Low Discrepancy Toolbox Manual

Michael Baudin

Version 0.1
May 2013

Abstract

This document is an introduction to the Low Discrepancy module v0.5.

Contents

1 Gallery **4**
- 1.1 The Van Der Corput sequence . . . 4
- 1.2 Halton Sequence . . . 6
- 1.3 Scrambled (RR2) Halton Sequence . . . 6
- 1.4 Leaped Halton Sequence . . . 10
- 1.5 Reverse-Halton Sequence . . . 10
- 1.6 Sobol Sequence . . . 12
- 1.7 Faure Sequence . . . 12
- 1.8 Niederreiter Sequence . . . 12
- 1.9 Scrambled Sobol Sequence . . . 14
- 1.10 Note on 2D plots . . . 15
- 1.11 In 3D . . . 17

2 Nets **21**
- 2.1 Definitions . . . 21
- 2.2 Theorems . . . 23
- 2.3 Discrepancy . . . 24
- 2.4 Estimate of the mean by a low discrepancy sequence . . . 24
- 2.5 Halton sequence . . . 25
- 2.6 Faure sequence . . . 26
- 2.7 Niederreiter base 2 sequence . . . 29
- 2.8 Sobol sequence . . . 29
- 2.9 Leaped sequences and (t,m,s)-nets . . . 31
- 2.10 Notes and references . . . 36

3 The Van Der Corput sequence **37**
- 3.1 Introduction . . . 37
- 3.2 Van Der Corput sequence in base 2 . . . 38
- 3.3 Van Der Corput sequence in base 3 . . . 40
- 3.4 Properties . . . 42
- 3.5 Example . . . 43
- 3.6 Property . . . 47

4 The Halton sequence **49**
- 4.1 Introduction . . . 49
- 4.2 Scrambled Halton Sequence . . . 50

5 The Faure sequence **53**
5.1 Principles . . . 53
5.2 Properties . . . 54
5.3 Examples . . . 54
5.4 Properties of 1D projections . . . 56
5.5 Value of the prime base . . . 57
5.6 Notes and references . . . 61

6 Efficiency **62**
6.1 Error bounds . . . 62
6.2 The Ishigami example . . . 63
6.3 Efficiency of Quasi Monte-Carlo . . . 65

7 Projections **67**
7.1 Halton Sequence . . . 67
7.2 Faure Sequence . . . 72
7.3 Sobol Sequence . . . 76
7.4 Niederreiter (base 2) Sequence . . . 79
7.5 Two-dimensional correlations with increasing dimensions . . . 79
7.5.1 Numerical experiments . . . 79
7.5.2 Conclusions . . . 84

8 Thanks **85**

Bibliography **86**

Chapter 1

Gallery

The goal of this section is to gather experiments showing simple experiments and pictures of low disrepancy sequences.

1.1 The Van Der Corput sequence

The Van Der Corput base 2 sequence is the first dimension of the Halton sequence in 1 dimension. In the following script, we generate 15 points from the Halton sequence in 1 dimension.

```
u=lowdisc_ldgen(15,1,"halton")
```

The previous script produces the following output.

```
-->u=lowdisc_ldgen(15,1,"halton")
 u  =
    0.5
    0.25
    0.75
    0.125
    0.625
    0.375
    0.875
    0.0625
    0.5625
    0.3125
    0.8125
    0.1875
    0.6875
    0.4375
    0.9375
```

We notice that the values are organized block by block, where the size of a block is a power of two. The block #1 has x=0.5 (1 point). The block #2 has x=0.25 and x=0.75 (2 points). The block #3 has x=0.125 to x=0.875 (4 points). This is consistent with the fact that the Van Der Corput sequence that we consider uses the base 2.

In order to understand how this sequence fills the interval [0,1], we analyze the elements of the sequence in more detail. The following function prints each element of the sequence, by

decomposing the number $n = u2^p$, where u is the low discrepancy number and p is an appropriate integer. This allows to get the binary decomposition of the number u.

```
function printVDC(u,p)
    nrows=size(u,"r")
    for i=1:nrows
        n=u(i)*2^p;
        d=number_tobary(n,[],[],p);
        s=strcat(string(d'),"");
        mprintf("u=%f, n=%3d, binary=0.%s\n",u(i),n,s)
    end
endfunction
```

The following script computes the elements of the Van Der Corput sequence, block by block: we get the first point, then two more points, then 4 points, and finally 8 points.

```
lds = lowdisc_new("halton");
lds = lowdisc_configure(lds,"-dimension",1);
lds = lowdisc_startup (lds);
[lds,u]=lowdisc_next(lds,1);
printVDC(u,1)
[lds,u]=lowdisc_next(lds,2);
printVDC(u,2)
[lds,u]=lowdisc_next(lds,4);
printVDC(u,3)
[lds,u]=lowdisc_next(lds,8);
printVDC(u,4)
lds = lowdisc_destroy (lds);
```

The previous script produces the following output. For example, $u = 0.5625$ is decomposed in base-2 as 0.1001, since $0.5625 = 1/2 + 1/2^4$.

```
u=0.500000, n=  1, binary=0.1
u=0.250000, n=  1, binary=0.01
u=0.750000, n=  3, binary=0.11
u=0.125000, n=  1, binary=0.001
u=0.625000, n=  5, binary=0.101
u=0.375000, n=  3, binary=0.011
u=0.875000, n=  7, binary=0.111
u=0.062500, n=  1, binary=0.0001
u=0.562500, n=  9, binary=0.1001
u=0.312500, n=  5, binary=0.0101
u=0.812500, n= 13, binary=0.1101
u=0.187500, n=  3, binary=0.0011
u=0.687500, n= 11, binary=0.1011
u=0.437500, n=  7, binary=0.0111
u=0.937500, n= 15, binary=0.1111
```

Notice that the most significant digit (the 0 or 1 just after the decimal point) is alternatively 0, 1, 0, etc... Hence, the point altenates below and over the central value x=0.5.

In order to see where the points are located in the interval [0,1], we plot the points block-by-block, where the y-coordinate corresponds to the index of the block.

```
scf();
lds = lowdisc_new("halton");
lds = lowdisc_configure(lds,"-dimension",1);
lds = lowdisc_startup (lds);
k=1;
imax=5;
for i=1:5
    npoints=2^(i-1);
    [lds,u]=lowdisc_next(lds,npoints);
    plot(u',i,"bo")
    xstring(u',i,string(k:k+npoints-1))
    k=k+npoints;
end
lds = lowdisc_destroy (lds);
a = gca();
a.data_bounds=[
0 0
1 imax+1
];
xtitle("Van Der Corput Sequence (base 2)",..
"X","Block Index")
```

The previous script produces the figure 1.1.

We see that each block fills the interval with regular spacings, but that the order of the points is so that two consecutive points in the sequence always have x=0.5 in-between.

1.2 Halton Sequence

```
u=lowdisc_ldgen ( 1000 , 2 , "halton" );
scf();
plot(u(:,1),u(:,2),"bo")
xtitle("Halton: 1000 points","X1","X2")
```

The previous script produces the figure 1.2.

1.3 Scrambled (RR2) Halton Sequence

The following script produces a scrambled Halton sequence, with the permutation of Kocis and Whiten (RR2).

```
u=lowdisc_ldgen ( 1000 , 2 , "halton-scrambled" );
scf();
plot(u(:,1),u(:,2),"bo")
xtitle("Scrambled Halton: 1000 points","X1","X2")
```

The previous script produces the figure 1.3.

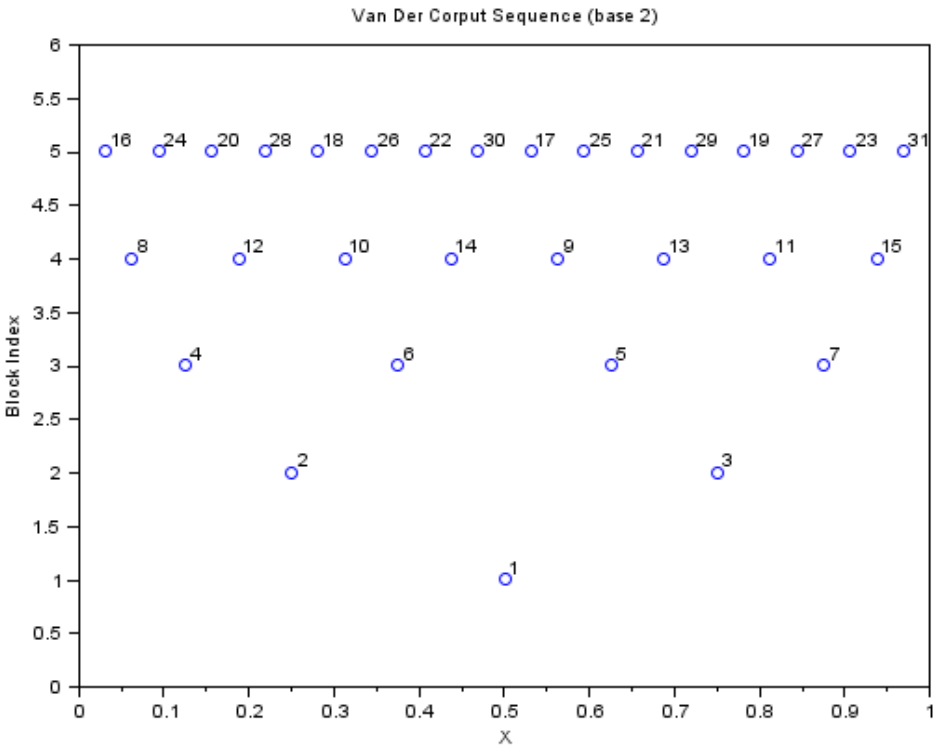

Figure 1.1: The Van Der Corput sequence in base 2.

