
**IMPLEMENTATION AND TEST OF
THE NIEDERREITER-XING THIRD CONSTRUCTION
FOR LOW DISCREPANCY SEQUENCES**

CHONG HUI TAN
SINGAPORE MANAGEMENT UNIVERSITY
50 STAMFORD ROAD, #05-01, SINGAPORE 178899
CHONGHUITAN@SMU.EDU.SG

ABSTRACT.

1. INTRODUCTION

Monte Carlo (MC) methods are used to simulate stochastic models for the purpose of pricing and risk management in quantitative finance. With respect to integration, the distributional properties of the random points produced by Monte Carlo methods can however be improved asymptotically by deterministically chosen points. Classes of such points, such as the van der Corput, Halton, Faure, and Sobol' sequences are classically known. The Niederreiter sequences are defined more recently. The property that characterizes their good distributional properties is known as discrepancy. These point classes have been shown to have low discrepancies and are therefore known as low discrepancy sequences. The application of low discrepancy sequences to integration problems is categorized under Quasi-Monte Carlo (QMC) methods.

In Niederreiter and Xing [2001], the authors report the construction of low discrepancy sequences using tools from function field theory. In brief, the idea of their work is as follows. What is common about how the van der Corput, Faure and Niederreiter sequences are constructed is the digital method. This defines the n -th element of the sequence by transforming the digits of n when it is expressed in base q into the digits (also in base q) of s numbers which constitute the coordinates of a point in $[0, 1]^s$. The transformation is essentially linear with a fixed set of coefficients that characterize a particular low discrepancy sequences. By Niederreiter and Xing [2001], the transformation matrix

Date: 19 December 2007.

that is used to generate a low discrepancy sequence is defined by means of a suitable one-variable function field over a finite field together with a finite sequence of places of the function field. From this generalized perspective, the van der Corput, Faure and Niederreiter sequences arise from the simplest rational function fields. By applying the digital method to function fields of higher genera, Niederreiter and Xing have made potentially available a large class of low discrepancy sequences in addition to the classical ones.

In this article, we contribute to the literature an implementation of the third construction of Niederreiter and Xing [2001]. This is the most powerful method amongst the four described therein as it imposes the least on the input data. A previously published implementation of a restricted version of the third construction using the computational algebra system Kash and C++ can be found in Pirsic [2002] and an implementation of a scrambled version can be found in Hong and Hickernell [2003], which is based on the former paper. Using the computational algebra system Magma (Bosma et al. [1997]), we have been able to implement the third construction in its entire generality. We also take the opportunity to investigate the statistical properties of some of the low discrepancy sequences generated by our code and gauge how they perform on a standard pricing problem. For the benefit of the Monte Carlo community, we have made available the necessary code to automatically get the online Magma Calculator to compute low discrepancy sequences and return them in a suitable file format on the local computer (www.mysmu.edu.sg/faculty/chonghuitan).

The paper is organized as follows. In Sections 2 and 3, we review the notion of low discrepancy sequences and the digital method. In Section 4, we describe the third construction. The Magma implementation is given in the Appendix. In Section 5, we give some examples of low discrepancy sequences and consider their statistical properties. In Section 6, we apply these examples to a high-dimensional pricing problem.

Standard notation from function field theory used in this article:

- \mathbb{F}_q : finite field with q elements
- $\mathcal{L}(D)$: the Riemann-Roch space of functions f such that $(f) + D \geq 0$
- $l(D)$: the dimension of $\mathcal{L}(D)$ as a vector space
- $v_P(-)$: the valuation at P
- $v_\infty(-)$: the valuation at ∞ in the rational function field $k(x)$, defined by $v_\infty(f/g) := \deg(g) - \deg(f)$

2. LOW DISCREPANCY SEQUENCES

The basis of the MC and the QMC method rests upon the approximation

$$\int_{[0,1]^s} f(x) dx \approx \frac{1}{n} \sum_{i=1}^n f(x_i),$$

where the sequence of points $x_1, x_2, \dots, x_n \in [0, 1]^s$ are in some sense well-distributed. In the MC method, these points are chosen randomly with respect to the uniform measure on $[0, 1]^s$. In the QMC method, they are carefully constructed with arithmetic techniques. Low discrepancy sequences arise from this context.

