What Are Quasirandom Numbers And Are They Good For Anything Besides Integration?

Michael Mascagni* Aneta Karaivanova†

January 5, 2000

Abstract

Monte Carlo methods are very general methods but suffer from the slow, $O(N^{-1/2})$, convergence characteristic of random sampling. In recent years, the convergence of the Monte Carlo method for numerical integration has been improved by replacing **p**seudo**r**andom **n**umbers (PRNs) with more uniformly distributed numbers known as **q**uasi**r**andom **n**umbers (QRNs). However, the advantages for numerical integration hold for integrals over the unit cube depend on many factors, such as the dimension and smoothness of the integrand, and whether or not the support of the integrand coincides with the unit cube. In the best cases, convergence for numerical integration becomes almost $O(N^{-1})$, while in the worst cases convergence falls back to that obtained with PRNs. In this paper we will discuss the main motivation for QRNs, and give a brief overview of some methods of generating QRNs. Most importantly, we begin to consider whether Monte Carlo applications besides integration can be accelerated with QRNs. To this end we propose and study the applicability of QRNs for evaluating the largest and the smallest eigenvalue of a given matrix. This application makes use of QRNs in the evaluation of discrete Markov chains. In many areas of Monte Carlo transport, the evaluation of Markov chains is likewise undertaken. We believe that our success with this Markov chain example motivates further work in the application of QRNs to accelerate convergence of problems in transport Monte Carlo.

The proposed quasi-Monte Carlo method based on Markov chain evaluation is a method for the computation of the extreme eigenvalues of a matrix. This method:

- can be considered as modification of the *Power Method* but it avoids the LU-decomposition and the forward and backward solving of the factored system at every iteration step in the Inverse Power Method and Inverse Shifted Power Method. The resolvent matrix is presented as a series. This representation permits us to use the well-known random walk process on the elements of the matrix to compute a matrix vector product with higher powers of the given matrix. This in turns leads to a substantial reduction in the computational effort required for solution.
- improves the convergence rate of the corresponding Monte Carlo method. Numerical tests are performed on sparse matrices of size 128, 1024, 2000 using PRNs and Sobol', Halton and Faure quasirandom sequences.

*Department of Computer Science, Florida State University, 203 Love Building, Tallahassee, FL 32306-4530, **USA**, E-mail: mascagni@cs.fsu.edu, URL: http://www.cs.fsu.edu/~mascagni

†Department of Computer Science, Florida State University, 203 Love Building, Tallahassee, FL 32306-4530, **USA**, *AND* Bulgarian Academy of Sciences, Central Laboratory for Parallel Processing, Sofia, **Bulgaria**, E-mail: aneta@scri.fsu.edu

Introduction

Monte **C**arlo **m**ethods (MCMs) are based on the simulation of stochastic processes whose expected values are equal to computationally interesting quantities. MCMs offer simplicity of construction, and are often designed to mirror some process whose behavior is only understood in a statistical sense. However, there are a wide class of problems where MCMs are the only known computational methods of solution. 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. One generic approach to improving the convergence of MCMs has been the use of highly uniform random numbers in place of the usual PRNs. While PRNs are constructed to mimic the behavior of truly random numbers, these highly uniform numbers, called QRNs, are constructed to be as evenly distributed as is mathematically possible. Indeed, pseudorandom numbers are scrutinized via batteries of statistical tests that check for statistical independence in a great variety of ways. In addition, these tests check for uniformity of distribution, but not with excessively stringent requirements. Thus, one can think of computational random numbers as either those that possess considerable independence, the PRNs, or those that possess considerable uniformity, the QRNs.

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 term $\sum_{n=1}^N \chi_{[0,u)}(x_n)$ counts the number of x_n's in the interval $[0, u)$, and thus $\left| \frac{1}{N} \sum_{n=1}^N \chi_{[0,u)}(x_n) - u \right|$ measures the difference between the actual distribution of points in the interval $[0, u)$ and the uniform distribution on $[0, u)$. By taking the supremum we are characterizing the distribution of the $\{x_n\}_{n=1}^N$ through it's maximal deviation from uniformity.[1] The mathematical motivation for QRNs 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. Let us assume that we are interested in the numerical value of $I = \int_0^1 f(x)\, dx$, and we seek to optimize approximations of the form $I \approx \frac{1}{N} \sum_{n=1}^N f(x_n)$. A solution to the optimization of the integration nodes, $\{x_n\}_{n=1}^N$, comes from the famous Koksma-Hlawka inequality:

Theorem (Koksma-Hlawka, [13]): 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:

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

This simple bound on the integration error is a product of $V(f)$, the total variation of the integrand in the sense of Hardy and Krause, and D_N^*, the star discrepancy of the integration points. A major area of research in Monte Carlo is variance reduction, which indirectly deals with minimizing $V(f)$. Quasirandom number generation deals with minimization of the other factor.