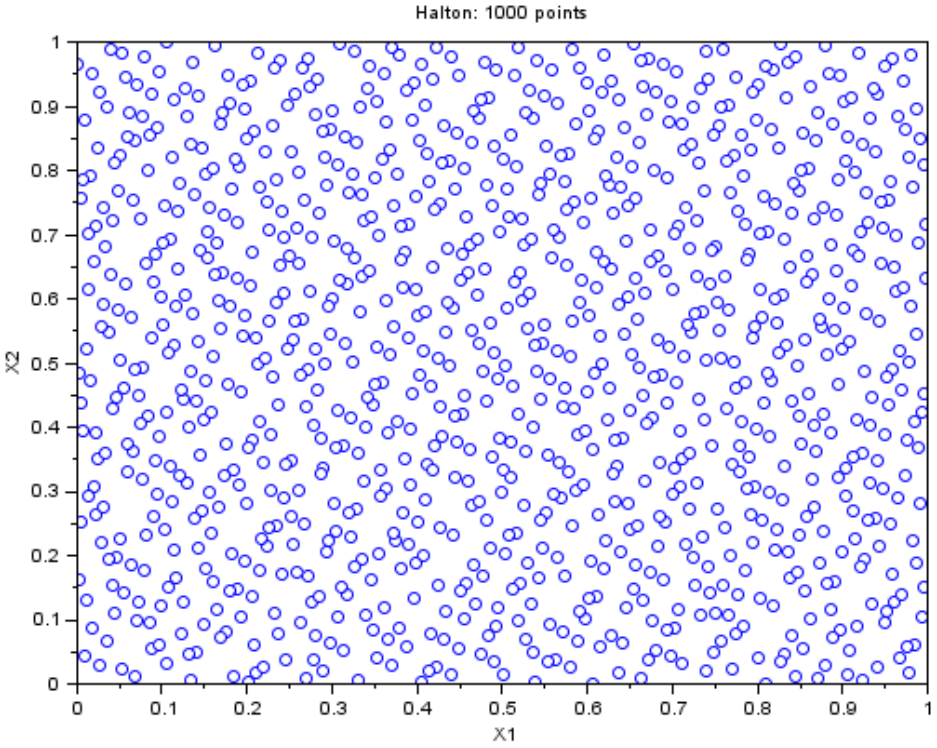

Figure 1.2: The Halton sequence - 1000 points in 2 dimensions.

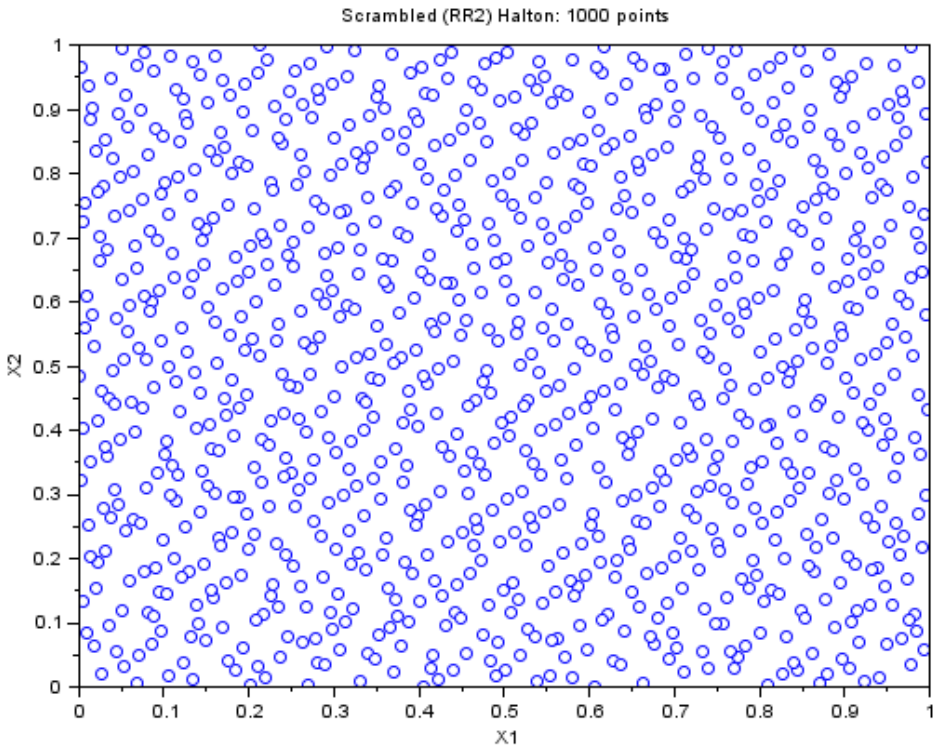

Figure 1.3: The scrambled Halton-RR2 sequence - 1000 points in 2 dimensions.

1.4 Leaped Halton Sequence

The following script produces a leaped Halton sequence. The leap parameter is automatically computed by the `lowdisc_haltonsuggest` function.

```
u=lowdisc_ldgen ( 1000 , 2 , "halton-leaped" );
scf();
plot(u(:,1),u(:,2),"bo")
xtitle("Leaped Halton: 1000 points","X1","X2")
```

The previous script produces the figure 1.4.

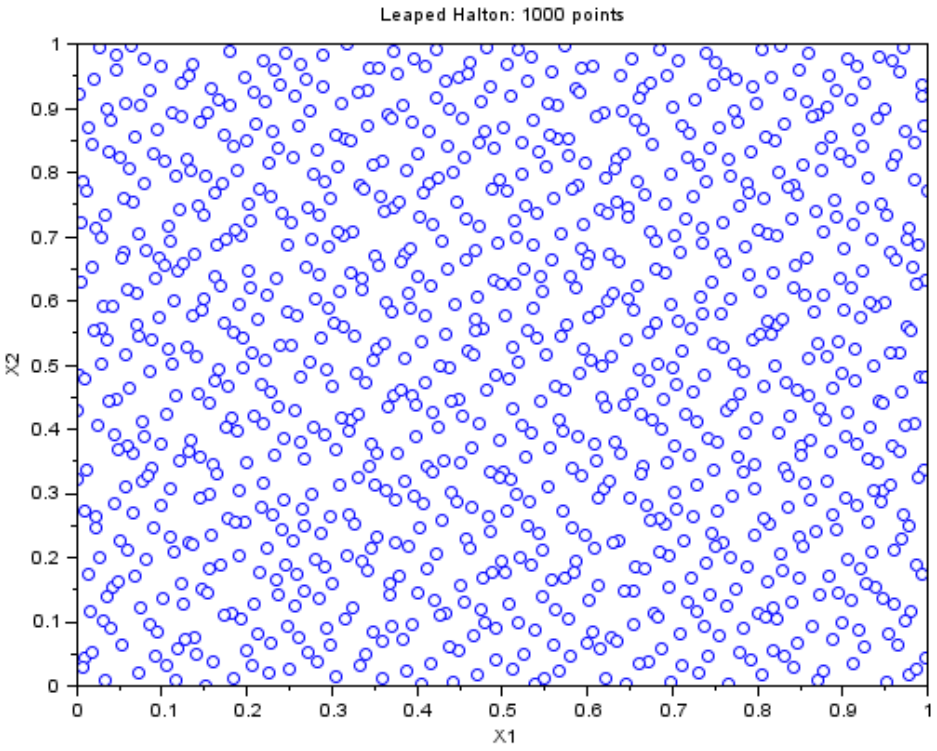

Figure 1.4: The Halton leaped sequence - 1000 points in 2 dimensions.

1.5 Reverse-Halton Sequence

```
u=lowdisc_ldgen ( 1000 , 2 , "halton-reverse" );
scf();
plot(u(:,1),u(:,2),"bo")
xtitle("Reverse-Halton: 1000 points","X1","X2")
```

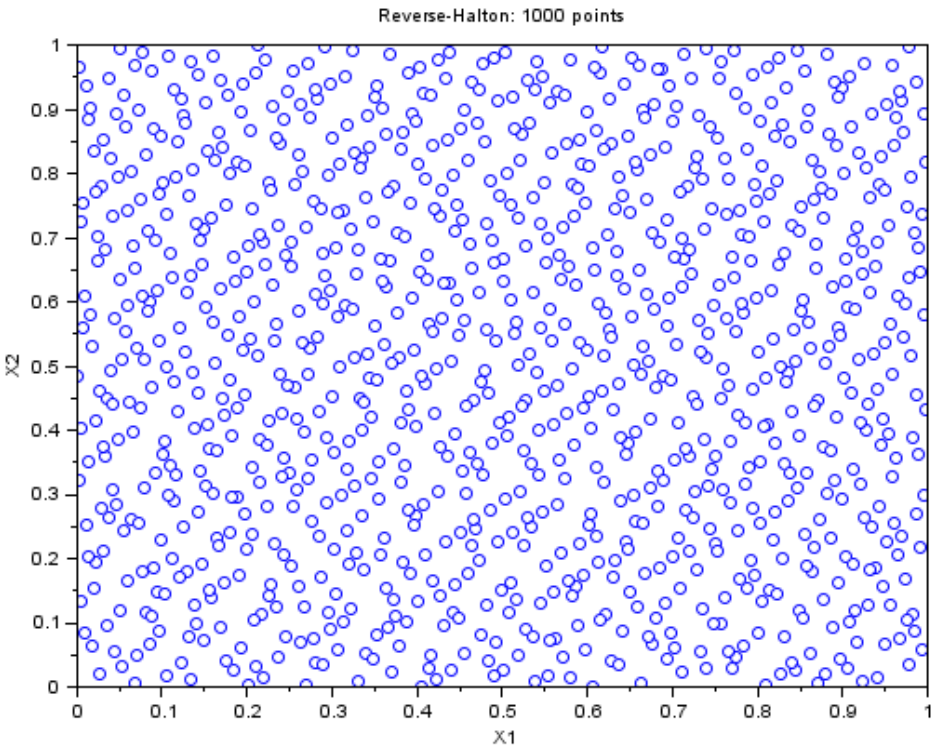

Figure 1.5: The Reverse Halton sequence - 1000 points in 2 dimensions.