In general, how well the average of function values approximates the integral depends on the function itself as well as how the sample points x_1, x_2, \dots, x_n are distributed in $[0, 1]^s$. This is more precisely described by the Koksma-Hlawka inequality:

$$\left| \int_{[0,1]^s} f(x) dx - \frac{1}{n} \sum_{i=1}^n f(x_i) \right| \leq V(f) D^*(x_1, x_2, \dots, x_n),$$

where $V(f)$ is a measure of the smoothness of the function f called the Hardy-Krause variation (Glasserman [2003]), and $D^*(x_1, x_2, \dots, x_n)$ is called the star discrepancy of the sample points x_1, x_2, \dots, x_n . If f is given, such as when a given financial instrument is to be priced, we may then seek for samples with values of star discrepancies as small as possible.

For $x_1, x_2, \dots, x_n \in [0, 1]^s$, the star discrepancy $D^*(x_1, x_2, \dots, x_n)$ is defined to be

$$D^*(x_1, x_2, \dots, x_n) = \sup_A \left| \frac{\#\{x_i \in A\}}{n} - m(A) \right|,$$

where the supremum is taken over all rectangles of the form $A = \prod_{i=1}^s [a_i, b_i] \subset [0, 1]^s$, the fraction over n is the proportion of the sample that belongs to A , and $m(A)$ is volume of A . It measures how well a sample of points is distributed in $[0, 1]^s$ with respect to the approximation of volumes of boxes by the fraction of those points in the sample that belong to the boxes.

It has been conjectured that the star discrepancy of an infinite sequence grows asymptotically as $n^{-1}(\log n)^s$. More precisely, for an infinite sequence of points $S : x_1, x_2, \dots$ in $[0, 1]^s$ and for each n , let $D_n^*(S)$ denote the star discrepancy of the first n terms of the sequence. Then the conjecture is that if

$$D_n^*(S) = O\left(n^{-1}(\log n)^\epsilon\right),$$

then $\epsilon \geq s$. This has been proven when $s = 1$. Any sequence S whose star discrepancy grows at $\epsilon = s$ is called a low discrepancy sequence (Niederreiter and Xing [2001]).

Niederreiter and Xing use methods from the theory of function fields to generate what are called (t, s) -sequences which are special cases of low discrepancy sequences. Such a (t, s) -sequence S over a finite field \mathbb{F}_q satisfies

$$D_n^*(S) \leq C_q(s, t)n^{-1}(\log n)^s + O(n^{-1}(\log n)^{s-1}), \quad (n \geq 2)$$

where

$$C_q(s, t) = \begin{cases} \frac{q^t}{s} \left(\frac{q-1}{2 \log q} \right)^s, & \text{if either } s = 2, \text{ or } q = 2, s = 3, 4, \\ \frac{q^t}{s!} \cdot \frac{q-1}{2 \lfloor q/2 \rfloor} \left(\frac{\lfloor q/2 \rfloor}{\log q} \right)^s, & \text{otherwise.} \end{cases}$$

An infinity of low discrepancy sequences is consequentially made available to us. The bound on the star discrepancy above suggests the heuristic of searching for each s , (t, s) -sequences with values of t as small as possible. However, we cannot therefore conclude that a smaller value of t implies a lower star discrepancy. It will be seen in the Niederreiter-Xing algorithm later that this heuristic translates into the search of function fields with many places of small degrees.

The classical low discrepancy sequences of van der Corput, Faure, Niederreiter and Sobol' are (t, s) -sequences. A van der Corput sequence is a $(0, 1)$ -sequence in base p , where p is a prime number. A Faure sequence is a $(0, s)$ -sequence in base $p \geq s$. The Niederreiter sequences are (t, s) -sequences in any prime power base q (in fact, Niederreiter allows the base to be arbitrary). A Sobol' sequence is a $(T_2(s), s)$ -sequence in base 2, where $T_2(s)$ is the sum of degrees of the first s primitive polynomials (in a listing of non-decreasing degrees) over \mathbb{F}_2 . The Halton sequence is not a (t, d) -sequence even though it is a low discrepancy sequence.

The reader may refer to Fang and Wang [1994] for a good coverage of the basic issues concerning the distribution of points on a domain for the purpose of integration.

3. THE DIGITAL METHOD

The van der Corput, Halton, Faure and Niederreiter are generated by means of the digital method which we will now describe.