[1] In one dimension the star discrepancy is exactly the infinity norm of the difference between the empirical cumulative distribution of the points and the exact cumulative uniform distribution. The fact that the infinity norm is used is based on the traditional way probabilists place a metric on a space of probability measures.

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 QRNs can be as low as N^{-1}.[2] 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, [19], 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, [11], Sobol', [21, 1], Faure, [7, 8], and Niederreiter, [17, 2].

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 QRNs. This fact was born out by careful work of Caflisch and his students Morokoff and Moskowitz. They studied the efficacy of QRNs to numerical integration, [3, 15, 16, 4], by carefully investigating the dimensionality of the integrand and its smoothness as it impacts convergence. In a nutshell, their results showed that at high dimensions, $s \approx> 40$, quasi-Monte Carlo integration ceases to be an improvement over regular Monte Carlo integration. Perhaps more startling was that they showed that a considerable fraction of the enhanced convergence is lost in quasi-Monte Carlo integration when the integrand is discontinuous. In fact, even in two dimensions one can lose the approximately $O(N^{-1})$ quasi-Monte Carlo convergence for an integrand that is discontinuous on a curve such as a circle. In the best cases the convergence drops to $O(N^{-2/3})$, which is only slightly better than regular Monte Carlo integration.

Methods of 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 developed by one of the authors. 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 Sobol' sequence, a well known quasirandom 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.

The first quasirandom number sequence was proposed by Halton, [11], and is based on the Van der Corput sequence with different prime bases 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 $j-1 = a_n \ldots a_0$ in base b, then $\phi_b(j-1) = 0.a_0 \ldots a_n$. 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.

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

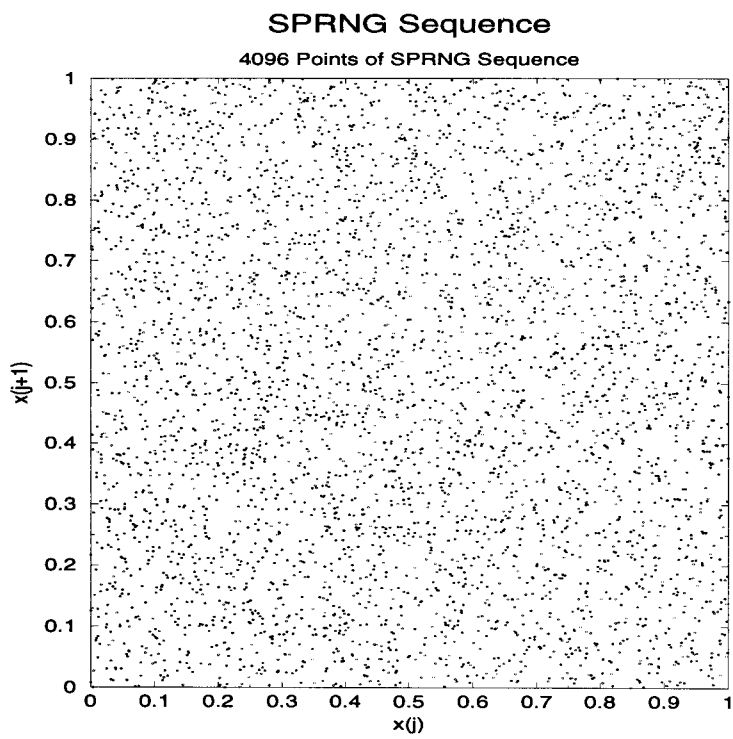

Figure 1: Tuples produced by successive elements from a SPRNG pseudorandom number generator.

