{m-1} i}|$, for f and h - normalized.

Remark

We remark that in equation (7) the length of the Markov chain l is equal to the number of iterations in the power method. However in equation (8) the length of the Markov chain is equal to the number of terms in truncated series for the resolvent matrix. In this second case the parameter m corresponds to the number of iterations.

4 Quasirandom Sequences for Matrix Computations

Let us recall that power method-based iterations are built around computing $h^T A^i f$, see equations (4) and (5). We will try to turn these Markov chain computations into something interpretable as an integral. To do so, it is convenient to define the sets $G = [0, n)$ and $G_i = [i - 1, i)$, $i = 1, \ldots, n$, and likewise to define the piecewise continuous functions $f(x) = f_i$, $x \in G_i, i = 1, \ldots, n$, $a(x, y) = a_{ij}, x \in G_i, y \in G_j$, $i, j = 1, \ldots, n$ and $h(x) = h_i$, $x \in G_i, i = 1, \ldots, n$.

Because $h(x), a(x, y), f(x)$ are constant when $x \in G_i$, $y \in G_J$, we choose:

$$p(x) = p_i, \quad x \in G_i \quad (\sum_{i=1}^{n} p_i = 1),$$

$$p(x,y) = p_{ij}, \;\; x \in G_i, \;\; y \in G_j, \;\; (\sum_{j=1}^{n} p_{ij} = 1, i = 1, \ldots, n).$$

Now define a random (Markov chain) trajectory as

$$T_i = (y_0 \rightarrow y_1 \rightarrow \ldots \rightarrow y_i),$$

where y_0 is chosen from initial probability density $p(x)$, and the probability of choosing y_j given y_{j-1} is $p(y_{j-1}, y_j)$. The trajectory, T_i, can be interpreted as a point in the space $G \times \ldots \times G = G^{i+1}$ where the probability density of such a point is:

$$p_i(y_0, y_1, \ldots y_i) = p(y_0)p(y_0, y_1) \ldots p(y_{i-1}, y_i). \tag{9}$$

Let us know call $W_j^\star$ the continuous analog of W_j's in equation (6) giving is:

$$(h, A^i f) = E[W_i^\star f_{k_i}] =$$

$$\int_{G_0} \ldots \int_{G_i} p_i(y_0, y_1, \ldots y_i) h(y_0) a(y_0, y_1) \ldots a(y_{i-1}, y_i) f(y_i) dy_0 \ldots dy_i.$$

This expression lets us consider computing $h^T A^i f$ to be equivalent to computing an $(i+1)$-dimensional integral. This integral can be numerically approximated using QRNs and the error in this approximation can then be analyzed with Koksma-Hlawka-like (equation (2)) bounds for quasi-Monte Carlo numerical integration. We do not know A^i explicitly, but we do know A and can use the previously described Markov chain to produce a random walk on the elements of the matrix to approximate $h^T A^i f$.

Consider $h^T A^i f$ and an $(i+1)$-dimensional QRN sequence with star-discrepancy, D_N^*. Normalizing the elements of A with $\frac{1}{n}$, and the elements of h and f with $\frac{1}{\sqrt{n}}$ we have previously derived the following error bound (for a proof see [9]):

$$|h_N^T A_N^l f_N - \frac{1}{N} \sum_{s=1}^{N} h(x_s) a(x_s, y_s) \ldots a(z_s, w_s) f(w_s)| \leq |h|^T |A|^l |f| D_N^*. \tag{10}$$

If A is a general sparse matrix with d nonzero elements per row, and $d \ll n$, then the *importance sampling method* can be used. The normalizing factors in the error bound in equation (10) are then $1/d$ for the matrix, A, and $1/\sqrt{(}n)$ for the vectors, h and f.

5 Numerical Results

Why are we interested in studying MCMs for the eigenvalue problem? Because the computational complexity of MCMs for this 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 MCMs very efficient for large, sparse, eigenvalue problems, for which deterministic methods are not computationally efficient. Also, Monte Carlo algorithms have high parallel efficiency, i. e. the time to solution of a problem on p processors decreases by almost exactly p over the cost of the same computation on a single processor. In fact, in the case where a copy of the non-zero matrix elements of A is sent to each processor, the execution time for computing an extremal eigenvalue on p processors is bounded by $O(lN/p)$. This result assumes that the initial communication cost of distributing the matrix, and the final communication cost of collecting and averaging the distributed statistics is negligible compared to the cost of generating the Markov chains and forming the statistic, θ.