Let q be a positive power of a prime number, \mathbb{F}_q be the finite field with q elements, and for a natural number n , write

$$n = \sum_{r=0}^{\infty} a_r(n)q^r$$

for its representation in base q .

The digital method transforms each natural number n to a point in $[0, 1]^s$:

$$n \mapsto (x_n^{(1)}, x_n^{(2)}, \dots, x_n^{(s)}) \in [0, 1]^s.$$

Each coordinate $x_n^{(i)}$, when expressed in base q ,

$$\begin{aligned} x_n^{(i)} &= (0.x_n^{(i)}(1)x_n^{(i)}(2)x_n^{(i)}(3)\cdots)_q \\ &:= \sum_{j=1}^{\infty} x_n^{(i)}(j)q^{-j} \in [0, 1], \end{aligned}$$

has its digits given by linear transformations of the digits $a_r(n)$:

$$x_n^{(i)}(j) = \sum_{r=0}^{\infty} c_{j,r}^{(i)} a_r(n) \pmod{q}.$$

Note that the sum above is actually finite since the expansion of n in base q has only a finite number of digits.

Also note the significance of the indices i, j, r and n :

- i refers to the i -th coordinate of a point from $[0, 1]^s$
- j refers to the j -th digit in the base- q expansion of a number from $[0, 1]$
- r refers to the r -th digit in the base- q expansion of a natural number
- n refers to the n -th number in the low discrepancy sequence

The coefficients $c_{j,r}^{(i)}$ ($i = 1, 2, \dots, s$, $j \geq 1$, $r \geq 0$) are key to generating a low discrepancy sequence. For convenience, we call them the generator coefficients of the particular sequence.

3.1. van der Corput and Halton Sequences. Let p be a prime number. The van der Corput sequence belongs to $[0, 1]$. It applies the digital method with

$$c_{j,r}^{(1)} = \begin{cases} 1 & \text{if } j = r \\ 0 & \text{if } j \neq r \end{cases}.$$

Thus,

$$n = \sum_{r=0}^{\infty} a_r(n)p^r \mapsto \sum_{r=0}^{\infty} a_r(n)p^{-r} =: [n]_p \in [0, 1].$$

The Halton sequence in dimension s is given by

$$n \mapsto ([n]_{p_1}, [n]_{p_2}, \dots, [n]_{p_s}) \in [0, 1]^s,$$

where $p_1 < p_2 < \dots < p_s$ are the first s prime numbers.

3.2. Faure Sequences. A Faure sequence in dimension s may be generated by selecting a prime number p such that $p \geq s$. This ensures that there are at least s elements in the finite field \mathbb{F}_p .

Let a_1, a_2, \dots, a_s be s distinct elements of \mathbb{F}_p . The coefficients $c_{j,r}^{(i)}$ are defined by means of the Dolbeault expansion of $(x - a_i)^{-j}$ (expansion in negative powers of x):

$$\begin{aligned} \sum_{r=j}^{\infty} c_{j,r-1}^{(i)} x^{-r} &:= (x - a_i)^{-j} \\ &= x^{-j} (1 - a_i x^{-1})^{-j} \\ &= x^{-j} \sum_{r=0}^{\infty} \binom{-j}{r} (-a_i x^{-1})^r \\ &= \sum_{r=0}^{\infty} \binom{j+r-1}{r} a_i^r x^{-r-j} \\ &= \sum_{r=j}^{\infty} \binom{r-1}{r-j} a_i^{r-j} x^{-r}. \end{aligned}$$

In other words,

$$c_{j,r}^{(i)} = \begin{cases} \binom{r}{r+1-j} a_i^{r+1-j} \pmod{p} & \text{for } r \geq j-1, \\ 0 & \text{otherwise.} \end{cases}$$

3.3. Niederreiter Sequences. Let q be a prime power. A Niederreiter sequence in dimension s may be generated with distinct irreducible polynomials F_1, F_2, \dots, F_s of $\mathbb{F}_q[x]$.

