The Stata Journal

The *Stata Journal*, electronic version (ISSN 1536-8734) is a publication of Stata Press, and Stata is a registered trademark of StataCorp LP.

The Stata Journal (2006)
6, Number 2, pp. 214–228

Generating Halton sequences using Mata

David M. Drukker
StataCorp
College Station, TX
ddrukker@stata.com

Richard Gates
StataCorp
College Station, TX
rgates@stata.com

Abstract. This paper discusses the advantages of Halton sequences over pseudorandom uniform numbers when using simulation to approximate integrals numerically. We describe two types of sequences and give Mata examples. Finally, we document the Mata function `halton()`, currently in release 9.1 of Stata, which computes both a Halton sequence and its Hammersley variant. Options to use these point sets are available in the Stata 9 program `asmprobit`, a multinomial-probit estimator, and in the Stata 9.1 Mata function `ghk()`, the Geweke–Hajivassiliou–Keane multivariate-normal simulator.

Keywords: st0103, halton(), Halton set, Hammersley set, quasirandom numbers

1 Introduction

With increasing numerical computing power, there is a corresponding increasing interest in using computer-intensive simulation techniques to compute approximations to likelihood functions that involve multidimensional integrals that do not have a closed-form solution. Evaluating these likelihood functions using simulation, also known as Monte Carlo integration, involves taking pseudorandom draws from the standard uniform distribution. In applying simulation techniques to numerical integration, however, the randomness of these draws is not as important as their uniform coverage over the domain of integration. The more uniform the coverage over the domain of integration, the better the numerical approximation. The coverage over the domain of integration can be dramatically improved by using quasirandom points.[1] When a quasirandom sequence is used in one of these simulation algorithms, the numerical integration technique is then referred to as quasi–Monte Carlo integration. This paper discusses the currently popular Halton point sets and their Hammersley variant.

The Stata 9 program [R] **asmprobit** that estimates the parameters of a multinomial probit model uses Halton point sets and the Geweke–Hajivassiliou–Keane (GHK) simulator to approximate the multivariate-normal integral. This paper provides some insight into why the Hammersley point sets are the default in `asmprobit` and why you may want to use pseudorandom point sets for multinomial probit models with more than 10 alternatives. In the 9.1 release of Stata, we provide two Mata functions that are used by `asmprobit`: `halton()`, a function that generates Halton or Hammersley point sets, and `ghk()`, a function that carries out the GHK simulator and which optionally uses Halton, Hammersley, or pseudorandom point sets.

1. Pseudorandom point sets pass no-correlation tests, whereas quasirandom point sets are correlated by design.

In section 2, we review Monte Carlo integration and introduce discrepancy as a measure of uniform coverage of a set of points over the unit interval. In section 3, we discuss the theory behind Halton sequences, their Hammersley variant, and using Mata to compute the point sets.[2] Section 4 gives the syntax of the Mata `halton()` function. In section 5, we demonstrate these point sets for some specific scenarios and illustrate their coverage over the unit interval graphically and numerically.

2 Monte Carlo integration

Assume that we have a multivariate function, $g(\mathbf{x})$, with domain $\mathbf{x} \in R^d \subseteq \Re^d$, a subset of the d dimensional reals. To perform Monte Carlo integration of $g(\mathbf{x})$, the domain of integration must be transformed to the unit interval of dimension d, $\overline{C}^d = [0,1]^d$, so that $\int_{R^d} g(\mathbf{x})\ d\mathbf{x} = \int_{\overline{C}^d} f(\mathbf{y})\ d\mathbf{y}$. This transformation allows us to approximate $\int_{\overline{C}^d} f(\mathbf{y})\ d\mathbf{y}$ by $(1/n)\sum_{i=1}^n f(\mathbf{u}_i)$, where the $\{\mathbf{u}_i \in C^d, i = 1, \ldots, n\}$, $C^d = [0,1)^d$, are uniform random vectors. We use the point set $\{\mathbf{u}_i \in C^d, i = 1, \ldots, n\}$ that minimizes the approximation error, $\left|(1/n)\sum_{i=1}^n f(\mathbf{u}_i) - \int_{\overline{C}^d} f(\mathbf{y})\ d\mathbf{y}\right|$.[3] Bounds on the error can be expressed, in part, by using the notion of the discrepancy of the $\mathbf{u}_i$'s over C^d. We will introduce this concept next.

2.1 Discrepancy

There are many ways of measuring discrepancy, or the lack of uniformity of a point set over a domain. Here we will describe discrepancy, using one type of discrepancy measure (termed the star discrepancy by Niederreiter [1992, definition 2.1]), as applied in the error analysis of quasi–Monte Carlo integration. For other discrepancy measures, see Niederreiter (1992) and Fang and Wang (1994). First, we will define the discrepancy for a one-dimensional point set and then generalize to a d-dimensional point set.