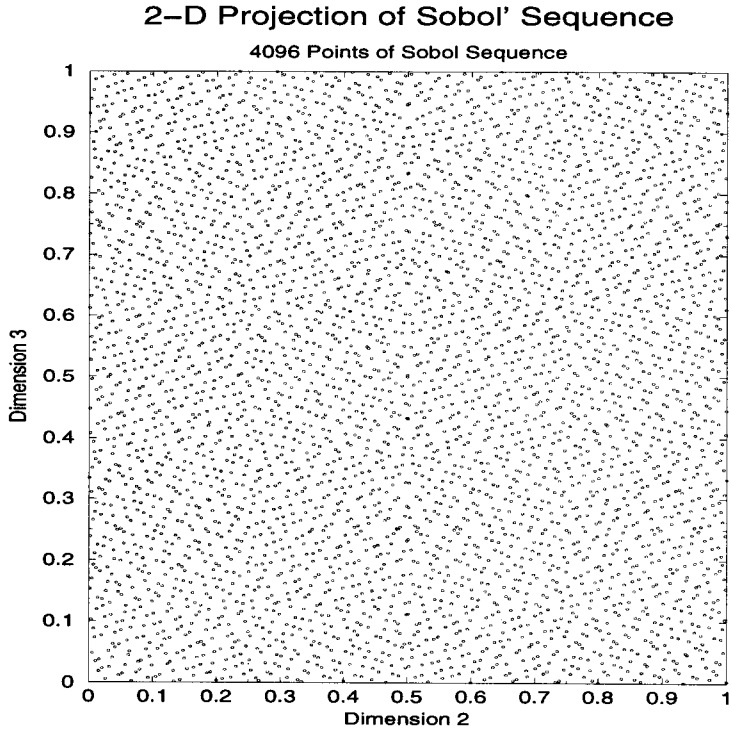

Figure 2: Tuples produced by the 2nd and 3rd dimension of the Soboĺ sequence.

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 Sobol' sequence, can be thought of in these terms. Sobol', [21], found a clever way to define more complicated direction numbers that the "unit vectors" which define the Van der Corput sequence. Besides producing very good quality QRNs, the reliance on direction numbers means that the Sobol' sequence is both easy to implement and very computationally efficient.

Since this initial work, Faure, Niederreiter and Sobol', [7], [18], [21], 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 $O(N^{-1}(\log N)^s)$, [18]. 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 quasirandom number generators. There are existing implementations of the Halton, Faure, Niederreiter and Sobol' sequences, [1], [2], [8], 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.

An extremely interesting alternative to the traditional methods for quasirandom number generation is based on a well-known defect of the most commonly used pseudorandom number generator: the linear congruential generator (LCG). Recall that an LCG has the following very simple form:

$$x_n = ax_{n-1} + b \pmod{m}. \tag{3}$$

When the multiplier, a, additive constant, b, and modulus, m, are chosen appropriately one obtains a purely periodic sequence with period as long as $Per(x_n) = 2^k$, when m is a power-of-two, and $Per(x_n) = m - 1$, when m is prime. It is well known that s-tuples made up from LCGs lie on lattices composed of a family of parallel hyperplanes, [14]. The most widely used measure of quality for an LCG is it's performance on the spectral test, [12]. The spectral test measures the maximum spacing between hyperplanes that cover the lattice produced by overlapping s-tuples from the LCG. The maximal hyperplane spacing that is produced by the spectral test, $\frac{1}{\nu_s(a,m)}$, is a geometric constant uniquely defined by the multiplier and modulus of an LCG and the dimension of the space. The spectral test is independent of the additive constant, and so it is commonly used to measure the quality of a given multiplier-modulus pair for quadrature-like applications in s dimensions. The s-dimensional discrepancy of a point set also measures that point set's suitability for use as integration nodes. Thus it should be no surprise that LCGs with good lattice properties should produce low-discrepancy point sets. More precisely, the method of "good lattice points" (GLP) attempts to create low-discrepancy point sets by constructing s-dimensional lattices, [20]. It has been shown that the GLP figure of merit is related to the spectral test, [6], and so LCGs with good spectral test results *are* GLP integration rules when considered over their full periods.

This relationship instantly leads to a way to produce low-discrepancy point sets based on full-period LCG tuples. Based on theoretical and implementational reasons, the moduli for LCGs are chosen either to be powers-of-two or primes. By choosing a prime-modulus, m, for an LCG to be used in this form of quasirandom number generation, one is implicitly choosing a point set of $m-1$ s-tuples. This is clearly very restrictive. On the other hand, if one picks $m = 2^k$ as the LCG

modulus, then considerably more flexibility is achieved in the size of the point sets available. Let us assume that a and b are chosen so that the $\{x_n\}$ produced by the LCG have the maximal period: $Per(x_n) = 2^k$. The conditions for this are that $a \equiv 1 \pmod 4$ and $b \equiv 1 \pmod 2$. Now consider the j least-significant bits of the $\{x_n\}$ by defining $y_n = x_n \pmod{2^j}$ with $1 \leq j \leq k$. It is easy to show that the $\{y_n\}$ can be produced directly by an LCG with $a = a \pmod{2^j}$, $b = b \pmod{2^j}$, and $m = 2^j$ in equation (3).