The previous script produces the figure 1.5.

1.6 Sobol Sequence

```
u=lowdisc_ldgen ( 1000 , 2 , "sobol" );
scf();
plot(u(:,1),u(:,2),"bo")
xtitle("Sobol: 1000 points","X1","X2")
```

The previous script produces the figure 1.6.

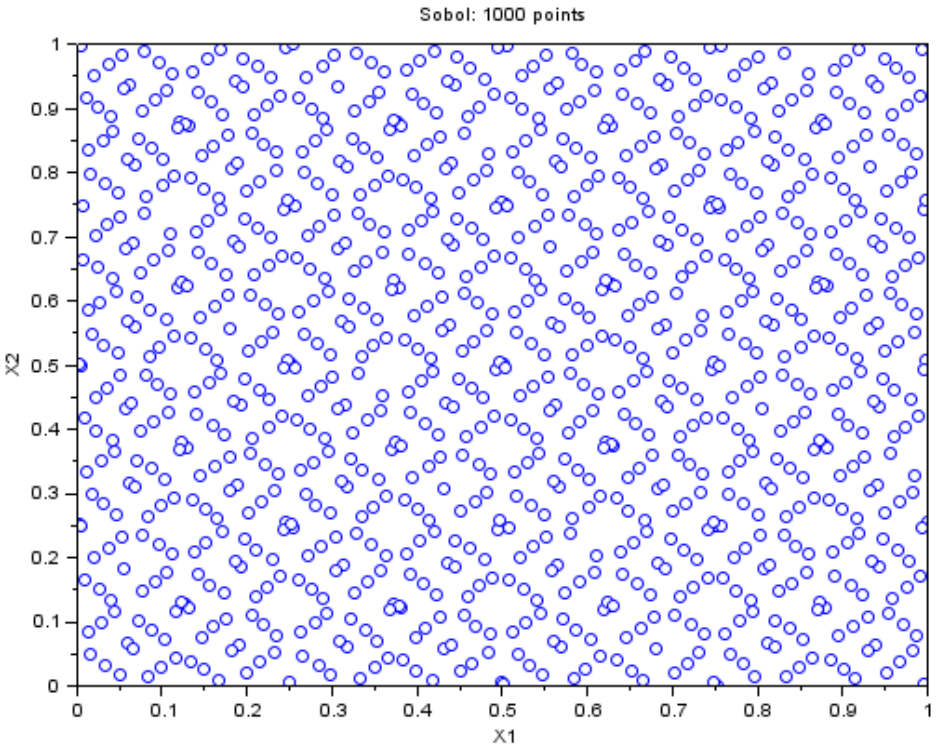

Figure 1.6: The Sobol sequence - 1000 points in 2 dimensions.

1.7 Faure Sequence

```
u=lowdisc_ldgen ( 1000 , 2 , "faure" );
scf();
plot(u(:,1),u(:,2),"bo")
xtitle("Faure: 1000 points","X1","X2")
```

The previous script produces the figure 1.7.

1.8 Niederreiter Sequence

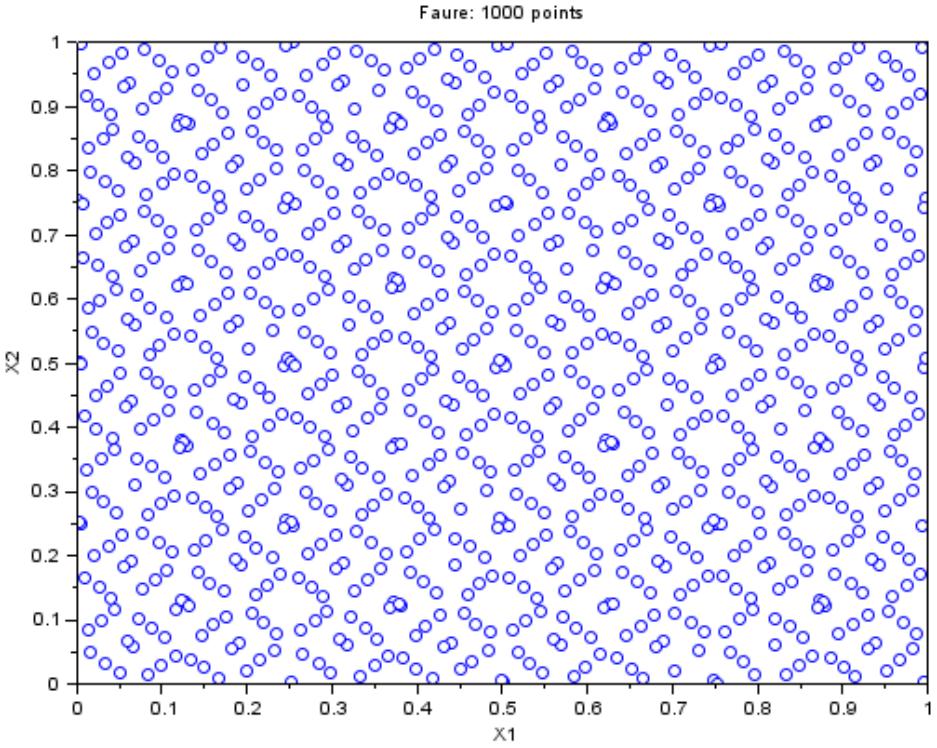

Figure 1.7: The Faure sequence - 1000 points in 2 dimensions.

```
u=lowdisc_ldgen ( 1000 , 2 , "niederreiter" );
scf();
plot(u(:,1),u(:,2),"bo")
xtitle("Niederreiter: 1000 points","X1","X2")
```

The previous script produces the figure 1.8.

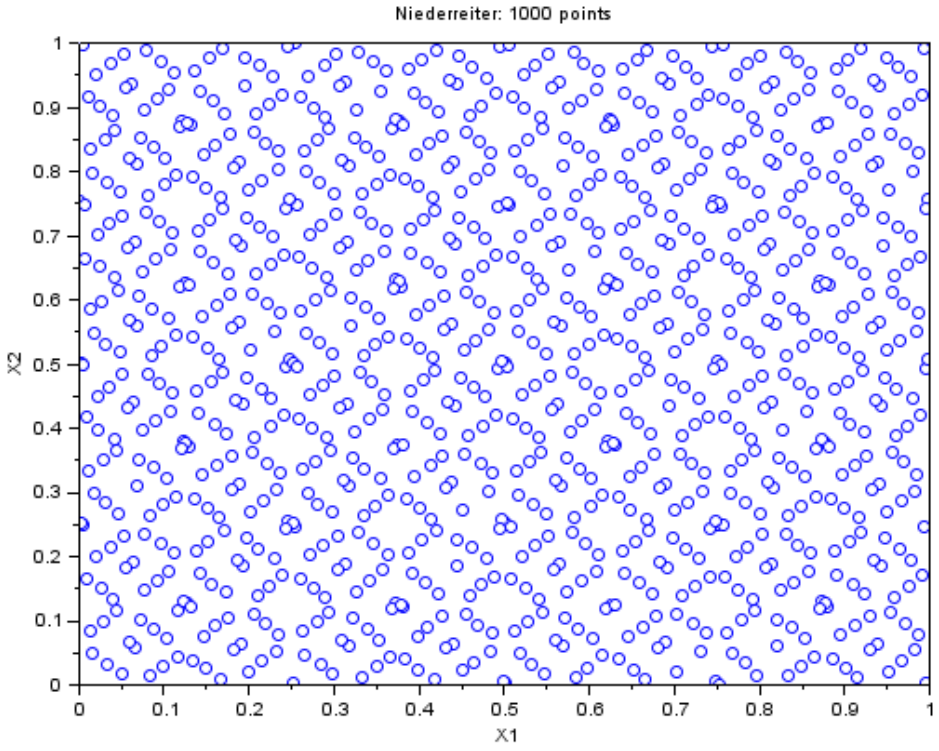

Figure 1.8: The Niederreiter sequence in base 2 - 1000 points in 2 dimensions.