Assume that we have a set of points $\chi_n = \{x_i \in [0,1), i = 1, \ldots, n\}$. This point set is of dimension 1 since each x_i is a scalar. For any value $u \in [0,1)$, let the function $N(u, \chi_n)$ be the number of x_i's less than or equal to u. The discrepancy can be expressed as $D(n, \chi_n) = \sup_{u\in[0,1)} |\{N(u, \chi_n)/n\} - u|$. "sup" stands for supremum, which is the least upper bound. In other words, the discrepancy is the largest possible distance between a $u \in [0,1)$ and the average number of x_i's that are less than or equal to u. The more uniformly the x's are scattered, the closer $N(u, \chi_n)/n$ will be to u for any $u \in [0,1)$, thereby minimizing the discrepancy. Fang and Wang (1994) (and Niederreiter [1992]) show that the sequence $(2i-1)/2n, i = 1, \ldots, n$ has the lowest discrepancy, $1/2n$, on $[0,1)$. You will see in the next section that this sequence is used in the first dimension of the Hammersley set.

2. Sequences are the functions used to generate point sets. We concentrate on the point sets to make the discussion more accessible.

3. Actually, we would like to find the function (sequence) that minimizes the approximation error, but concentrating on point sets eases the discussion.

For point sets of dimension $d \geq 2$, we now have $\mathbf{u} \in C^d$ and $\chi_n^d = \{\mathbf{x}_i \in C^d, i = 1, \ldots, n\}$. We introduce the function $\nu(\mathbf{u}) = \prod_{i=1}^{d} u_i$, which is the volume of $[\mathbf{0}, \mathbf{u})$. The discrepancy is then expressed as

$$D(n, \chi_n^d) = \sup_{u \in C^d} \left| \frac{N(\mathbf{u}, \chi_n^d)}{n} - \nu(\mathbf{u}) \right|$$

As pointed out by Fang and Wang (1994), determining a sequence that has a minimal discrepancy when $d > 1$ is difficult. As a result, sequences are found that have an asymptotically (as n gets large) small discrepancy.

A bound on the error of the numerical approximation to the multivariate integral can be expressed as

$$\left| \frac{1}{n} \sum_{i=1}^{n} f(\mathbf{x}_i) - \int_{\overline{C}^d} f(\mathbf{y}) d\mathbf{y} \right| \leq V(f) D(n, \chi_n^d)$$

(Niederreiter [1992, theorem 2.11]), where $V(f)$ is the total variation of the function f as defined by Niederreiter (1992, 19). For a given f, the error bound depends on n and the sequence generating the points χ_n^d. We choose the sequence χ_n^d that has the smallest discrepancy for a given n and d. For all the sequences we consider, the discrepancy is decreasing in n. (This discrepancy is intuitive; more points should provide a better approximation.) Here we consider Halton, Hammersley, and uniform pseudorandom sequences for generating point sets. For feasible n, Halton and Hammersley sequences produce lower discrepancies for small d, but the uniform pseudorandom sequences produce smaller discrepancies for larger dimensions. As discussed below, for realistic n the crossing point appears to be around $d = 10$.

3 Algorithms for generating Halton point sets

Halton and Hammersley sequences use number-theoretic methods to ensure a high degree of uniformity in the point sets. Here we describe how the point sets are generated and provide a brief introduction to their number-theoretic foundation.

Any nonnegative integer, i, has a digit expansion $i = \sum_{j=1}^{q} b_{j,p}(i) p^{j-1}$, where $p^{q-1} \leq i < p^q$, base $p > 0$, and $b_{j,p}(i) \in \{0, 1, 2, \ldots, p-1\}$ is the jth digit of the base p representation of the base 10 number i. We will restrict p to be a prime number. For example, 113 in base 5 is 33 in base 10. Thus for $p = 5$ and $i = 33$, we have $q = 3$, $b_{1,5}(33) = 3$, $b_{2,5}(33) = 1$, and $b_{3,5}(33) = 1$, or $33 = 3 \cdot 5^0 + 1 \cdot 5^1 + 1 \cdot 5^2$. Henceforth, we will subscript the number with its base if the base is other than 10, for instance, $33 = 311_5$.

The ith member of a Halton sequence is obtained by computing the radical inverse of i with base p,[4]

$$r_p(i) = \sum_{j=1}^{q} \frac{b_{j,p}(i)}{p^j} \tag{1}$$

For our example, $r_5(33) = 3/5 + 1/25 + 1/125$. The Halton set of dimension d is generated from the first d primes, p_k, $k = 1, \ldots, d$, so that on draw i, $\mathbf{h}_i = \{r_{p_1}(i), r_{p_2}(i), \ldots, r_{p_d}(i)\}$. The Hammersley set uses a sequence of n evenly spaced points on $(0, 1)$ in the first dimension with the first $d-1$ Halton sequences in dimensions $2, \ldots, d$,

$$\widetilde{\mathbf{h}}_i = \left[\left\{\frac{2i-1}{2n}\right\}, r_{p_1}(i), r_{p_2}(i), \ldots, r_{p_{d-1}}(i)\right], \quad i = 1, \ldots, n$$

where $\{x\}$ is the fractional part of x (e.g., $0.2 = \{1.2\}$).[5] The latter can be used when evaluating a likelihood that requires m integral evaluations (m contributions to the likelihood) using Hammersley sets of length n, where the first index of each set is $i = 1, n+1, 2n+1, \ldots, (m-1)n+1$.