Since LCGs modulo a power-of-two have $Per(x_n) = 2^j$, the use of overlapping s-tuples of LCGs leads to quasirandom point sets of size 2^j. If one knows that approximately 2^j quasirandom points will be required in a given calculation, this is fine. However, it is usually the case that at most one knows only the error required in the computation, *a priori*. In such a situation having access only to quasirandom point sets with power-of-two sizes seems quite restrictive. In fact, properties of LCGs make this not nearly as problematic. Note that the conditions on a and b required to achieve the maximal period are properties only of the least significant bits of a and b. Thus one can choose a and b such that $a = a \pmod{2^j}$, $b = b \pmod{2^j}$ lead to $Per(x_n) = 2^j$ for all $j \geq 2$! Thus one can hope to find a single multiplier-additive constant pair that will provide good quality quasirandom point sets for all reasonable powers of two.

Quasi-Monte Carlo Methods for Eigenvalue Problem

We now consider the numerical evaluation of the eigenvalues of a matrix as our model problem for exploring the utility of QRNs in Markov chain problems. Suppose A is an $n \times n$ matrix with eigenvalues:

$$|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_n|.$$

Consider the problem of evaluating one or more eigenvalues of A, i.e. the values of λ for which

$$Au = \lambda u \tag{4}$$

holds.

Here we present quasi-Monte Carlo methods for evaluating the largest and the smallest (the extreme) eigenvalues of a given matrix based on well known MCMs for this problem. These methods can be considered as modifications of the well known *Power Method*, [9]. However, a major difference with the power method is that these MCMs avoid the LU-decomposition and the forward and backward solving of the factored system at every iteration step in the Inverse Power Method and Inverse Shifted Power Method. Instead, a given matrix and its resolvent are presented compactly as a series. This presentation permits us to use the well-known random walk process on the elements of the matrix to evaluate the eigenvalues. This leads to a great savings in the overall computation.

Markov Chains for Eigenvalue Algorithms

Consider a matrix $A = \{a_{ij}\}_{i,j=1}^n, A \in R^{n \times n}$, and vectors $f = (f_1, \ldots, f_n)^t \in R^n$ and $h = (h_1, \ldots, h_n)^t \in R^n$.

Consider the following Markov chain:

$$k_0 \rightarrow k_1 \rightarrow \ldots \rightarrow k_i, \tag{5}$$

where $k_j = 1, 2, \ldots, n$ for $j = 1, \ldots, i$ are natural numbers. The rules for constructing the chain (5) are:

$$Pr(k_0 = \alpha) = p_\alpha, \;\; Pr(k_j = \beta | k_{j-1} = \alpha) = p_{\alpha\beta}. \tag{6}$$

The vector $p = \{p_\alpha\}_{\alpha=1}^n$ is called the *initial density vector*, and the matrix $P = \{p_{\alpha\beta}\}_{\alpha\beta=1}^n$ is called the *transition density matrix*. In the usual MCM

$$p_\alpha = \frac{1}{n}, \;\; p_{\alpha\beta} = \frac{1}{n}.$$

The following choice

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

corresponds to *importance sampling* algorithms 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 the random variables W_j using the following recursion formula:

$$W_0 = \frac{h_{k_0}}{p_{k_0}}, \;\; W_j = W_{j-1} \frac{a_{k_{j-1}k_j}}{p_{k_{j-1}k_j}}, \;\; j = 1, \ldots, i. \tag{8}$$

Following [22], it is easy to show that

$$E\{W_i f_{k_i}\} = (h, A^i f), \;\; i = 1, 2, \ldots. \tag{9}$$

Extreme eigenvalues

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

$$[I - qA]^{-m} = \sum_{i=1}^{\infty} q^i C_{m+i-1}^i, \;\; |q\lambda| < 1 \tag{10}$$

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 $\mu = \frac{1}{1-q\lambda}$, 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:

$$\lambda_{max} = lim_{i \to \infty} \frac{(h, A^i f)}{(h, A^{i-1} f)}, \tag{11}$$

- or using the Power Method applied to the resolvent matrix:

$$\mu^{(m)} = \frac{([I - qK]^{-m} f, h)}{([I - qK]^{-(m-1)} f, h)} \underrightarrow{m \to \infty} \mu = \frac{1}{1 - q\lambda}, \;\; f \in \mathbb{R}^n, h \in \mathbb{R}^n. \tag{12}$$

for $q > 0$.

For computing the smallest eigenvalue we use the fact that for negative values of q, the largest eigenvalue, μ_{max}, of R_q corresponds to the smallest eigenvalue λ_{min} of the matrix A.

The Monte Carlo estimation for (11)) is (according to (8):

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

The Monte Carlo estimation for the second case (when we use the resolvent matrix) is based on the following theorem:

Theorem 1. *Let λ'_{max} be the largest eigenvalue of the matrix $A' = \{|a_{ij}|\}_{i,j=1}^{n}$. If q is chosen such that $|\lambda'_{max} < 1|$, then*

$$([I - qA]^{-m} f, h) = E\left\{\sum_{i=0}^{\infty} q^i C_{m+i-1}^i (A^i f, h)\right\}.$$

(The proof of this theorem can be found in [5].)