1.9 Scrambled Sobol Sequence

```
//
// No Scrambling
//
lds=lowdisc_new("sobol");
lds=lowdisc_configure(lds,"-dimension",2);
lds = lowdisc_startup (lds);
[lds,u] = lowdisc_next (lds,100);
lds=lowdisc_destroy(lds);
//
// Owen Scrambling
//
```

```
lds=lowdisc_new("sobol");
lds=lowdisc_configure(lds,"-dimension",2);
lds=lowdisc_configure(lds,"-scrambling","Owen");
lds = lowdisc_startup (lds);
[lds,u0] = lowdisc_next (lds,100);
lds=lowdisc_destroy(lds);
//
// Faure-Tezuka Scrambling
//
lds=lowdisc_new("sobol");
lds=lowdisc_configure(lds,"-dimension",2);
lds=lowdisc_configure(lds,"-scrambling","Faure-Tezuka");
lds = lowdisc_startup (lds);
[lds,uFT] = lowdisc_next (lds,100);
lds=lowdisc_destroy(lds);
//
// Owen-Faure-Tezuka Scrambling
//
lds=lowdisc_new("sobol");
lds=lowdisc_configure(lds,"-dimension",2);
lds=lowdisc_configure(lds,"-scrambling","Owen-Faure-Tezuka");
lds = lowdisc_startup (lds);
[lds,uOFT] = lowdisc_next (lds,100);
lds=lowdisc_destroy(lds);
//
// Make a plot
//
scf();
subplot(2,2,1);
plot(u(:,1),u(:,2),"b.");
xtitle("Sobol","X1","X2");
subplot(2,2,2);
plot(u0(:,1),u0(:,2),"b.");
xtitle("Owen␣Scrambled␣Sobol","X1","X2");
subplot(2,2,3);
plot(uFT(:,1),uFT(:,2),"b.");
xtitle("Faure-Tezuka␣Scrambled␣Sobol","X1","X2");
subplot(2,2,4);
plot(uOFT(:,1),uOFT(:,2),"b.");
xtitle("Owen-Faure-Tezuka␣Scrambled␣Sobol","X1","X2");
```

The previous script produces the figure 1.9.

1.10 Note on 2D plots

As we have seen, in two dimensions, the Sobol, Niederreiter (base 2) and the Faure sequences are essentially the same.

```
i=(1:2^4-1)';
u1=lowdisc_ldgen ( 2^4-1 , 2 , "sobol" );
u2=lowdisc_ldgen ( 2^4-1 , 2 , "faure" );
```

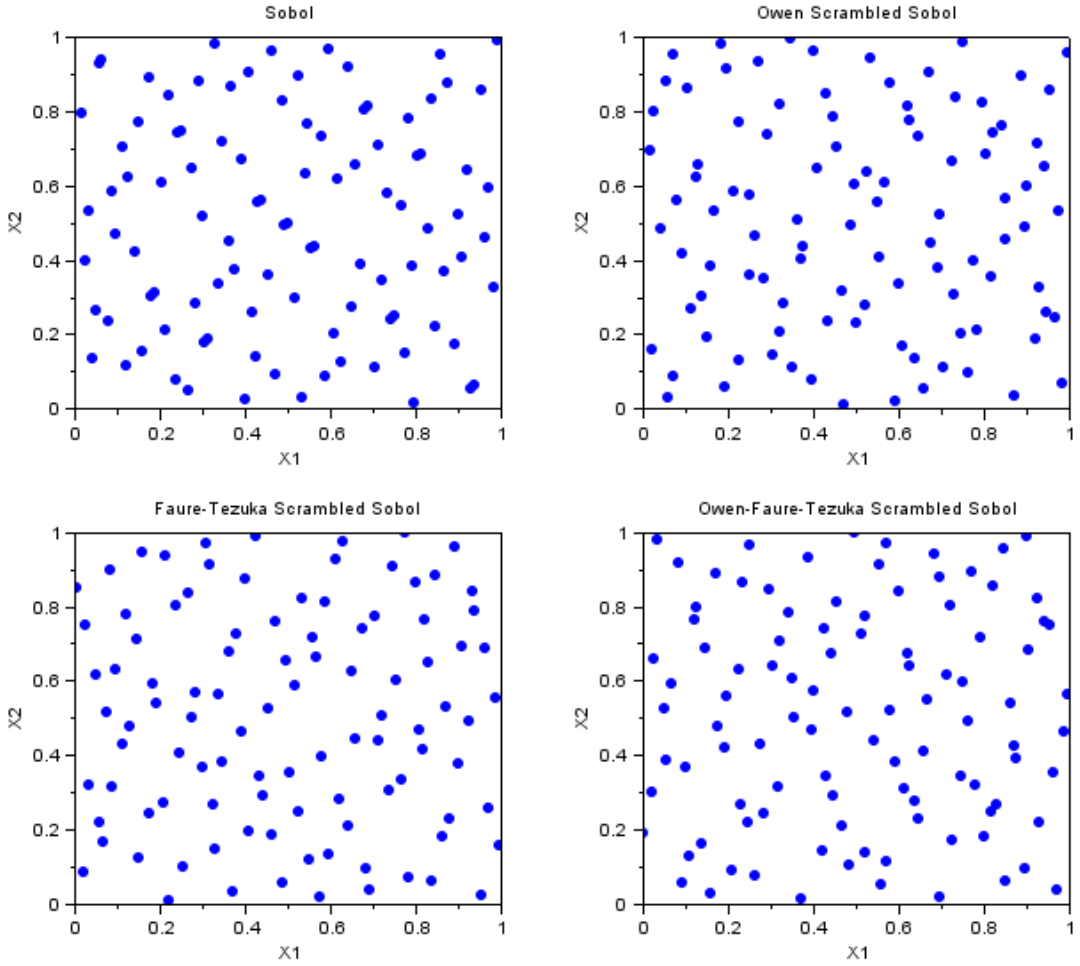

Figure 1.9: The scrambled Sobol sequence 100 points in 2 dimensions.

```
u3=lowdisc_ldgen ( 2^4-1 , 2 , "niederreiter" );
disp([i,u1,u2,u3])
```

The previous script produces the table 1.10.

	Sobol		Faure		Niederreiter	
n	x_1	x_2	x_1	x_2	x_1	x_2
1	0.5	0.5	0.5	0.5	0.5	0.5
2	0.75	0.25	0.25	0.75	0.25	0.75
3	0.25	0.75	0.75	0.25	0.75	0.25
4	0.375	0.375	0.125	0.625	0.125	0.625
5	0.875	0.875	0.625	0.125	0.625	0.125
6	0.625	0.125	0.375	0.375	0.375	0.375
7	0.125	0.625	0.875	0.875	0.875	0.875
8	0.1875	0.3125	0.0625	0.9375	0.0625	0.9375
9	0.6875	0.8125	0.5625	0.4375	0.5625	0.4375
10	0.9375	0.0625	0.3125	0.1875	0.3125	0.1875
11	0.4375	0.5625	0.8125	0.6875	0.8125	0.6875
12	0.3125	0.1875	0.1875	0.3125	0.1875	0.3125
13	0.8125	0.6875	0.6875	0.8125	0.6875	0.8125
14	0.5625	0.4375	0.4375	0.5625	0.4375	0.5625
15	0.0625	0.9375	0.9375	0.0625	0.9375	0.0625

Figure 1.10: The first 15 points of the Sobol, Faure and Niederreiter (base 2) sequences in 2 dimensions.

The points are coming by blocks which size is a power of 2. A part of the reason is because all these sequences in two dimensions are based on the first prime, which is 2.

The Niederreiter (base 2) and Faure sequences produce exactly the same points, in the same order. The Sobol points are also the same, but come in a different order. For example, the point n=15 in Sobol sequence is the same as point n=8 in Niederreiter (base 2) sequence.

1.11 In 3D

Since the 2D Niederreiter, Sobol and Faure sequences are the same, we consider 3D sequences.

```
u=lowdisc_ldgen ( 1000 , 3 , "niederreiter" );
scf();
lowdisc_proj2d(u)
subplot(2,2,3)
xstring(0.5,0.5,"Niederreiter:␣1000␣points")

u=lowdisc_ldgen ( 1000 , 3 , "sobol" );
scf();
lowdisc_proj2d(u)
subplot(2,2,3)
xstring(0.5,0.5,"Sobol:␣1000␣points")
```

```
u=lowdisc_ldgen ( 1000 , 3 , "faure" );
scf();
lowdisc_proj2d(u)
subplot(2,2,3)
xstring(0.5,0.5,"Faure: 1000 points")
```

The previous script produces the figures 1.11, 1.12 and 1.13.