3.1 Halton set discrepancy

Let $\mathcal{U}_n^d$ be a point set from the uniform distribution of length n and dimension d; then a bound on the set's discrepancy can be expressed as

$$D(n, \mathcal{U}_n^d) \leq K\sqrt{\frac{\log\log n}{n}}$$

for some $K > 0$. The bound is independent of d. On the other hand, the bounds for the Halton and Hammersley point sets, $\mathcal{H}_n^d$ and $\widetilde{\mathcal{H}}_n^d$, respectively, are the following functions of d and n

$$D(n, \mathcal{H}_n^d) \leq \frac{C_{\mathcal{H}}(d)(\log n)^d}{n}$$

and

$$D(n, \widetilde{\mathcal{H}}_n^d) \leq \frac{C_{\widetilde{\mathcal{H}}}(d)(\log n)^{d-1}}{n}$$

(Fang and Wang 1994). Here the $C_{\mathcal{H}}(d)$ and $C_{\widetilde{\mathcal{H}}}(d)$ increase exponentially with d (Niederreiter 1992, chap. 4). Thus the Halton and Hammersley sets used in quasi–Monte Carlo integration are useful only for low-dimensional problems.

For moderate n, say, $n \in [50,\ 5{,}000]$, the advantages of the Halton and Hammersley point sets relative to the uniform point sets are tied up in the unknown constants of these bounds. We will illustrate this point by simulating the discrepancy for specific cases in section 5.

The practical implication is that although Halton and Hammersley points will provide more accurate approximations to the integral for small d, for larger d pseudouniform point sets should be used.

4. See Niederreiter (1992, 24) for a discussion of this radical inverse function.

5. This equation generalizes the definition in Fang and Wang (1994), which assumes that $\frac{2i-1}{2n} < 1$.

3.2 Computing Halton point sets

In our work in maximum simulated likelihood, we have seen two numerical techniques for computing Halton point sets. The first comes directly from the Halton sequence definition in (1) and the second, a result from Wang and Hickernell (2000) and ado-code posted on Statalist (see http://www.stata.com/statalist/archive/2004-08/msg00222.html), cleverly computes the ith value in a sequence from the $i-1$.

Algorithmically, we express (1) in Mata code. We first choose an index $i = 1, 2, \ldots$ in the sequence and assume that we are working with dimension $1 \leq k \leq d$, where d is the dimension of the integration. We choose the base p_k from the set of increasing primes $p_k \in \{2, 3, 5, 7, 11, \ldots\}$, to use an optimal set of bases for generating a Halton set. Then the following snippet of Mata code will iterate q times, where $p_k^{q-1} \leq i < p_k^q$, and on completion r_i will be the ith value from the Halton sequence for dimension k.

```
pk = p[k]
pj = 1
ri = 0
j = i
while (j>0) {
   pj = pj/pk
   bj = mod(j,pk)
   ri = ri + pj*bj
   j = trunc((j-bj)/pk)
}
```

Returning to the previous example where $k = 3$ and $p_3 = 5$, the variable `i` is 33, `pk` is the prime 5, and `bj` stores the constants $b_{j,5}(33)$, $j = 1, 2, 3$, that are computed in the `while` loop: $b_{1,5}(33) = 3 = 33 \bmod 5$ and $b_{2,5}(33) = 1 = (33-3)/5 \bmod 5$, and $b_{3,5}(33) = 1 = (6-1)/5 \bmod 5$, where $6 = (33-3)/5$.

The second algorithm for generating a Halton sequence cleverly generates the radical inverse of i from that of $i-1$. Before showing the general form of this relationship, we illustrate how this can be done by continuing the previous example. We left off with $i = 34$ using $p = 5$. The coefficients of the base 5 expansion are $b_{1,5}(34) = 4$, $b_{2,5}(34) = 1$, and $b_{3,5}(34) = 1$. Its radical inverse is $r_5(34) = 4/5 + 1/25 + 1/125 = 0.848 = 0.411_5$. We can obtain $r_5(34)$ from $r_5(33)$ by $r_5(34) = r_5(33) + 1/5$, or $0.311_5 + 0.1_5 = 0.411_5$.