After some algebraic manipulations one can obtain

$$\lambda \approx \frac{1}{q}\left(1 - \frac{1}{\mu^{(m)}}\right) = \frac{(A[I - qA]^{-m} f, h)}{([I - qA]^{-m} f, h)} =$$

$$\frac{E\sum_{i=1}^{\infty} q^{i-1} C_{i+m-2}^{i-1} W_i f(x_i)}{E\sum_{i=0}^{\infty} q^i C_{i+m-1}^i W_i f(x_i)} = \frac{E\sum_{i=0}^{l} q^i C_{i+m-1}^i W_{i+1} f(x_i)}{E\sum_{n=0}^{l} q^i C_{i+m-1}^i W_i f(x_i)}, \tag{14}$$

where $W_0 = \frac{h_{k_0}}{p_{k_0}}$ and W_i are defined by (8). The coefficients C_{n+m}^n are calculated using the relation:

$$C_{i+m}^i = C_{i+m-1}^i + C_{i+m-1}^{i-1}.$$

Remark

We remark that for (11) the length of the Markov chain l is equal to the number of iterations i in the Power Method. However in (12) the length of the Markov chain is equal to the number of terms in truncated Neumann series for the resolvent matrix. In this second case the parameter m corresponds to the number of the iterations.

QRN Sequences for Power method

Using the following procedure:

$$G = [0, n)$$

$$G_i = [i-1, i), \;\; i = 1, \ldots, n$$

$$f(x) = f_i, \;\; x \in G_i, i = 1, \ldots, n$$

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

$$h(x) = h_i, \;\; x \in G_i, i = 1, \ldots, n,$$

we can consider computing $h^T A^i u$ to be equivalent to computing a $(i+1)$-dimensional integral.

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

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

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

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

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

$$|h_N^T A_N f_N - \frac{1}{N}\sum_{s=1}^N h(x_s)a(x_s,y_s)f(y_s)| = |\sum_{i=1}^n \sum_{j=1}^n h_i a_{ij} f_j (v_{ij} - \frac{1}{N}[\#\text{ of points in } Box(ij)])| \leq$$

$$\sum_{i=1}^n \sum_{j=1}^n |h_i a_{ij} f_j||(v_{ij} - \frac{1}{N}[\#\text{ of points in } Box(ij)])| \leq \sum_{i=1}^n \sum_{j=1}^n |h_i a_{ij} f_j| D_N^* = |h|^T |A||f|.D_N^*,$$

where $|h| = \{|h_i|\}_{i=1}^n$, $A = \{|a_{ij}|\}_{i,j=1}^n$ and $|f| = \{|f_i|\}_{i=1}^n$.
Finaly, we have

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

Analogously, considering $h^T A^l f$ and $l+1$-dimensional sequence we have

$$|h_N^T A_N^l f_N - \frac{1}{N}\sum_{s=1}^N h(x_s)a(x_s,y_s)\dots a(z_s,w_s)f(w_s)| \leq |h|^T |A|^l |f|.D_N^*. \tag{16}$$

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

$$G = [0,1)$$

$$G_{ij} = [\frac{\sum_{k'=1}^{j-1} |a_{ik'}|}{\sum_{k'=1}^{d_i} |a_{ik'}|}, \frac{\sum_{k'=1}^{j} |a_{ik'}|}{\sum_{k'=1}^{d_i} |a_{ik}|}), \;\; i = 1,\dots,n, \;\; j = 1,\dots,d,$$

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

$$a(x,y) = a_{ij}, x \in [\frac{i-1}{n}, \frac{i}{n}), \;\; y \in G_{ij}, \;\; i = 1,\dots,n, \;\; j = 1,\dots,d.$$