Write $e_i := \deg F_i$. For given i, j , write $j-1 = \lfloor \frac{j-1}{e_i} \rfloor e_i + r_{i,j}$, where $r_{i,j} \in \{0, 1, \dots, e_i - 1\}$. Define

$$f_j^{(i)} = \frac{x^{r_{i,j}}}{F_i^{\lfloor \frac{j-1}{e_i} \rfloor + 1}}.$$

The coefficients $c_{j,r}^{(i)}$ are defined by means of the Dolbeault expansion of $f_j^{(i)}$:

$$f_j^{(i)} = \sum_{r=0}^{\infty} c_{j,r}^{(i)} x^{-r-1}.$$

4. THE THIRD CONSTRUCTION OF NIEDERREITER AND XING

A function field K/\mathbb{F}_q is assumed to have one rational point P_∞ , a positive divisor D with $\deg(D) = 2g(K/\mathbb{F}_q)$, such that $P_\infty \notin \text{Supp}(D)$. By the Riemann-Roch Theorem, since $l(D) = g + 1$ and $l(D - (2g + 1)P_\infty) = 0$, there exist integers $0 = n_0 < n_1 < \dots < n_g \leq 2g$ such that

$$l(D - n_f P_\infty) = l(D - (n_f + 1)P_\infty) + 1$$

for $0 \leq f \leq g$.

Choose s distinct places P_1, P_2, \dots, P_s which are different from P_∞ and set $e_i = \deg(P_i)$ ($i = 1, 2, \dots, s$).

The algorithm reads:

- (1) For each integer f such that $0 \leq f \leq g$, choose $w_f \in \mathcal{L}(D - n_f P_\infty) \setminus \mathcal{L}(D - (n_f + 1)P_\infty)$ so that $\{w_0, w_1, \dots, w_g\}$ forms a basis of $\mathcal{L}(D)$. For each $1 \leq i \leq s$, successively add basis vectors along the chain

$$\mathcal{L}(D) \subset \mathcal{L}(D + P_i) \subset \mathcal{L}(D + 2P_i) \subset \dots$$

to obtain a basis $\{w_0, w_1, \dots, w_g, k_1^{(i)}, k_2^{(i)}, \dots, k_{n_i e_i}^{(i)}\}$ of $\mathcal{L}(D + nP_i)$.

- (2) Let z be a local uniformizing parameter at P_∞ and for $r = 0, 1, \dots$, set

$$z_r = \begin{cases} z^r, & \text{if } r \notin \{n_0, n_1, \dots, n_g\}, \\ w_f, & \text{if } r = n_f \text{ for some } f \in \{0, 1, \dots, g\}, \end{cases}$$

and for $1 \leq i \leq s$ and $j \geq 1$, expand $k_j^{(i)}$ at P_∞ into

$$k_j^{(i)} = \sum_{r=0}^{\infty} a_{j,r}^{(i)} z_r,$$

where $a_{j,r}^{(i)} \in \mathbb{F}_q$.

-
- (3) Define the elements to be used in the digital method $c_{j,0}^{(i)}, c_{j,1}^{(i)}, \dots$ by

$$(c_{j,0}^{(i)}, c_{j,1}^{(i)}, \dots) = (a_{j,0}^{(i)}, a_{j,1}^{(i)}, \dots, \dots)^*,$$

where the asterisk means that the n_0 -th, n_1 -th, \dots , n_g -th elements are removed from the sequence.

The following result holds for the generator coefficients $(c_{j,r}^{(i)})_{i,j,r}$ generated by the third construction (Niederreiter and Xing [2001]):

Theorem 4.1. *The third construction produces a digital (t, s) sequence with*

$$t = g(K/\mathbb{F}_q) + \sum_{i=1}^s (\deg(P_i) - 1).$$

This result suggests that one should search for function fields with many places of low degrees for sources of good low discrepancy sequences.

The van der Corput, Faure and Niederreiter sequences are generated by suitable inputs into the algorithm. The common underlying function fields for them are the rational function fields over suitable finite fields. As the genus of a rational function field is $g = 0$, the divisor D is 0, the basis of $\mathcal{L}(0)$ is the singleton comprising the element 1. The point P_∞ is the point at infinity.