An increment of $.1_p$ can be used until the sum exceeds 1 in base 10, when we must make a rightward carry operation. Wang and Hickernell (2000) define a rightward carry addition operator, $\oplus$. Let $b_{q+1,p}(i) = 0$ and
$k = \text{argmin}_j\{\, b_{j,p}(i) < p-1\}$, such that $1 \leq j \leq q+1$, and then

$$r_p(i+1) = r_p(i) \oplus \frac{1}{p} \equiv \frac{1 + b_{k,p}(i)}{p^k} + \sum_{j=k+1}^{q} \frac{b_{j,p}(i)}{p^j} \tag{2}$$

For example, when $i = 33 = 113_5$, so $k = 1$ and $b_{1,5}(33) = 3$. Then (2) yields $r_5(34) = (1 + 3)/5 + 1/25 + 1/125$. For $i = 34 = 114_5$, $k = 2$, and $b_{2,5}(34) = 1$, yielding $r_5(35) = (1 + 1)/25 + 1/125$. The exceptional case, $k = q + 1$, occurs when $i + 1 = p^q$. For example, at $i = 124 = 444_5$, $k = 4$ and since we defined $b_{4,5}(124)$ to be 0, $r_5(125) = (1 + 0)/5^4 = 1/625 = 0.0001_5$.

The way to compute $r_p(i+1)$ from $r_p(i)$ numerically comes from Wang and Hickernell (2000), who note that (2) can be expressed as

$$r_p(i + 1) = r_p(i) + \frac{1 + p}{p^k} - 1 \tag{3}$$

with k such that $\dfrac{1}{p^k} \leq 1 - r_p(i) < \dfrac{1}{p^{k-1}}$

Below is the Mata code implementing (3). Here assume that `p` is a vector containing the first 20 primes. We will use the kth prime as the base. The scalar `ri` contains the Halton number for index i before the code is executed and is updated to contain the $i + 1$th Halton number of the sequence.

```
pk = p[k]
pj = 1
x = 1 - ri - eps
while (pj > x) pj = pj/pk
ri = ri+(pk+1)*pj-1
```

Here we decrement $1-r_p(i) \equiv$ `1 - ri` by a small amount, `eps = epsilon(100)`, say, to prevent machine imprecision from terminating the loop too soon (see [M-5] **epsilon( )**).

Both approaches to computing Halton point sets can be used: the first approach is useful for implementing *start*, where the user specifies the starting index of the sequence; and the second algorithm is useful after the first point of the sequence is computed. Moreover, the *start* option could be useful for producing Halton point sets sequentially in blocks, where each block can pick up where the last block left off. For example, the likelihood evaluator for `asmprobit` loops over cases computing the simulated probability of each choice given the regression coefficient estimates. The Halton (or Hammersley) point set used in the simulated probability for each case picks up from the last index used in the simulated probability of the previous case.

Wang and Hickernell (2000) use (2) and (3) to randomize the Halton sequences. This randomization is done by generalizing the definition of the function $r_p(i)$ by defining $r_p(i, u)$ as $r_p(i + 1, u) = r_p(i, u) + \{(1 + p)/p^k\} - 1$ for $i = 0, \ldots$ and $r_p(0, u) = u$, where u is a draw from the uniform distribution, $\mathrm{U}(0, 1)$. For $u = 0$ the standard Halton sequence is generated. Another way to produce randomized Halton sequences is to shift the Halton sequence by adding u as described by Train (2003, 234), i.e., $\{r_p(i) + u\}$ (in Mata $\{x\}$ is implemented using `mod(x,1)`; see [M-5] **mod( )**). The two techniques will not, however, generate the same sequence.

These randomized Halton point sets can be used to estimate the error of the numerical integration by repeatedly estimating the integral, using different uniform variates as starting values for the sequence. A bound on the desired error of the numeric integration, ϵ, can be obtained using the variance of the integral estimates, σ^2, and the Chebyshev inequality, $\Pr(\epsilon < k\sigma) \geq 1 - 1/k^2$, where the constant k determines the confidence bound (e.g., $k = 5$ gives $\Pr = 0.96$).

We will not delve further into this topic, but for a good example of a Monte Carlo algorithm to compute multivariate normal probabilities that could use randomized Halton point sets, see Genz (1992). Moreover, Stata 9.1 provides the Mata function `ghalton()` to compute these generalized Halton point sets. We will next document this function as well as the Mata functions `halton()` and `_halton()`.

4 The Mata function halton()

4.1 Syntax

real matrix	`halton(`*real scalar n*`,` *real scalar d* [`,` *real scalar start* [`,` *real scalar hammersley*]]`)`
void	`_halton(`*real matrix x* [`,` *real scalar start* [`,` *real scalar hammersley*]]`)`
real colvector	`ghalton(`*real scalar n*`,` *real scalar base*`,` *real scalar u*`)`

4.2 Description

`halton(`n`,` d`)` returns an $n \times d$ matrix containing a Halton set of length n and dimension d.

`halton(`n`,` d`,` *start*`)` does the same thing, but the first row of the returned matrix contains the sequence starting at index *start*. The default is *start* $= 1$.

`halton(`n`,` d`,` *start*`,` *hammersley*`)`, with *hammersley* $\neq 0$, returns a Hammersley set of length n and dimension d with the first row of the returned matrix containing the sequences starting at index *start*.

`_halton(`x`)` modifies the $n \times d$ matrix so that it contains a Halton or Hammersley set of dimension d of length n.