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

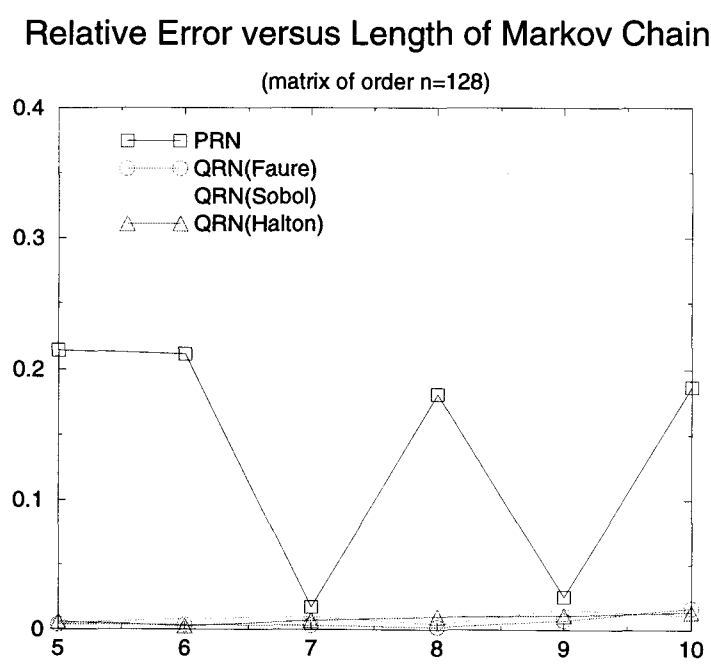

Figure 3: Relative errors in computing λ_{max} using different length of Markov chains for a sparse matrix 128 $\times$ 128. The random walks are realized using PRN and Faure, Sobol' and Halton sequences.

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

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

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 $\|A\| < 1$.

Some Numerical Results

When we replace PRNs with QRNs in this situation, so as to carefully control the process so we may rigorously analyze the convergence, we see an improvement in convergence. The computational complexity is $O(lT)$, where l is the length of the Markov chains and T is the number of the chains.

Numerical tests are performed on general sparse matrices of size 128, 1024, 2000 using PRNs and Sobol', Halton and Faure quasirandom sequences. The test matrices are sparse and stored in *packed row format* (i.e. only nonzero elements are kept). The estimated λ_{max} and the corresponding relative errors are presented on *Tables 1, 2 and 3.* The exact value of λ_{max} for all test matrices is 64.0000153. The results show improvement of the accuracy. Numerical experiments using Resolvent MC method have been also performed - the relative errors in computing λ_{max} using Markov chains with different length are presented on *Figures 3, 4, and 5.*

The relative errors in computing $h^T A^k f$ with A a sparse matrix of order 2000, are presented in *Figure 4.* The number of Markov chains, N, is 20000 and the length of the chains is equal to k. The

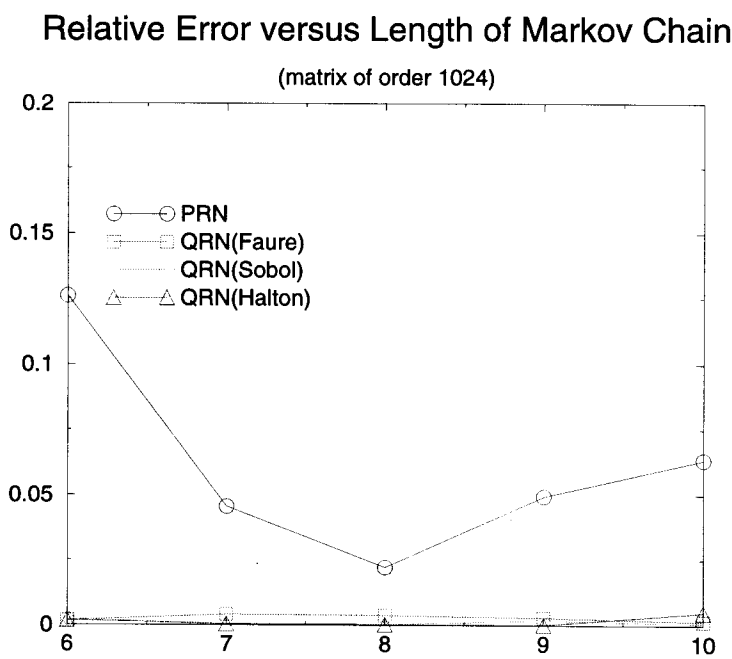

Figure 4: Relative errors in computing λ_{max} using different length of Markov chains for a sparse matrix 1024×1024. The random walks are realized using PRN, Faure, Sobol and Halton sequences.