The van der Corput sequence has $s = 1$, and P_1 is the rational place corresponding to the rational point 0. The filtration of spaces

$$\mathcal{L}(0) \subset \mathcal{L}(P_1) \subset \mathcal{L}(2P_1) \subset$$

has the filtered basis $1; 1/x; 1/x^2; \dots$. The local uniformizer at infinity $z = \frac{1}{x}$ leads to the expansion

$$1/x^j = z^j, \quad (j = 0, 1, 2, \dots).$$

Hence the generator coefficients are given by

$$c_{j,r}^{(1)} = \begin{cases} 1 & \text{if } j = r \\ 0 & \text{if } j \neq r \end{cases}.$$

The Faure sequence is defined over a finite field \mathbb{F}_p and has a dimension $s \leq p$. The places P_1, P_2, \dots, P_s are rational places corresponding to distinct points $a_1, a_2, \dots, a_s \in \mathbb{F}_p$. For each $i = 1, 2, \dots, s$, the filtration of spaces

$$\mathcal{L}(0) \subset \mathcal{L}(P_i) \subset \mathcal{L}(2P_i) \subset \dots$$

has the filtered basis $1; (x - a_i)^{-1}; (x - a_i)^{-2}; \dots$. The local uniformizer at infinity $z = \frac{1}{x}$ leads to the expansion

$$(x - a_i)^{-j} = \sum_{r=j}^{\infty} \binom{r-1}{r-j} a_i^{r-j} x^{-r-j}, \quad (j = 0, 1, 2, \dots).$$

Hence the generator coefficients are given by

$$c_{j,r}^{(i)} = \begin{cases} \binom{r}{j-1} a_i^{r+1-j} \pmod{p} & \text{for } r \geq j-1, \\ 0 & \text{otherwise.} \end{cases}$$

The Niederreiter sequence is defined over a finite field \mathbb{F}_q . The places P_1, P_2, \dots, P_s correspond to the irreducible polynomials F_1, F_2, \dots, F_s of $\mathbb{F}_q[x]$, respectively of degree $e_i := \deg F_i$, and listed in increasing degrees. The filtration of spaces

$$\mathcal{L}(0) \subset \mathcal{L}(P_i) \subset \mathcal{L}(2P_i) \subset \dots$$

has the filtered basis $1; f_1^{(i)}, f_2^{(i)}, \dots, f_{e_i}^{(i)}; f_{e_i+1}^{(i)}, f_{e_i+2}^{(i)}, \dots, f_{2e_i}^{(i)}; \dots$, with the local expansion at P_∞ given by

$$f_j^{(i)} = \sum_{r=0}^{\infty} c_{j,r}^{(i)} x^{-r-1},$$

where for each $j \geq 1$, $j-1 = \lfloor \frac{j-1}{e_i} \rfloor e_i + r_{i,j}$ and $r_{i,j} \in \{0, 1, \dots, e_i - 1\}$.

Refer to the Appendix for our implementation of the third construction in Magma.

5. NUMERICAL TESTS

APPENDIX A. MAGMA CODE

The following is an implementation of the third construction of Niederreiter and Xing using the computational algebra system Magma. The code is amply commented. Nevertheless, we briefly describe what the code does.

The code is divided into four functions - InitList, BaseExtension, SeriesExpansion, StartProcess - and a parameters definition portion towards the end. Roughly speaking, InitList is concerned with the preamble of the construction as is described in Section 4, BaseExtension is concerned with Point 1, SeriesExpansion is concerned with Point 2, and StartProcess is concerned with Point 3 as well as chaining the whole program together by calling the prior three functions in sequence.

Keeping the notation from Section 4, the parameters that need to be set for the code to run are the function field K , natural numbers s , JJ and RR which are upper bounds of the indices i , j and r respectively, and possibly the distinguished place P_∞ , the divisor D and the sequence of places P_1, P_2, \dots, P_s . Setting P_∞, D and P_1, P_2, \dots, P_s as 0 leaves the program to select them for us.

The last line of the code starts the program running and the output is the matrix

$$\left(c_{j,r}^{(i)} \right)_{1 \leq i \leq s, 1 \leq j \leq JJ, 0 \leq r \leq RR}$$

```

/*
 * Parameters
 * K: function field
 * s: of (t,s)-sequence, signifies dimension
 *
 * Returns
 * P_inf
 * seq: the sequence of places P1,P2,...,Ps
 * D: the positive divisor of degree 2g
 * numseq: the sequence n0,n1,...,ng
 */
InitList := function(K,s,P_inf,D,seq)
/*
 * Genus
 */
g := Genus(K);

/*
 * Define the maximal finite and infinite orders
 */
O_S := MaximalOrderFinite(K);
O_inf := MaximalOrderInfinite(K);
BR_S := BaseRing(O_S);
BR_inf := BaseRing(O_inf);
X := BR_S.1;

/*
 * Define P_infinity
 */
if Type(P_inf) eq Type(0) then
  P_inf := InfinitePlaces(K)[1];
end if;