`ghalton(`n`,` *base*`,` u`)` returns an $n \times 1$ vector containing a (generalized) Halton sequence using base *base* and starting from $u \in [0, 1)$. For $u = 0$, the standard Halton sequence is generated. If u is a uniform $(0, 1)$ variable, the randomized Halton sequence is generated.

4.3 Remarks

The Halton sequences are generated from the first d primes and generally have more uniform coverage over the unit cube of dimension d than that of sequences generated from pseudouniform random numbers. However, Halton sequences based on large primes ($d > 10$) can be highly correlated, and their coverage can be worse than that of the pseudorandom uniform sequences.

The Hammersley set contains the sequence $(2i-1)/(2n)$, $i = 1, \ldots, n$, in the first dimension and Halton sequences for dimensions $2, \ldots, d$. If *index* is given then the first dimension is `mod((2*i-1)/(2*n),1)`, `i` = *index*, ..., *index* + `n` −1.

`_halton()` modifies x and can be used when repeated calls are made to generate long sequences in blocks. Here update the *start* index between calls by using
start = *start* + `rows(`x`)`.

`ghalton()` uses the base *base*, preferably a prime, and generates a Halton sequence using $0 \leq u < 1$ as a starting value. If u is uniform $(0, 1)$, the sequence is a randomized Halton sequence. For $u = 0$, the sequence is the standard Halton sequence. Blocks of sequences can be generated by `ghalton()` by using the last value in the vector returned from a previous call as u. For example,

```
base = 5
x = J(n,1,0)
for (i=1; i<=k; i++) {
   x[.] = ghalton(n, base, x[n])
   ...
}
```

4.4 Conformability

`halton(`n`,` d`,` *start*`,` *hammersley*`)`:

input:

n:	1×1
d:	1×1
start:	1×1 (optional)
hammersley:	1×1 (optional)

output:

result:	$n \times d$

`_halton(`x`,` *start*`,` *hammersley*`)`:

input:

x:	$n \times d$
start:	1×1 (optional)
hammersley:	1×1 (optional)

output:

x:	$n \times d$

`ghalton(`*n*`,` *base*`,` *u*`)`:

input:

n:	1×1
base:	1×1
u:	1×1

output:

result:	$n \times 1$

4.5 Diagnostics

The maximum dimension, d, is 20. The scalar index *start* must be a positive integer, and the scalar u must be $0 \leq u < 1$.

5 Analysis

Here we use examples and simulations to provide intuition for why the Halton and Hammersley point sets provide better approximations to low-dimensional integrals, whereas pseudorandom points do better for higher-dimensional integrals. Our simulations lead to conjecture that pseudorandom point sets will outperform Halton and Hammersley point sets for $d > 10$.

Figure 1 shows a matrix scatterplot of Halton, Hammersley, and pseudorandom point sets of dimension 4 and length 200. For the pseudorandom point sets, each dimension is started with a different random-number generator seed. The scatter for dimensions 2–4 of the Hammersley and Halton point sets are the same for dimensions 1–3.[6]

6. The (1,1), (2,1), and (2,2) Halton boxes correspond to the (2,2), (3,2), and (3,3) Hammersley boxes, where box (i,j) is in the ith row and the jth column.

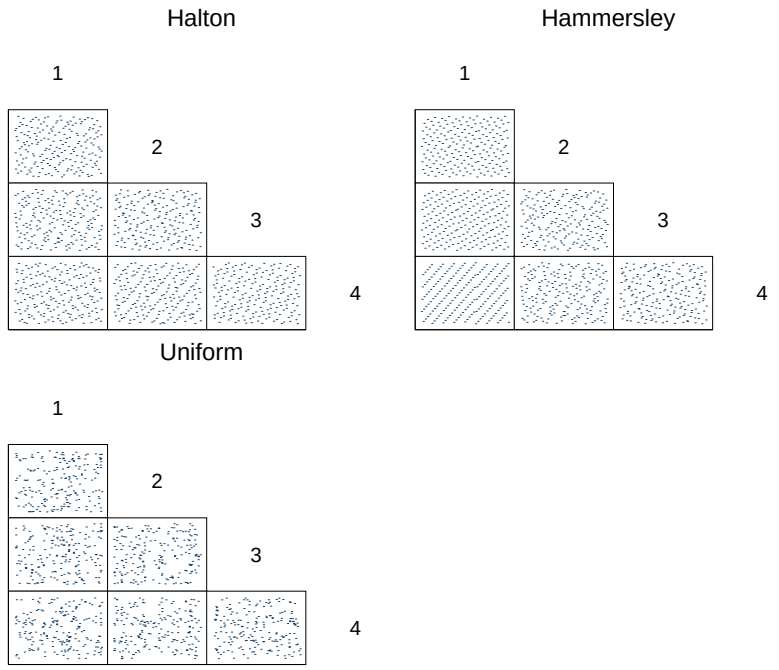

Figure 1: Matrix scatterplot of the first 200 points of the Halton and Hammersley sets of dimension 4 as well as a set of 200×4 uniform pseudorandom numbers