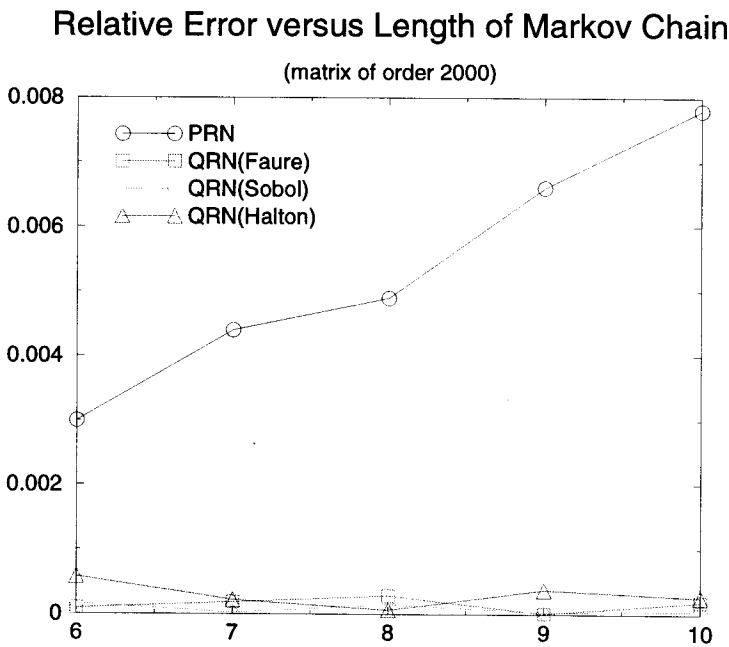

Figure 5: Relative errors in computing λ_{max} using different length of Markov chains for a sparse matrix 2000×2000. The random walks are realized using PRN, Faure, Sobol and Halton sequences.

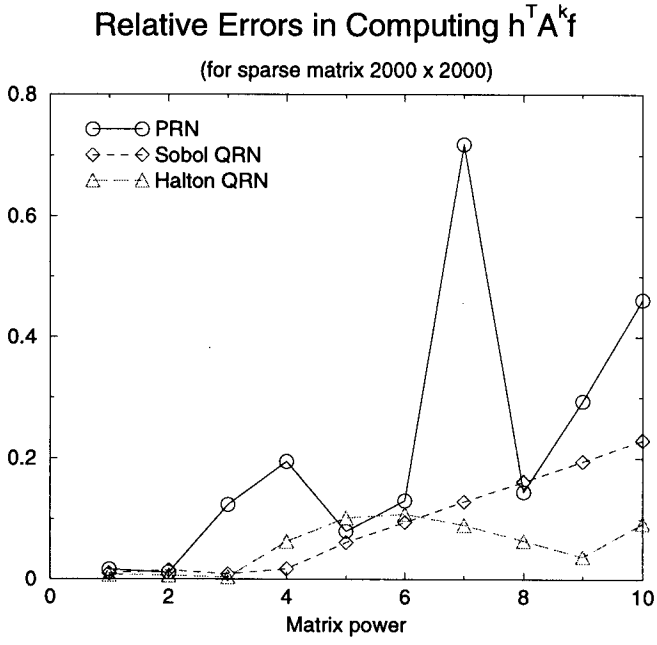

Figure 6: Relative errors in computing $h^T A^k f$ for $k = 1, 2, \ldots, 10$ for a sparse matrix 2000×2000. The corresponding Markov chains are realized using PRN, Sobol and Halton sequences.

Table 1: Results for a matrix 128×128 using Monte Carlo iterations with PRNs and QRNs. The length of the Markov chains is 5 and the number of chains is 1280.

	PRN	*QRN(Faure)*	*QRN(Sobol)*	*QRN(Halton)*
Estimated λ_{max}	61.2851	63.0789	63.5916	65.1777
Relative Error	0.0424	0.0143	0.0063	0.0184

results confirm that the Sobol and Halton quasirandom sequences produce higher precision results than similar length pseudorandom sequences. The more important fact is the smoothness of the quasirandom "iterations" with k. This is important because these eigenvalue algorithms compute a Raleigh quotient which requires the division of values from consecutive iterations. Another important feature of quasi-Monte Carlo methods is the increased smoothness of convergence as the number of samples increases. This property is not readily apparent on the figure, but will be verified in future work.

Conclusions and Future Work

In this short paper we have presented a very idiosyncratic overview of the current state of quasi-Monte Carlo methods. To date, QRNs have shown their greatest effectiveness in convergence acceleration with the evaluation of numerical integrals where the dimensionality of the integrand

Table 2: Results for a matrix 1024×1024 using Monte Carlo iterations with PRNs and QRNs. The length of the Markov chains is 6 and the number of chains is 1280.

	PRN	*QRN(Faure)*	*QRN(Sobol)*	*QRN(Halton)*
Estimated λ_{max}	67.0243	57.7299	65.9630	61.6351
Relative Error	0.0472	0.0979	0.0306	0.0369

Table 3: Results for a matrix 2000×2000 using Monte Carlo iterations with PRNs and QRNs. The length of the Markov chains is 8 and the number of chains is 20000.