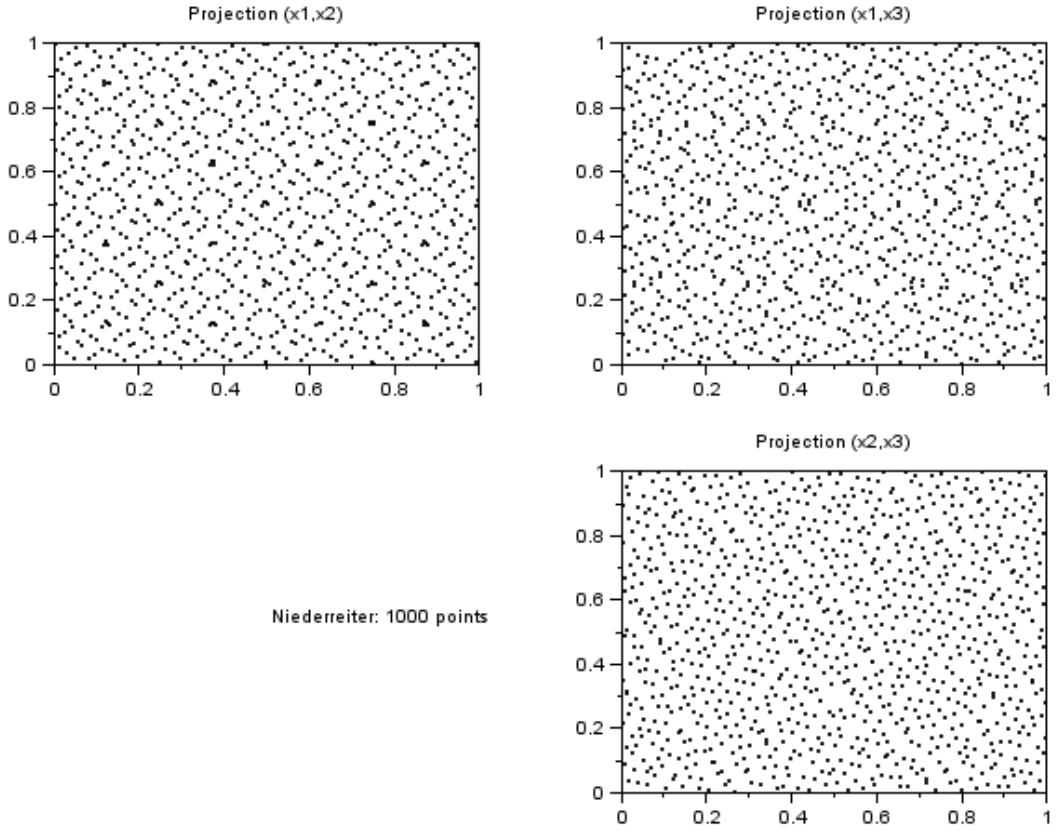

Figure 1.11: Niederreiter sequence in base 2 - 1000 points in 3 dimensions.

We see that the projection (x_1, x_2) is the same for Sobol and Niederreiter (base 2), as was seen for 2D sequences. But, the projections (x_1, x_3) and (x_2, x_3) of Sobol and Niederreiter are different. Moreover, all projections of Faure sequence are now different from the other sequences. Furthermore, the projection (x_1, x_2) of Faure is different from the Faure 2D sequence.

Indeed, the Faure sequence uses a basis which is the smallest prime number larger than the number of dimensions. In 3 dimensions, the basis used by Faure is 3. Since Sobol and Niederreiter (base 2) use the base 2, this explains why the sequences cannot be equal anymore.

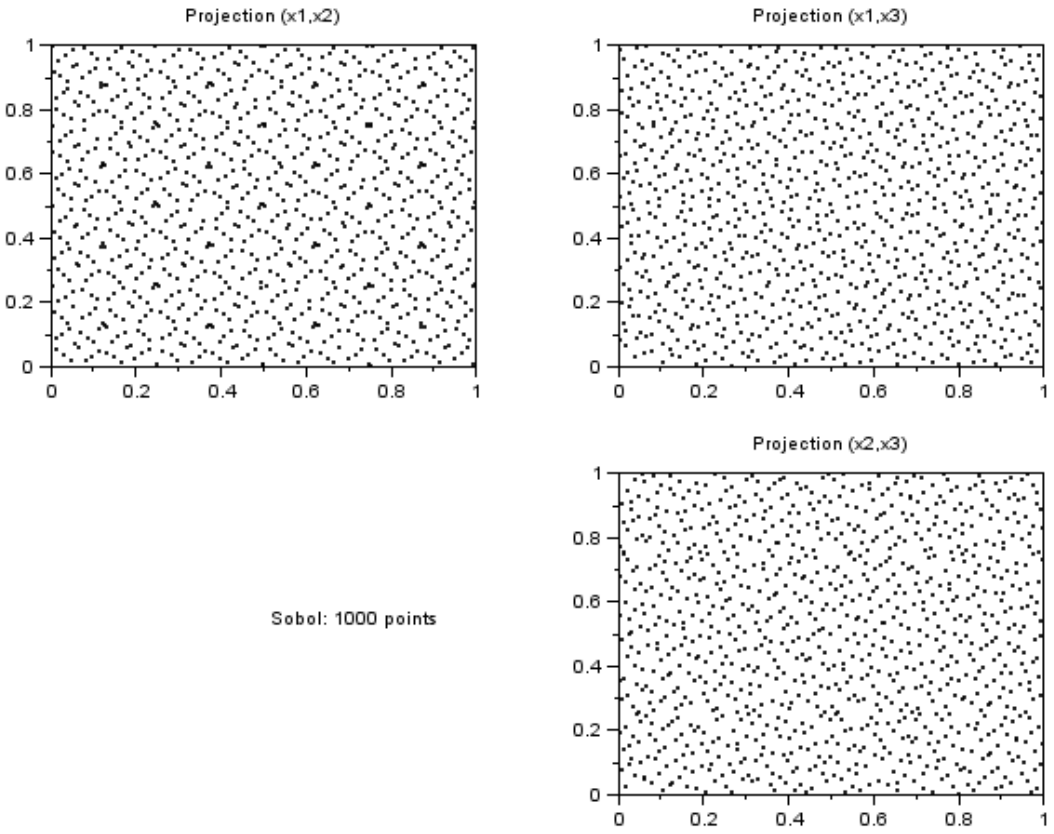

Figure 1.12: Sobol sequence - 1000 points in 3 dimensions.

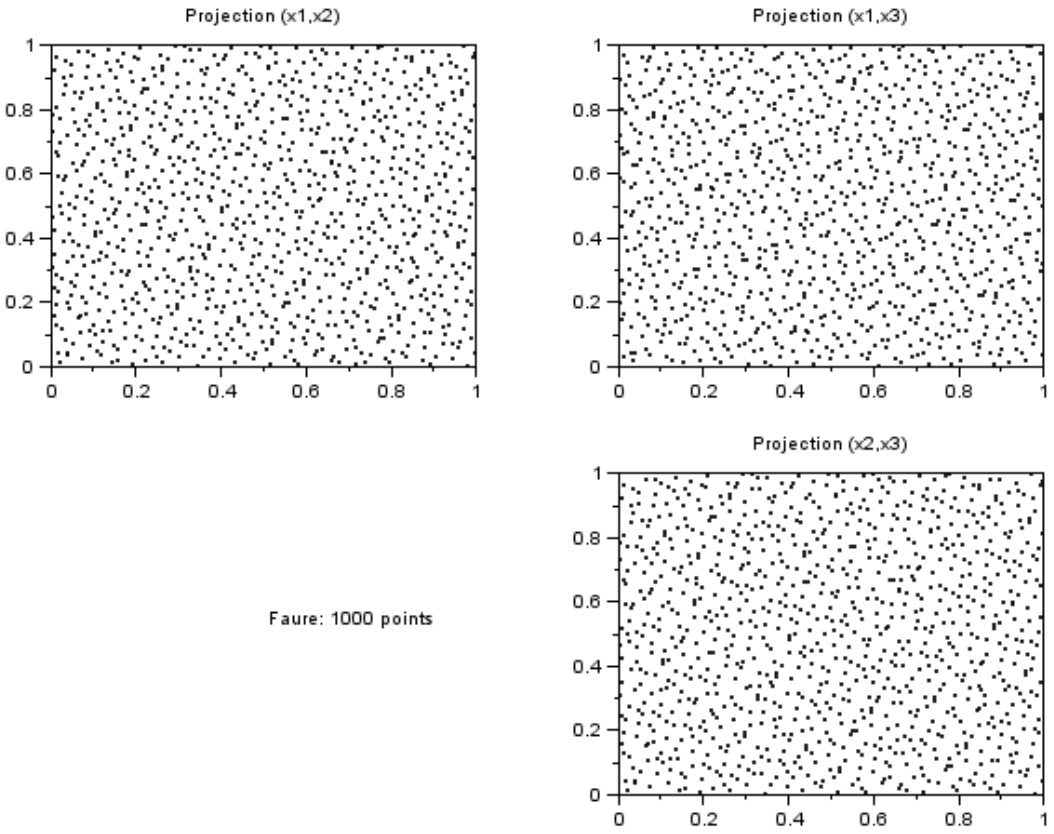

Figure 1.13: Faure sequence - 1000 points in 3 dimensions.

Chapter 2

Nets

2.1 Definitions

The following definition introduces the elementary intervals in one dimensions, which are particular intervals in $[0, 1)$.

Definition 2.1.1. *(*Elementary interval in one dimension.*) Let b be the base, with $b \geq 2$. An elementary interval in base b is an interval $[a/b^d, (a+1)/b^d)$ with integers a and d such that $d \geq 0$ and $0 \leq a < b^d$.*

Notice that the elementary interval are closed on the left and open on the right.
The following definition introduces the multi-dimensional intervals.

Definition 2.1.2. *(*Multi-dimensional interval.*) The set P is an interval of $[0, 1)^s$ if:*

$$P = \prod_{j=1}^{s} [\alpha_j, \beta_j), 0 \leq \alpha_j \leq \beta_j \leq 1,$$

for j=1,2,...,s, where $\alpha_j, \beta_j \in [0, 1)$ and $\alpha_j \leq \beta_j$.

With low discrepancy sequences, we are interested in specific multi-dimensional intervals, associated with a given base b.