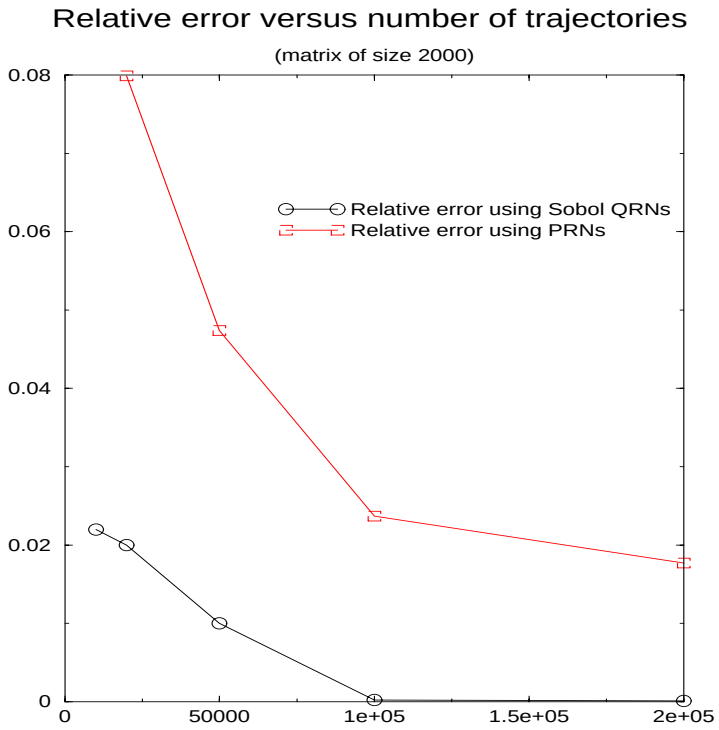

Fig. 3. Relative errors in computing the dominant eigenvalue for a sparse matrix of size 2000 $\times$ 2000. Markov chains realizations are produced using PRNs and Soboĺ QRNs.

Numerical tests were performed on general sparse matrices of size 128, 1024, 2000 using PRNs and Soboĺ, Halton and Faure quasirandom sequences. An improvement in both the magnitude of error and the convergence rate

Table 1. Monte Carlo estimates using PRNs and QRN sequences for computing the dominant eigenvalue of two matrices of size 128 and 2000 via the power method.

	PRN	$Faure$	$Sobol'$	$Halton$
$Est.$ $\lambda 128_{max}$	61.2851	63.0789	63.5916	65.1777
$Rel.$ $Error$	0.0424	0.0143	0.0063	0.0184
$Est.$ $\lambda 2000_{max}$	58.8838	62.7721	65.2831	65.377
$Rel.$ $Error$	0.0799	0.01918	0.0200	0.0215

were achieved using QRNs in place of PRNs. The results shown in *Table 1* were obtained using the power method. They show the results for computing λ_{max} using power method with both PRNs and different quasirandom sequences. For these examples the length of the Markov chain corresponds to the power of the matrix in the scalar product (the "power" in the power method).

Figure 3 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 $100,000$ or more PRNs are used. The fact that similar accuracy with these kinds of calculations can be achieved with QRNs at 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.

The results in *Figures 4* and *5* were obtained using the resolvent method (i. e. , the power method applied to the resolvent matrix, as described in §3). These results show the relative errors in computing λ_max for the same matrices of order 1024 and 2000. For the resolvent method the length of the Markov chain corresponds to the truncation number in the series that presents resolvent matrix. In both these figures, the errors when using QRNs are significantly smaller than those obtained when using PRNs. In addition, the error using PRNs grows significantly with the length of the Markov chain. This is in sharp contrast to all three QRN curves, which appear to show that the error in these cases remains relatively constant with increasing Markov chain length.

In addition to convergence test, we also performed parallel computations to empirically examine the parallel efficiency of these quasi-Monte Carlo methods. The parallel numerical tests were performed on a Compaq Alpha parallel cluster with 8 DS10 processors each running at 466 megahertz using MPI to provide the parallel calls. Each processor executes the same program