For a numerical exposition of uniform coverage of the point sets, we simulate a discrepancy measure by partitioning the $C^d = [0,1)^d$ into q^d subintervals and for each grid

$$\left[\frac{i_1 - 1}{q}, \frac{i_1}{q}\right) \times \left[\frac{i_2 - 1}{q}, \frac{i_2}{q}\right) \times \cdots \left[\frac{i_d - 1}{q}, \frac{i_d}{q}\right)$$

for $i_j = 1, \ldots, q$; we generate uniforms $u_{kj} \in \left[\frac{i_j - 1}{q}, \frac{i_j}{q}\right)$, and for repetition, $k = 1, \ldots, m$, we compute

$$D(\mathbf{X}, \mathbf{u}_k) = \left| \frac{N(\mathbf{X}, \mathbf{u}_k)}{n} - \prod_{j=1}^{d} u_{kj} \right|$$

where $\mathbf{X} = (\mathbf{x}_1, \ldots \mathbf{x}_n)'$ is the $n \times d$ matrix containing one of the point sets, $\mathbf{u}_k = (u_{k1}, u_{k2}, \ldots, u_{kd})'$, and the function $N(\mathbf{X}, \mathbf{u}_k)$ counts the number of vectors, $\mathbf{x}_i$, $i = 1, \ldots, n$, that are less than or equal to $\mathbf{u}_k$. The vector, $\mathbf{u}_k$, that maximizes the discrepancy in this region is saved. A local search is then made in a region around this vector that has a grid width of $2/n$,

$$\left[u_{k1}-\frac{1}{n},u_{k1}+\frac{1}{n}\right)\times\left[u_{k2}-\frac{1}{n},u_{k2}+\frac{1}{n}\right)\times\ldots\left[u_{kd}-\frac{1}{n},u_{kd}+\frac{1}{n}\right)$$

The computations ensure that $u_{ki}-1/n\geq 0$ and $u_{ki}+1/n<1$ for all ki. A search for a new maximum is performed within the subregion generating up to $\lceil\sqrt{d}\,\rceil\cdot 10$ new uniform vectors. If a new maximum is found, another local search is done in a region centered about this new maximum. This searching continues until a new maximum is no longer found. After the random search for a local maximum completes, the algorithm moves on to another of the q^d grids.

By partitioning $[0,1)$ into subintervals and generating uniforms on $[(i_j-1)/q, i_j/q)$, $j=1,\ldots,q$, the random search for local maxima is concentrated on each subregion. We thought that this method would yield a more efficient (random) search for a maximum on $[0,1)^d$, especially for larger dimensions, d. However, the larger the dimension and with q held constant, the number of grids, q^d, and the total number of uniform vectors used in the search, N, increases dramatically. Moreover, the searches for local maxima are expensive in terms of the number of uniform vectors generated, but by casual observation it was worth the effort. We tried a more naive approach to simulating the discrepancy by generating millions of uniform vectors on C^d and taking the maximum, but the choice of N, the total number of vectors, was too subjective. By contrast, with our proposed search algorithm, the size of N is driven by the algorithm after we choose m, the number of uniform vectors for the initial search in each grid, and q, the number of partitions for each dimension. For each estimate, we used $q=10$ so we only simulate the discrepancies up to dimension $d=6$. We used $m=\lceil\sqrt{d}\rceil\cdot 10$ initial replications for each search for the local maximum on each grid, where $\lceil x\rceil$ is the integer value of x such that $x\leq\lceil x\rceil<x+1$ (see [M-5] **trunc( )**).

We will discuss the simulated discrepancy estimates for $n=200$ with $d=2,\ldots,6$ later and give the discrepancy estimates for the point sets displayed in figure 1, where $d=4$ and $n=200$.

To compute the simulated discrepancy for $d=4$ with $q=10$, $10^4=10{,}000$ grids were used. We used $m=20$ replicates within each grid to obtain the initial maxima. A total of $N_h=1{,}274{,}030$, $N_x=1{,}164{,}460$, and $N_u=1{,}078{,}790$ uniform vectors were used in searching for the maximum simulated discrepancy for the Halton, Hammersley, and uniform point sets, resulting in estimates of 0.051, 0.038, and 0.108, respectively. We repeated the experiment, using a different random-number seed and produced estimates of 0.047, 0.044, and 0.103, for the same Halton, Hammersley, and uniform point sets. The number of uniform vectors used to produce these estimates were $N_h=1{,}273{,}720$, $N_x=1{,}178{,}140$, and $N_u=1{,}089{,}620$, respectively.