Definition 2.1.3. *(*Elementary interval in base b in s dimensions.*) Let b be the base, with $b \geq 2$. An elementary interval in base b of the hypercube $[0, 1]^s$ is an interval*

$$\Pi_{i=1}^{s} [a_i/b^{d_i}, (a_i + 1)/b^{d_i})$$

with integers a_i and d_i such that $d_i \geq 0$ and $0 \leq a_i < b^{d_i}$, for $i = 1, 2, ..., s$.

Corollary 2.1.4. *(*Volume of an elementary interval in base b.*) The volume of an elementary interval in base b in s dimensions is*

$$\frac{1}{b^{d_1+d_2+...+d_s}}.$$

The following script computes all two dimensional elementary intervals in base 2 with area $1/2^3 = 1/8$, and creates the figure 2.1.

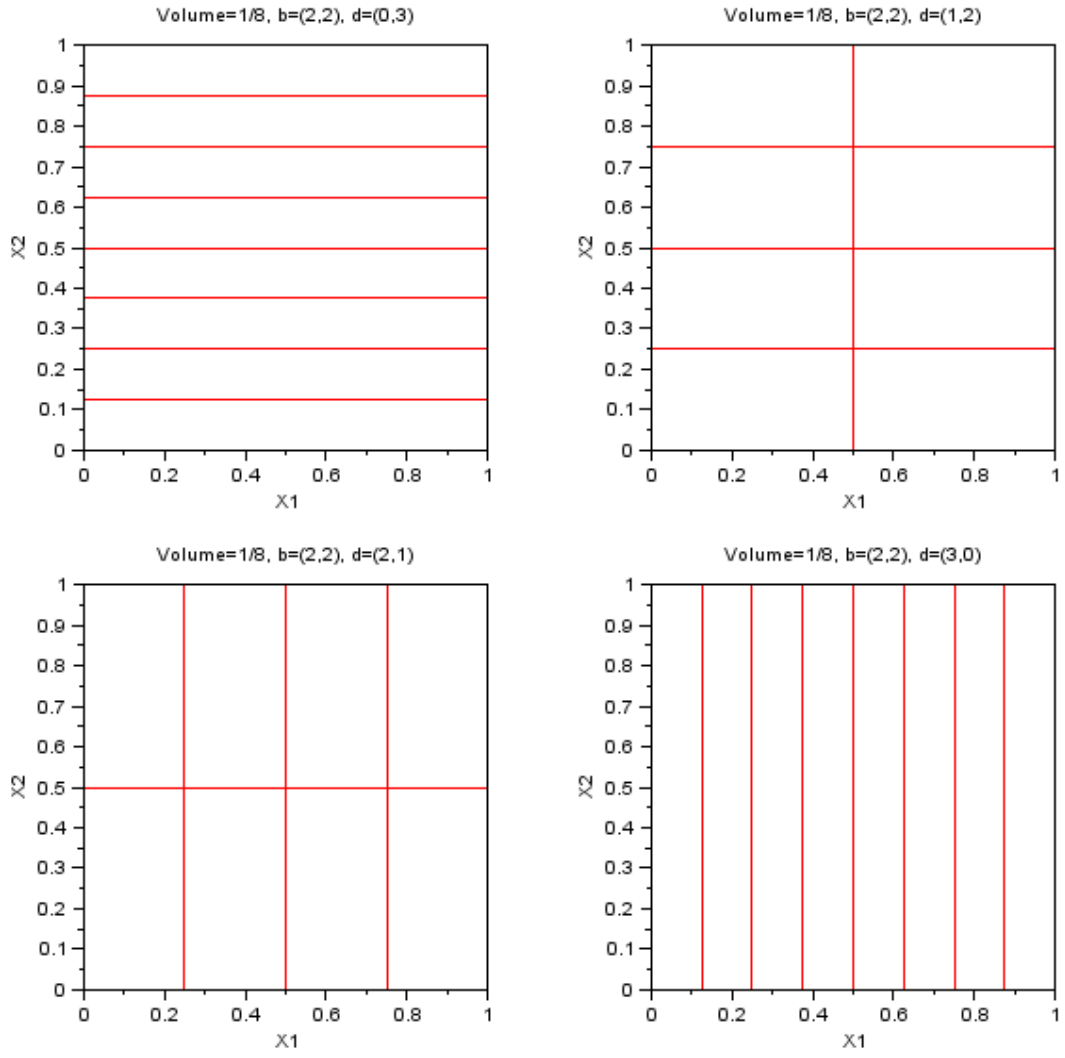

Figure 2.1: Elementary intervals in base 2 with volume 1/8.

```
scf();
lowdisc_plotbmbox(2,3)
```

Definition 2.1.5. *((t,m,s)-net in base b.) Let $b \geq 2$, $s \geq 1$ and $0 \leq t \leq m$ be integers. Then a point set with b^m points in $[0,1)^s$ is a (t,m,s)-net in base b if every elementary interval in base b with volume $1/b^{m-t}$ contains exactly b^t points.*

The previous definition is consistent with the fact that the volume of the elementary interval is equal to b^t/b^m, i.e. the fraction of points in the elementary interval.

A smaller base b should be favored because it requires b^m points for the property to show up. Also, when t decreases, the volume of the associated elementary intervals decreases, i.e. even small elementary intervals contain the right number of points. In other words, a smaller t parameter provides a greater uniformity.

At best, we have $t = 0$, for which a point set of b^m points is associated with elementary intervals with volume $1/b^m$, with one point by elementary interval.

Corollary 2.1.6. *((0,m,s)-net in base b.) Let $b \geq 2$, $s \geq 1$ be integers. Then a point set with b^m points in $[0,1)^s$ is a (0,m,s)-net in base b if every elementary interval in base b with volume $1/b^m$ contains exactly one point.*

Definition 2.1.7. *((t,s)-sequence in base b.) Let $b \geq 2$, $s \geq 1$ and $t \geq 0$ be integers. Then the infinite sequence x_0, x_1, ... of points in $[0,1)^s$ is a (t,s) sequence in base b if, for all $k \geq 0$ and $m \geq t$, the point set*

$$x_{kb^m}, x_{kb^m+1}, ..., x_{(k+1)b^m-1},$$

which has b^m points, is a (t,m,s)-net in base b.

2.2 Theorems

In this section, we present the main properties of low discrepancy sequences in terms of (t,m,s)-nets.

- Van der Corput in base b are (0,1)-nets in base b.
- Halton sequences are not (t,s)-sequences: the base b is not the same for all dimensions. However, a generalized (t,s)-sequence could be defined, with an extended definition taking into account for different bases.
- Sobol sequence is a (t,s)-sequence in base 2 with t=O(slog(s)). Sobol sequence use the same base b=2 for all dimensions.
- The Faure sequences are (0,s)-sequences in base b. Therefore, the Faure sequences are (0,m,s)-nets in base b, for any $m \geq 0$. Therefore, Faure sequences achieve the lowest possible value of t, but with increasing b (the smallest prime larger than s).
- Niederreiter sequences are (t,s)-sequences defined for any base b, where b is a power of a prime.

The table 2.2 presents the quality parameters t of two (t,s)-sequences : the Sobol sequence, and the Niederreiter (base 2) sequence.

s	1	2	3	4	5	6	7	8	9	10	11	12	13
Niederreiter (b=2)	0	0	1	3	5	8	11	14	18	22	26	30	34
Sobol	0	0	1	3	5	8	11	15	19	23	27	31	35

s	14	15	16	17	18	19	20
Niederreiter (b=2)	38	43	48	53	58	63	68
Sobol	40	45	50	55	60	65	71

s	21	22	23	24	25	26	27	28	29	30
Niederreiter (b=2)	73	78	83	89	95	101	107	113	119	125
Sobol	77	83	89	95	101	107	113	119	125	131

s	31	32	33	34	35	36	37	38	39	40
Sobol	137	143	149	155	161	167	173	180	187	194

Figure 2.2: The quality parameter t for the Sobol and Niederreiter sequences.

2.3 Discrepancy

Definition 2.3.1. *(*Star intervals.*) The $I^\star$ set is the set of origin-anchored intervals in $[0,1]^s$:*

$$I^\star = \left\{ P = \prod_{j=1}^{s} [0, \beta_j), 0 \leq \beta_j \leq 1 \right\}.$$

Definition 2.3.2. *(*Number of points and volume of an interval.*) Let x be a sequence of more than n points in $[0,1)^s$. Then $x(n)$ represents the set of the first n points of the sequence x. Let P be an interval of $[0,1)^s$. Therefore, the number $A(P, x(n))$ is the number of points from $x(n)$ in the interval P. Moreover, let $\lambda(P)$ be the volume of P:*

$$\lambda(P) = \lambda(P) \prod_{j=1}^{s} (\beta_j - \alpha_j)$$

Definition 2.3.3. *(*Star discrepancy.*) Let x be a sequence of more than n points in $[0,1)^s$. The star discrepancy $D^\star(x)$ measures the lack of uniformity of the first n points:*

$$D_n^\star(x) = \sup_{P \in I^\star} \left| \frac{1}{n} A(P, x(n)) - \lambda(P) \right|.$$

2.4 Estimate of the mean by a low discrepancy sequence

Theorem 2.4.1. *(*Koksma-Hlawka.*) Let f from $[0,1)^s$ to $\mathbb{R}$ be a function where $V(f)$, the variation in the sense of Hardy and Krause, is bounded, therefore, for any sequence of n points $x(n)$, we have*

$$\left| \frac{1}{n} \sum_{i=1}^{n} f\left(x^{(i)}\right) - \int_{[0,1]^s} f(x) dx \right| \leq V(f) D_n^\star(x).$$

Theorem 2.4.2. *(*Niederreiter.*) For any (t,s)-sequence in base b, we have*

$$D_n^\star(x) \leq C(b,s,t) n^{-1} (\log n)^s + O\left(n^{-1} (\log n)^{s-1}\right)$$

s	2	3	4	5	6	7	8	9	10
Halton	0.65	0.81	1.25	2.62	6.13	17.3	52.9	185.5	771.5
Sobol	0.26	0.25	0.541	0.833	1.6	2.6	7.6	19.6	45.0
Faure	0.26	0.13	0.099	0.025	0.019	0.0041	0.0089	2.1×10^{-3}	4.2×10^{-4}
Nieder.	0.26	0.13	0.086	0.025	0.019	0.0041	0.0030	6.1×10^{-4}	4.2×10^{-4}

Figure 2.3: The constant $C(b, s, t)$ for various low discrepancy sequences.