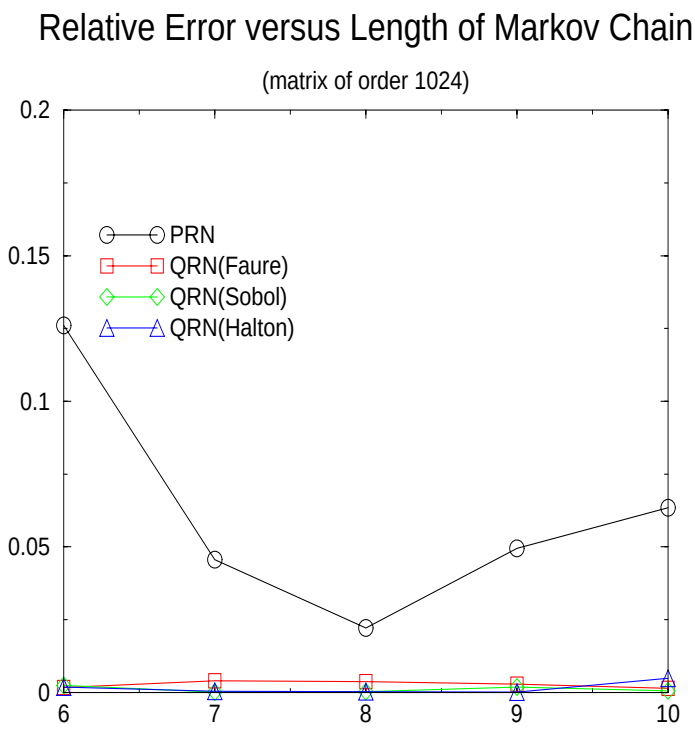

Fig. 4. Relative errors in computing λ_{max} using different length Markov chains for a sparse 1024 $\times$ 1024 matrix. The random walks are generating using PRNs, and Faure, Soboĺ and Halton QRN sequences.

for N/p trajectories (here p is the number of processors), and at the end of the trajectory computations, a designated host processor collects the results of all realizations and computes the desired average values. The results shown in *Table 2* that the high parallel efficiency of Monte Carlo methods is preserved with QRNs for this problem.

In these calculations we knew beforehand how many QRNs we would use in the entire calculation. Thus, we neatly broke the sequences into same-sized subsequences. Clearly, it is not expected that this kind of information will be known beforehand for all parallel quasi-Monte Carlo applications. In fact, an open and interesting problem remains that of providing extensible streams of QRNs that can extend calculations to a convergence determined termination. This remains a major challenge to the widespread use of QRNs in both parallel and serial computations.

These numerical experiments show that one can parallelize the quasi-Monte Carlo approach for the calculation of the extremal eigenvalue of a matrix. They also show that the parallel efficiency of the regular Monte Carlo approach to this problem is maintained by the quasi-Monte Carlo method. Fi-

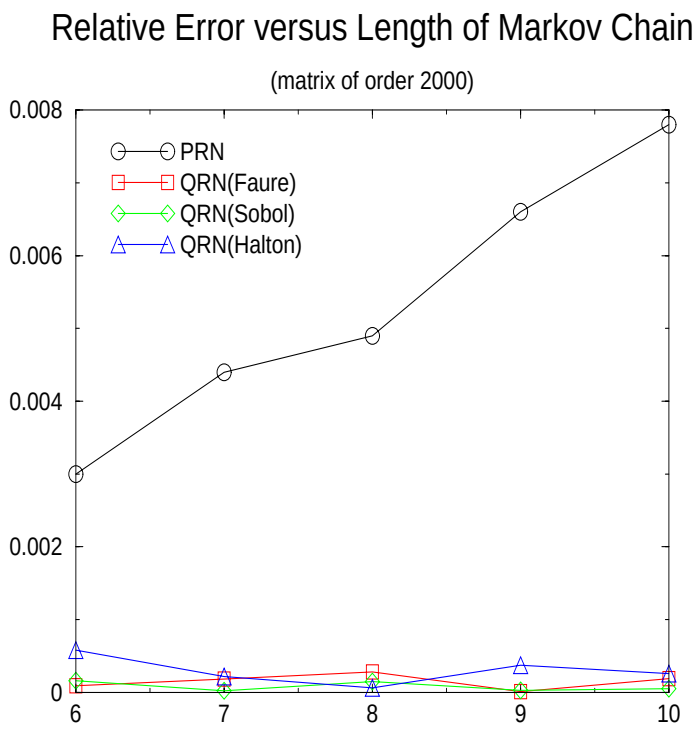

Fig. 5. Relative errors in computing λ_{max} using different length Markov chains for a sparse 2000 $\times$ 2000 matrix. The random walks are generating using PRNs, and Faure, Soboĺ and Halton QRN sequences.

nally, the most important fact is that the accelerated convergence of QRNs is seen for this Markov-chain based computation and is furthermore maintained in a parallel context.

6 Conclusions

Quasi-Monte Carlo methods and QRNs are powerful techniques for accelerating the convergence of ubiquitous MCMs. For computing the extremal eigenvalues of a matrix, it is possible to accelerate the convergence of well-known Monte Carlo methods with QRNs, and to take advantage the natural parallelism of MCMs. In fact, the execution time on p processors for computing the extreme eigenvalue of a matrix is bounded by $O(NT/p)$, (excluding the initial and final communication time) where N is the number of chains, and l is the average length of the Markov chains, both of which are independent of matrix size n. Therefore the quasi-Monte Carlo methods can be efficiently implemented on MIMD environment and in particular on a cluster of workstations under MPI.