In figure 2, we show a line plot of the simulated discrepancies for $d=4$ and as n increases from 50 to 500. Each simulated discrepancy is computed using $m=20$, $q=10$. The plot demonstrates that for $d=4$ the uniform point set requires a substantially larger n to achieve the same numeric accuracy in (quasi) Monte Carlo integration as do the Halton and Hammersley point sets. For larger n, the simulated discrepancy for the Halton and Hammersley point sets are about equivalent. The value of N for each

estimate was on the order of 10^6. Here the local maxima search interval width of $2/n$ has an effect on the size of N. We found that if we fixed the width to 0.01, say, then N decreased with n. At $n = 50$, the Halton simulated discrepancy is smaller than the Hammersley.

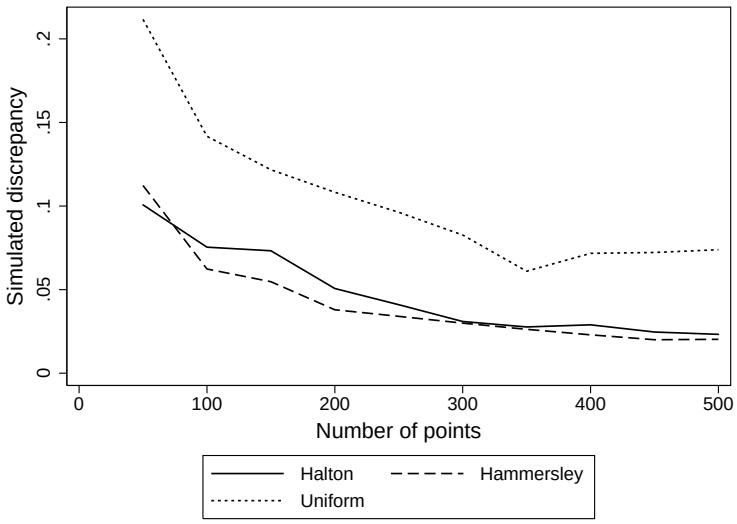

Figure 2: Line plot of the simulated discrepancy of the Halton, Hammersley, and a uniform point set with dimension 4 and as n goes from 50 to 500

Be careful when working with Halton sequences generated from large bases since these sequences can be highly correlated because of their large period [the number of points to cycle through $[0, 1)$]. For example, figure 3 contains matrix plots of Halton sets of length 200, 400, 600, and 800 based on primes 41, 43, 47, and 53. These are the primes that are used for dimensions 13–16. The simulated discrepancy (using $m = 20$ and $q = 10$) for these four dimension point sets are 0.218, 0.132, 0.079, and 0.047. For comparison, recall that the simulated discrepancy for our uniform point set of length 200 is 0.110, and we extended this point set to lengths 400, 600, and 800, which have simulated discrepancies of 0.072, 0.057, and 0.046, respectively. We speculate that for $d > 10$ dimensions and holding $n = 200$ that the uniform point sets will outperform the Halton and Hammersley. However, for such a large problem we would be compelled to use a value of n that is much larger than 200.

(*Continued on next page*)

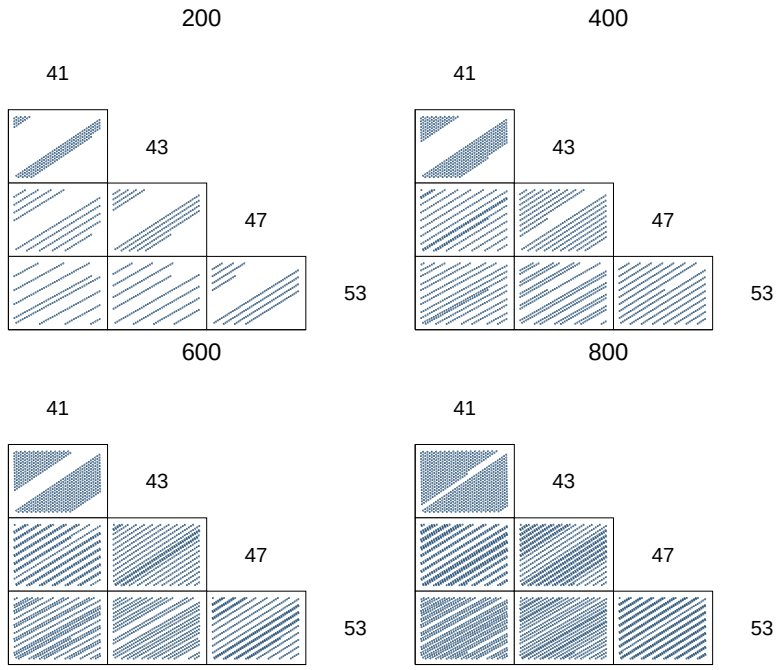

Figure 3: Matrix scatterplot of the first 200, 400, 600, and 800 points of a Halton set of dimension 4 computed from the primes 41, 43, 47, and 53

Table 1 contains the correlations of these sequences and, for reference, the correlations of the sequences based on primes 2, 3, 5, and 7.