In other words, any (t,s)-sequence is a low discrepancy sequence.

In the previous inequality, the dominating term, when n is large, is $1/n$. However, for moderate n, the factor $\log(n)^s$ must be taken into account. Indeed, we the function $\log(n)^s/n$ increases for n lower than $\exp(s)$. In other words, for increasing values of s, the number of points n required to get into the asymptotic discrepancy rate is larger and larger.

- Sobol sequence : $C(b, s, t)$ grows to infinity, when s increases.
- Faure sequences : $C(b, s, t)$ grows to zero, when s increases.
- Niederreiter sequences : $C(b, s, t)$ grows to zero, when s increases.

The table 2.3 presents the constant $C(b, s, t)$ for several sequences.

2.5 Halton sequence

In this section, we show how the Halton sequence fill elementary intervals for which the i-th side has length equal to a power of $1/b_i$, where b_i is the prime number associated with the i-th coordinate.

In the following script, we consider the first $2^2 \times 3 = 12$ points of the Halton sequence in two dimensions. Then we consider elementary intervals of area $1/12$, where the sides of the intervals have length equal to $1/2^2$ and $1/3$.

```
// Get the points 0,1,...,11 (insert zero)
u=lowdisc_ldgen ( 11 , 2 , "halton" );
u = [0,0;u];
//
scf();
subplot(1,2,1);
plot(u(:,1),u(:,2),"bo")
lowdisc_plotelembox([2 3],[2 1])
xtitle("Halton, Volume=1/12, 12 Points");
h = gca();
h.isoview="off";
//
subplot(1,2,2);
plot(u(:,1),u(:,2),"bo")
lowdisc_plotelembox([2 3],[1 1])
xtitle("Halton, Volume=1/6, 12 Points");
```

```
h = gca();
h.isoview="off";
```

The previous script produces the figure 2.4.

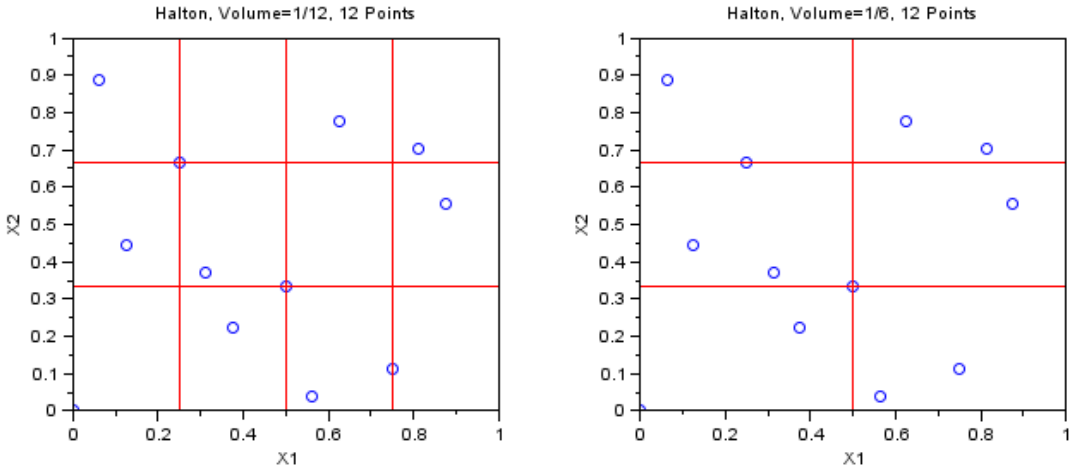

Figure 2.4: The Halton sequence : 12 points with elementary intervals of volume 1/6.

We see that there are exactly $6/12 = 2$ points in each interval.

2.6 Faure sequence

In dimension s=2, the Faure sequence uses the base 2. Moreover, the Faure sequence is a (0,2)-sequence in base 2 (i.e. the quality parameter t is zero). We choose m=4, and check that Faure is a (0,4,2)-net in base 2. Hence, we have to consider $b^m = 2^4 = 16$ points and check that there are $b^t = 1$ points for any elementary interval of area $1/b^m = 1/16$.

In the following script, we consider $2^4 = 16$ Faure points and plot elementary intervals with area $1/2^4 = 1/16$. We want to check that there is 1 point by elementary interval.

```
u=lowdisc_ldgen ( 2^4-1 , 2 , "faure" );
u = [0,0;u];
scf();
lowdisc_plotbmbox(2,4,u);
```

The previous script produces the figure 2.5.

In the following script, we consider $2^4 = 16$ Faure points in two dimensions and plot elementary intervals with area $1/2^3 = 1/8$: there are 2 points by box.

```
u=lowdisc_ldgen ( 2^4-1 , 2 , "faure" );
u = [0,0;u];
scf();
lowdisc_plotbmbox(2,3,u);
```

The previous script produces the figure 2.6.

In the following script, we consider 16 Faure points in two dimensions and plot elementary intervals with volume 1/8.

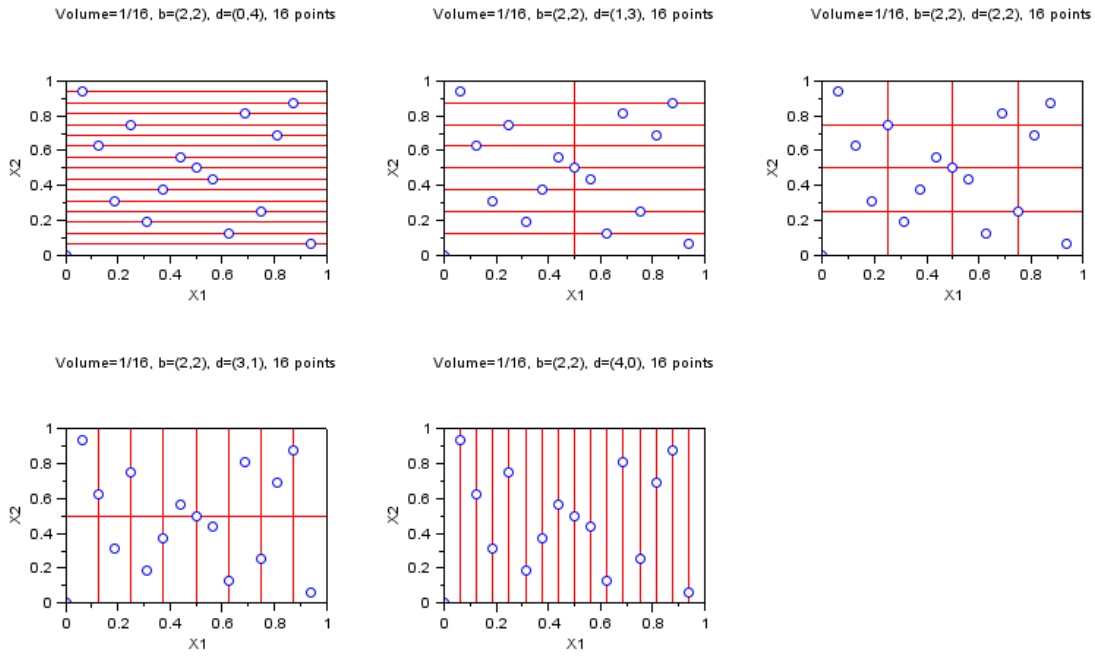

Figure 2.5: The Faure sequence : 16 points with elementary intervals of volume 1/16.

```
// Get the points 0,1,...,15 (insert zero)
u=lowdisc_ldgen ( 2^4-1 , 2 , "faure" );
u = [0,0;u];
scf();
//
subplot(2,2,1);
plot(u(1:8,1),u(1:8,2),"bo")
plot(u(9:16,1),u(9:16,2),"g*")
legend(["Points␣1-8","Points␣9-16"]);
lowdisc_plotelembox(2,[3 0]);
xtitle("Faure,␣Volume=1/8,␣16␣Points");
h = gca();
h.isoview="off";
//
subplot(2,2,2);
plot(u(1:8,1),u(1:8,2),"bo")
plot(u(9:16,1),u(9:16,2),"g*")
legend(["Points␣1-8","Points␣9-16"]);
lowdisc_plotelembox(2,[2 1]);
xtitle("Faure,␣Volume=1/8,␣16␣Points");
h = gca();
h.isoview="off";
//
subplot(2,2,3);
plot(u(1:8,1),u(1:8,2),"bo")
```