Table 2. Parallel MPI implementation of the *power Monte Carlo algorithm* and the *power quasi-Monte Carlo algorithm* for calculating the dominant eigenvalue of a sparse 2000 × 2000 matrix using PRNs, and Sobol and Halton QRNs. The number of Markov chains realized was 100000, and the exact value of λ_{max} is 64.00).

	$1pr.$	$2pr.$	$3pr.$	$4pr.$	$5pr.$	$6pr.$	$7pr.$	$8pr.$
$\mathrm{MCM_{pseudo}}$								
Time(s)	19	9	6	4	3	3	2	2
Efficiency		1.06	1.06	1.18	1.2	1.06	1.3	1.18
$\lambda_{\max}$	62.48	61.76	63.76	61.3151	61.39	64.99	63.53	62.94
$\mathrm{QMC_{Sobol}}$								
Time(s)	20	9	6	5	4	3	2	2
Efficiency		1.1	1.1	1	1	1.1	1.5	1.1
$\lambda_{\max}$	64.01	64.01	64.01	64.01	64.01	64.01	64.01	64.01
$\mathrm{QMC_{Halton}}$								
Time(s)	18	9	6	4	3	3	2	2
Efficiency		1.1	1.1	1	1	1.1	1.5	1.1
$\lambda_{\max}$	63.92	63.92	63.92	63.92	63.92	63.92	63.92	63.92

The authors have also investigated the use of QRNs in Monte Carlo methods for the solution of elliptic partial differential equations, [8]. Some small improvement in the quasi-Monte Carlo approach was seen in this case over the standard Monte Carlo approach. The MCMs explored in that paper and the present paper are all Markov-chain based. In the future, we hope to begin a comprehensive investigation into the use of QRNs in Markov chain calculations.

Acknowledgements

This paper is based upon work supported by the North Atlantic Treaty Organization under a Grant awarded in 1999, and a U. S. Army Research Office contract awarded in 1999.

References

1. P. Bratley, B. L. Fox and H. Niederreiter, "Implementation and tests of low-discrepancy point sets," *ACM Trans. on Modeling and Comp. Simul.*, **2**: 195–213, 1992
2. R. E. Caflisch, "Monte Carlo and quasi-Monte Carlo methods," *Acta Numerica*, **7**: 1–49, 1998.
3. I. Dimov, A. Karaivanova, "Parallel computations of eigenvalues based on a Monte Carlo approach," *Journal of Monte Carlo Methods and Applications*, **Vol.4**, Num.1, pp.33–52, 1998.

4. H. Faure, "Discrépance de suites associées à un système de numération (en dimension s)," *Acta Arithmetica*, **XLI**: 337–351, 1992.
5. J. H. Halton, "On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals," *Numer. Math.*, **2**: 84–90, 1960.
6. J. M. Hammersley and D. C. Handscomb, *Monte Carlo Methods*, Methuen, London, 1964.
7. J. F. Koksma, "Een algemeene stelling uit de theorie der gelijkmatige verdeeling modulo 1," *Mathematica B (Zutphen)*, **11**: 7–11, 1942/43.
8. M. Mascagni, A. Karaivanova and Y. Li, "A Quasi-Monte Carlo Method for Elliptic Partial Differential Equations," Monte Carlo Methods and Applications, in the press, 2001.
9. M. Mascagni and A. Karaivanova, "Are Quasirandom Numbers Good for Anything Besides Integration?" in *Proceedings of Advances in Reactor Physics and Mathematics and Computation into the Next Millennium (PHYSOR2000)*, 2000.
10. M. Mascagni and A. Srinivasan, "Algorithm 806: SPRNG: A Scalable Library for Pseudorandom Number Generation," *ACM Transactions on Mathematical Software*, **26**: 436–461, 2000, and at the URL `http://www.sprng.cs.fsu.edu`.
11. B. Moskowitz and R. E. Caflisch, "Smootness and dimension reduction in quasi-Monte Carlo methods", *J. Math. Comput. Modeling*, **23**: 37–54, 1996.
12. H. Niederreiter, *Random number generation and quasi-Monte Carlo methods*, SIAM: Philadelphia, 1992.
13. K. F. Roth, "On irregularities of distribution," *Mathematika*, **1**: 73–79, 1954.
14. I. M. Soboĺ, "The distribution of points in a cube and approximate evaluation of integrals," *Zh. Vychisl. Mat. Mat. Fiz.*, **7**: 784–802, 1967.