Table 1: Correlations of the Halton sequences computed from the primes 2, 3, 5, and 7 and from the primes 41, 43, 47, and 53

Prime	2	3	5
3	−0.0197		
5	−0.0026	−0.0226	
7	−0.0100	−0.0071	−0.0087

Prime	41	43	47
43	0.4442		
47	−0.1129	0.1141	
53	0.0594	−0.1025	0.0127

In figure 4, we show the simulated discrepancy computed for the Halton, Hammersley, and uniform point sets for dimensions 2–6 and $n = 200$.

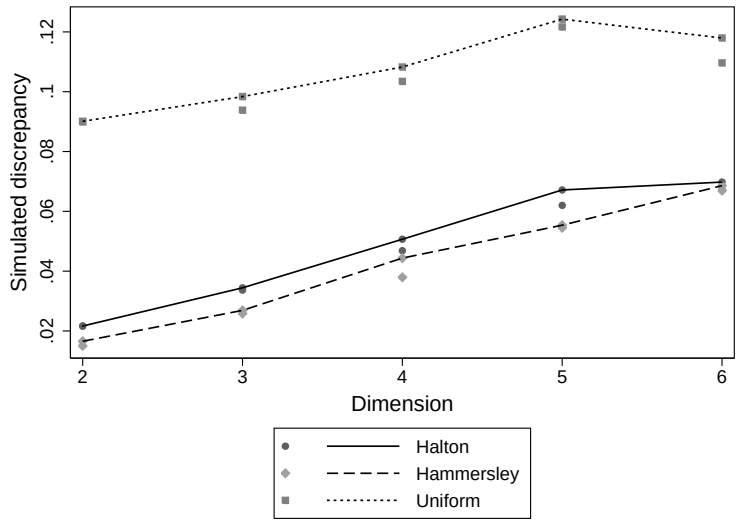

Figure 4: Plots of the simulated discrepancy for the Halton, Hammersley, and uniform point sets versus dimension

Table 2 gives the simulated discrepancies, the values of m, and the number of uniform vectors used for each estimate. The simulation parameter q is 10 for all simulations, and we ran the experiment twice to see how variable the estimates are.

Table 2: Dimension (d), initial replications (m), and the total number of uniform random vectors (N_h, N_x, and N_u) used to compute the simulated discrepancy for the Halton (d_h), Hammersley (d_x), and uniform (d_u) point sets, respectively

d	m	d_h	N_h	d_x	N_x	d_u	N_u
2	20	0.022	4,790	0.015	4,450	0.090	5,840
		0.022	4,910	0.017	4,380	0.090	6,070
3	20	0.034	71,920	0.026	62,210	0.094	81,530
		0.034	70,450	0.027	62,430	0.098	81,800
4	20	0.051	1,274,030	0.038	1,164,460	0.108	1,078,790
		0.047	1,273,720	0.044	1,117,140	0.103	1,089,620
5	30	0.067	22,744,800	0.055	21,166,660	0.122	18,933,860
		0.062	22,804,790	0.055	21,187,290	0.124	18,894,130
6	30	0.070	359,686,190	0.069	329,428,640	0.118	308,206,150
		0.067	359,581,140	0.067	329,310,390	0.110	307,968,840

The variability between simulated discrepancies for the two replicates demonstrates that the algorithm works only moderately well, but it is good enough to illustrate the theory of section 3.1.

Theoretically, the discrepancy for the uniform point set is independent of the dimension. The graph shows that the uniform discrepancy slightly increases to about 0.1 where a potential leveling off occurs, but we need to continue the simulations beyond dimension 6 to really tell. The plot also demonstrates that the discrepancies for the Halton and Hammersley sets are increasing with dimension, but not exponentially as the theory proposes (see section 3.1). We can speculate that this phenomenon occurs beyond dimension 6 with $n = 200$, but simulating beyond $d = 6$ with the same experiment parameters, m and q, is intractable. These simulation experiments support the theory that Hammersley point sets will give slightly better performance over the Halton point set in quasi–Monte Carlo integration.

Again, we use a specific uniform point set where each pseudorandom sequence for each dimension is started from a unique random-number seed, and as the dimension increased the seeds for lower dimensions are reused so that these sequences remained the same.

These simulations lead us to conjecture that pseudorandom point sets will provide better approximations to integrals of dimension greater than 10. In future work, we plan to seek stronger support for this conjecture and to look for other quasirandom sequences whose point sets work well in higher dimensions.

6 References

Fang, K.-T., and Y. Wang. 1994. *Number-theoretic Methods in Statistics*. London: Chapman & Hall.

Genz, A. 1992. Numerical computation of multivariate normal probabilities. *Journal of Computational and Graphical Statistics* 1: 141–149.

Niederreiter, H. 1992. *Random Number Generation and Quasi-Monte Carlo Methods*. Philadelphia: SIAM.

Train, K. 2003. *Discrete Choice Methods with Simulation*. Cambridge: Cambridge University Press.

Wang, X., and F. J. Hickernell. 2000. Randomized Halton sequences. *Mathematical and Computer Modeling* 32: 887–899.

About the authors

David Drukker is director of econometrics at StataCorp.

Richard Gates is a senior statistical software engineer at StataCorp.