	PRN	*QRN(Faure)*	*QRN(Sobol)*	*QRN(Halton)*
Estimated λ_{max}	58.8838	62.7721	65.2831	65.377
Relative Error	0.0799	0.01918	0.0200	0.0215

was not too big, and the integrand was smooth. There are clearly many situations where MCMs can be recast into such a form, but in many areas, such as transport Monte Carlo, the natural mathematical setting for the problem is a Markov chain. In this paper we have taken a simple MCM involving Markov chains and have shown that in this example QRNs can, in fact, accelerate convergence. Moreover this examples was consistent with other quasi-Monte Carlo applications where QRNs are known to provide results that have much smaller extreme excursions than results with PRNs. The results in this paper, though preliminary in nature, are compelling, and should be used by the reader to motivate their interest in the application of QRNs to more problems involving both discrete and continuous Markov chains. We hope that in the future we will publish a more extensive analysis of the effectiveness of QRNs for this problem. In addition, we hope to have more results that show the effectiveness of QRNs in accelerating the convergence of MCMs based on Markov chains. More importantly, we also hope to provide analysis of the many different ways that QRNs may be applied to particular Markov chain problems in order to establish both the rigorous basis as well as the intuition behind the best practices of applying QRNs to Markov chain problems.

References

[1] P. Bratley and B. L. Fox, "Algorithm 659; Implementing Sobol's quasirandom sequence generator," *ACM Trans. on Math. Software*, **14**: 88–100, 1988.

[2] 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

[3] R. E. Caflisch, "Monte Carlo and quasi-Monte Carlo methods," *Acta Numerica*, **7**: 1–49, 1998.

[4] R. E. CAFLISCH, W. MOROKOFF AND A. B. OWEN, "Valuation of mortgage-backed securities using Brownian bridges to reduce effective dimension," *J. Comput. Finance*, **1**: 27–46, 1997.

[5] I. DIMOVAND A. KARAIVANOVA, "Parallel computations of eigenvalues based on a Monte Carlo approach," *Journal of MC Methods and Appl.*, **Vol.4**, Num.1, pp.33–52, 1998.

[6] K. ENTACHER, P. HELLEKALEK AND P. L'ECUYER, "Quasi-Monte Carlo integration with linear congruential generators," presented at *Third International Conference on Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, and personal communication, 1998.

[7] H. FAURE, "Discrépance de suites associées à un système de numération (en dimension s)," *Acta Arithmetica*, **XLI**: 337–351, 1992.

[8] B. L. FOX, "Algorithm 647; Implementation and relative efficiency of quasirandom sequence generators," *ACM Trans. on Math. Software*, **12**: 362–376, 1986.

[9] G. H. GOLUB AND C. F. VAN LOAN, *"Matrix computations"*, The Johns Hopkins Univ. Press, Baltimore, 1996.

[10] J. H. HALTON, "Sequential Monte Carlo Techniques for the Solution of Linear Systems", *TR 92-033*, University of North Carolina at Chapel Hill, Department of Computer Science, 46 pp., 1992.

[11] J. H. HALTON, "On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals," *Numer. Math.*, **2**: 84–90, 1960.

[12] D. E. KNUTH, *The Art of Computer Programming: Volume 2, Seminumerical Algorithms*, Third Edition, Addison-Wesley: Reading, MA, 1998.

[13] J. F. KOKSMA, "Een algemeene stelling uit de theorie der gelijkmatige verdeeling modulo 1," *Mathematica B (Zutphen)*, **11**: 7–11, 1942/43.

[14] G. MARSAGLIA, "Random numbers fall mainly in the planes," *Proc. Nat. Acad. Sci. U.S.A.*, **62**: 25–28, 1968.

[15] W. MOROKOFF AND R. E. CAFLISCH, "Quasi-Monte Carlo integration," *SIAM J. Comput. Phys.*, **122**: 218–231, 1995.

[16] B. MOSKOWITZ AND R. E. CAFLISCH, "Smoothness and dimension reduction in quasi-Monte Carlo methods", *J. Math. Comput. Modeling*, **23**: 37–54, 1996.

[17] H. NIEDERREITER, *Random number generation and quasi-Monte Carlo methods*, SIAM: Philadelphia, 1992.

[18] H. NIEDERREITER, "Low-discrepancy and low-dispersion sequences," *J. Number Theory*, **30**: 51–70, 1988.

[19] K. F. ROTH, "On irregularities of distribution," *Mathematika*, **1**: 73–79, 1954.

[20] I. H. SLOAN AND S. JOE, *Lattice Methods for Multiple Integration*, Oxford University Press: New York, 1994.

[21] I. M. SOBOĹ, "The distribution of points in a cube and approximate evaluation of integrals," *Zh. Vychisl. Mat. Mat. Fiz.*, **7**: 784–802, 1967.

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