```

```

/*
 * Define P_1,P_2,...,P_s
 * These are obtained from places of lowest degrees
 */
if Type(seq) eq Type(0) then
  seq := [];
  i := 1;
  left := s;
  while left gt 0 do
    tmp := Exclude(Places(K,i),P_inf);
    I := [1..Minimum(left,#tmp)];
    seq := seq cat tmp[I];
    i := i + 1;
    left := left - #tmp;
  end while;
end if;

/*
 * Define D
 */
if Type(D) eq Type(0) then
  emp := [];
  for i in [1..(2*g)] do
    if #Exclude(Places(K,i),P_inf) eq 0 then
      emp := Append(emp,false);
    else
      emp := Append(emp,true);
    end if;
  end for;

  flag := false;
  D := 0 * P_inf;

  for i in [1..(2*g)] do
    part := Partitions(2*g,i);
    for j in part do
      flg := true;
      for k in j do
        if emp[k] eq false then
          flg := false;
          break;
        end if;
      end for;
      if flg eq true then
        for k in j do
          D := D + Exclude(Places(K,k),P_inf)[1];
        end for;
        flag := true;
        break;
      end if;
    end for;
    if flag eq true then
      break;
    end if;
  end for;
end if;

```

```

/*
 * Define n_0, n_1, ..., n_g
 */
D_prime := D - (2*g + 1) * P_inf;
numseq := [0..(2*g)];
gn := GapNumbers(D_prime, P_inf);
for i in gn do
  numseq := Exclude(numseq, 2*g+1-i);
end for;

return P_inf, seq, D, numseq;

end function;

/*
 * Parameters
 * vecseq: a sequence of abstract vectors
 * which maps via the RiemannRochSpace map to a basis of L(D)
 * D: divisor
 * E: divisor which is not less than D
 *
 * Returns
 * newList: a sequence of abstract vectors
 * use RR2f - the map of the RiemannRochSpace(E) - to obtain a basis of L(E)
 * The first #vecseq elements constitute the basis of L(D) that we started with
 *
 * Note: Magma maps an array of one element to one element - the check below is for this
 */
BaseExtension := function(vecseq, D, E)

RR1, RR1f := RiemannRochSpace(D);
RR2, RR2f := RiemannRochSpace(E);
tmp := Inverse(RR2f)(RR1f(vecseq));
if Type(tmp) ne Type([0]) then
  tmp := [tmp];
end if;
newvList := ExtendBasis(tmp, RR2);

return newvList, RR2f;

end function;

/*
 * elem is the element to be series-expanded
 * num is the degree at which expansion stops
 * plc is its place
 * extraunif is the list w0,w1,...,wg
 * extrapos is the list n0,n1,...,ng
 *
 * acc is the truncated power series
 * coeff is the list of coefficients of acc
 * unif is the uniformizer
 *
 * num must be greater than or equal 0
 *
 * This function works only for expansion of elements with no pole at plc
 */
SeriesExpansion := function(elem, num, plc, extraunif, extrapos)

```

```

acc := 0;
coeff := [];
unif := LocalUniformizer(plc);

for i in [0..num] do
  if i in extrapos then
    u := extraunif[Position(extrapos,i)];
  else
    u := unif^i;
  end if;

  ev := Evaluate(elem/u,plc);
  acc := acc + ev * u;
  elem := elem - ev * u;
  coeff := Append(coeff,ev);

end for;

return acc, coeff, unif;

end function;

/*
 * This is the workhorse function that
 * churns out the cubic matrix of c coefficients in the digital method
 * It weaves together the functions defined above
 *
 * Parameters
 * K: function field
 * s: dimension of underlying space of QMC points
 * JJ, RR: see below
 *
 * Returns
 * carray: an s x JJ x (RR+1) array
 */
StartProcess := function(K,s,JJ,RR,P_inf,D,seq)

/*
 * Genus
 */
g := Genus(K);

/*
 * Invoke InitList
 */
P_inf, seq, D, numseq := InitList(K,s,P_inf,D,seq);
n_g := numseq[g+1];

/*
 * Define w0,w1,...,wg as basis of L(D)
 * ws is [w0,w1,...,wg]
 */
V0, RRmp := RiemannRochSpace(D - n_g * P_inf);
V0 := Basis(V0);
for i in [1..n_g] do
  V0, RRmp := BaseExtension(V0,D - (n_g - i + 1) * P_inf,D - (n_g - i) * P_inf);
end for;

```

```

ws := Reverse(RRmp(V0));

carray := [];

for ii in [1..s] do

/*
* P is P_ii for some ii from [1,s]
*/
P := seq[ii];
V := V0;

/*
* VF is [k^{(ii)}_jj : jj = 1,2,...]
* Repeat until #V is at least 1+g+JJ
*/
i := 1;
while #V lt (1+g+JJ) do
V, RRmp := BaseExtension(V,D + (i - 1) * P, D + i * P);
i := i + 1;
end while;

VF := RRmp(V);
VF := VF[(g+2)..(JJ+g+1)];

tmparray := [];

for jj in [1..JJ] do
/*
* Invoke SeriesExpansion
*/
acc, coeff, unif := SeriesExpansion(VF[jj],RR+1+g+1,P_inf,ws,numseq);

/*
* Remove the n0-th,n1-th,...,ng-th elements, after which
* coeff comprises [c^{(ii)}_{jj,rr}, rr = 0,1,2,...,RR]
*/
for i in [0..g] do
j := numseq[g+1-i]+1;
if j le #coeff then
coeff := Remove(coeff,numseq[g+1-i]+1);
end if;
end for;

tmparray := Append(tmparray,coeff[1..(RR+1)]);

end for;

carray := Append(carray,tmparray);

end for;

return carray;

end function;

/*
* DEFINE FUNCTION FIELD

```

```

*
* p - prime characteristic
* q - order of prime field
*/
p := 2;
q := p^1;
k := GF(q);
R<x> := FunctionField(k);
P<y> := PolynomialRing(R);
K<alpha> := FunctionField(y - x);

/*
* SET PARAMETERS II,JJ,RR
*
* The indices ii,jj,rr from c^{(ii)}_{jj,rr}
*
* s is the dimension of the integrand space
* ii is in [1,s]
*
* jj is the decimal place of x^{(ii)}_n (in [0,1]) in base q
* JJ is the desired accuracy of the coordinates, i.e. up to M-th base-q-th place
* jj is in [1,JJ]
* N is the number of QMC points to be generated
* rr is the decimal place of n (a positive integer)
* rr is in [0,RR]
*/
s := 5;
N := 1000;
RR := Ceiling(Log(N)/Log(q));
JJ := 10;

/*
* Set P_inf, D, seq to 0 if they are not user-defined
*/

/*
* Specify P_inf here
*/
/* P_inf := InfinitePlaces(K)[1];*/
P_inf := 0;

/*
* Specify D here
*/
/* D := 0 * P_inf; */
D := 0;

/*
* Specify array of places [P1,P2,...,Ps] here
*/
/* seq := Exclude(Places(K,1) cat Places(K,2) cat Places(K,3), P_inf); */
seq := 0;

/*
* RUN HERE
* Returns an s x JJ x (RR+1) array
* Warning: P_inf must be defined before D or seq is defined
*/

```

`StartProcess(K,s,JJ,RR,P_inf,D,seq);`

REFERENCES

- W. Bosma, J. Cannon, and C. Playoust. The Magma algebra system I. The user language. *Journal of Symbolic Computing*, 24(3-4):235–265, 1997.
- K.-T. Fang and Y. Wang. *Number-theoretic Methods in Statistics*, volume 51 of *Monographs on Statistics and Probability*. Chapman & Hall, 1994.
- P. Glasserman. *Monte Carlo Methods in Financial Engineering*, chapter 5. Springer, 2003.
- H.S. Hong and F.J. Hickernell. Implementing scrambled digital sequences. *ACM Transactions on Mathematical Software*, 29(2):95–109, 2003.
- H. Niederreiter and C.P. Xing. *Rational points on curves over finite fields: Theory and Applications*, volume 285 of *London Mathematical Society Lecture Note Series*. Cambridge, 2001.
- G. Pirsic. A software implementation of niederreiter-xing sequences. In K.-T. Fang, F.J. Hickernell, and H. Niederreiter, editors, *Monte Carlo and Quasi-Monte Carlo Methods 2000*. 2002.