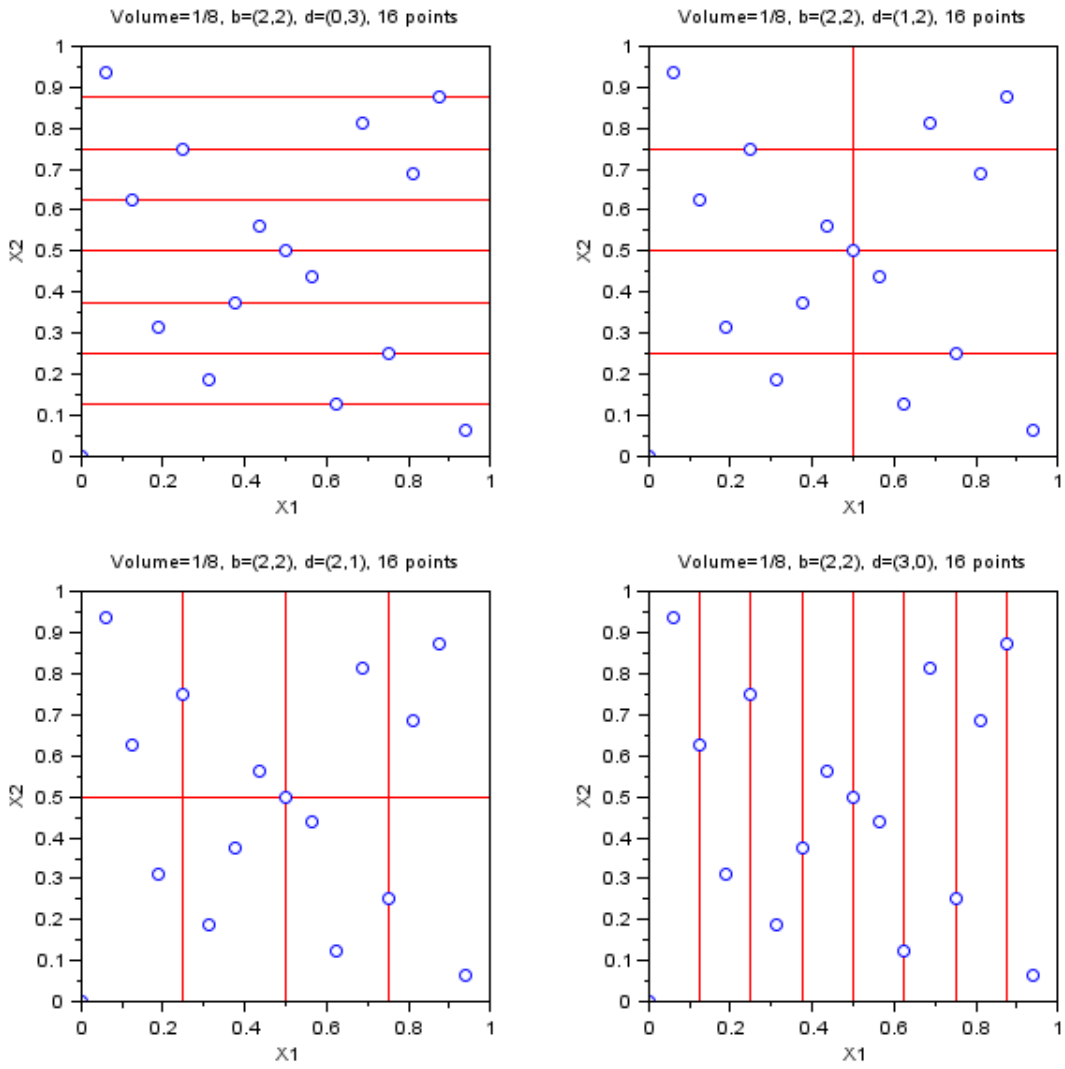

Figure 2.6: The Faure sequence : 16 points with elementary intervals of volume 1/8.

```
plot(u(9:16,1),u(9:16,2),"g*")
legend(["Points 1-8","Points 9-16"]);
lowdisc_plotelembox(2,[0 3]);
xtitle("Faure, Volume=1/8, 16 Points");
h = gca();
h.isoview="off";
//
subplot(2,2,4);
plot(u(1:8,1),u(1:8,2),"bo")
plot(u(9:16,1),u(9:16,2),"g*")
legend(["Points 1-8","Points 9-16"]);
lowdisc_plotelembox(2,[1 2]);
xtitle("Faure, Volume=1/8, 16 Points");
h = gca();
h.isoview="off";
```

The previous script produces the figure 2.7. We check that there are two points by interval. Moreover, the first 8 points fill all intervals (1 point by interval), while the points from 9 to 16 again fill the intervals.

2.7 Niederreiter base 2 sequence

In the following script, we consider $2^5 = 32$ Niederreiter (base 2) points in two dimensions and plot elementary intervals with area 2^4: there are 2 points by elementary interval. Indeed, Niederreiter is a (0,5,2)-net in base 2

```
u=lowdisc_ldgen ( 2^5-1 , 2 , "niederreiter" );
u = [0,0;u];
scf();
lowdisc_plotbmbox(2,4,u);
```

The previous script produces the figure 2.8.

In the following script, we consider $2^5 = 32$ Niederreiter (base 2) points in two dimensions and plot elementary intervals with area $2^3 = 8$: there are 4 points by elementary interval. Indeed, Niederreiter is a (0,5,2)-net in base 2

```
u=lowdisc_ldgen ( 2^5-1 , 2 , "niederreiter" );
u = [0,0;u];
scf();
lowdisc_plotbmbox(2,3,u);
```

The previous script produces the figure 2.9.

2.8 Sobol sequence

Sobol is a (0,4,2)-net in base 2. To check this, we consider $2^4 = 16$ Sobol points in two dimensions, and plot elementary intervals with area 1/16: there is 1 point by elementary interval. To plot the elementary intervals, we use 2^2 subdivisions for X_1, and 2^2 subdivisions for X_2.

```
u=lowdisc_ldgen ( 2^4-1 , 2 , "sobol" , %t );
```

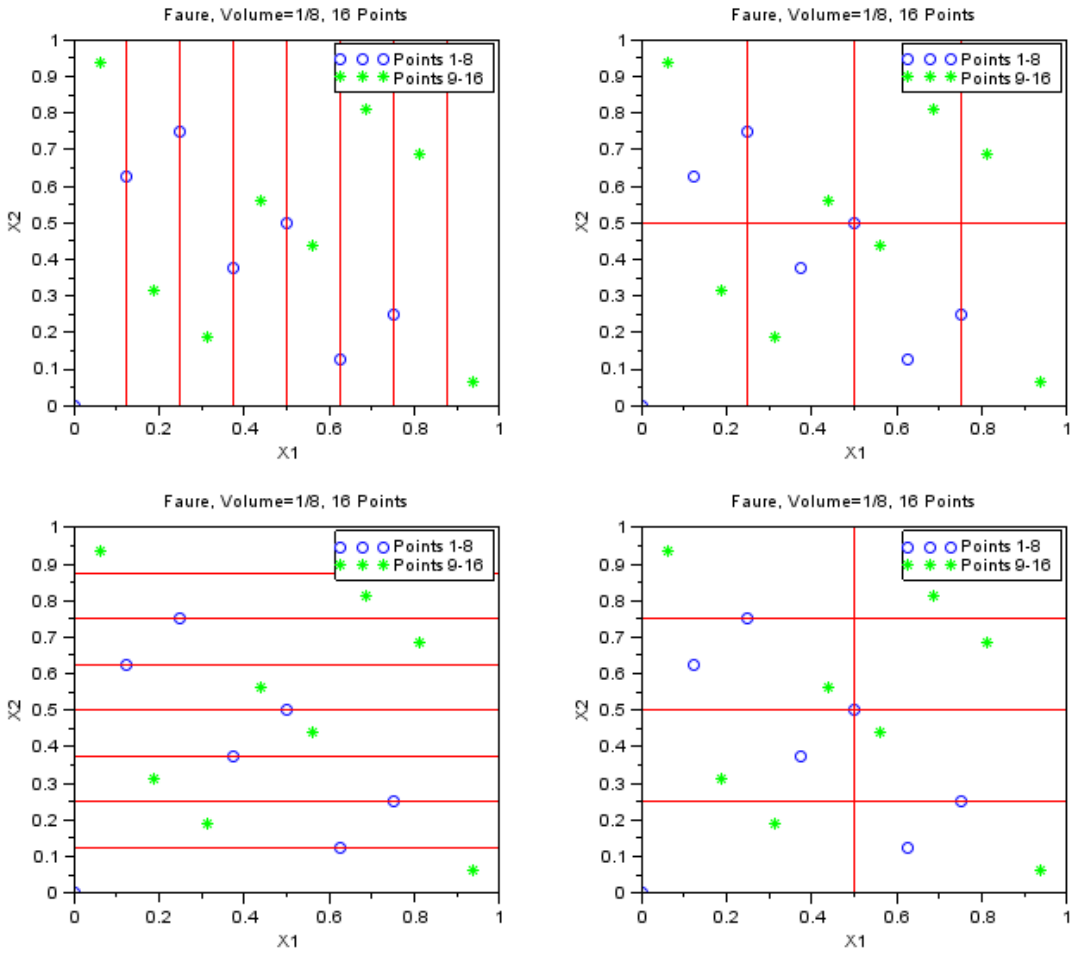

Figure 2.7: The Faure sequence : 16 points with elementary intervals of volume 1/8.

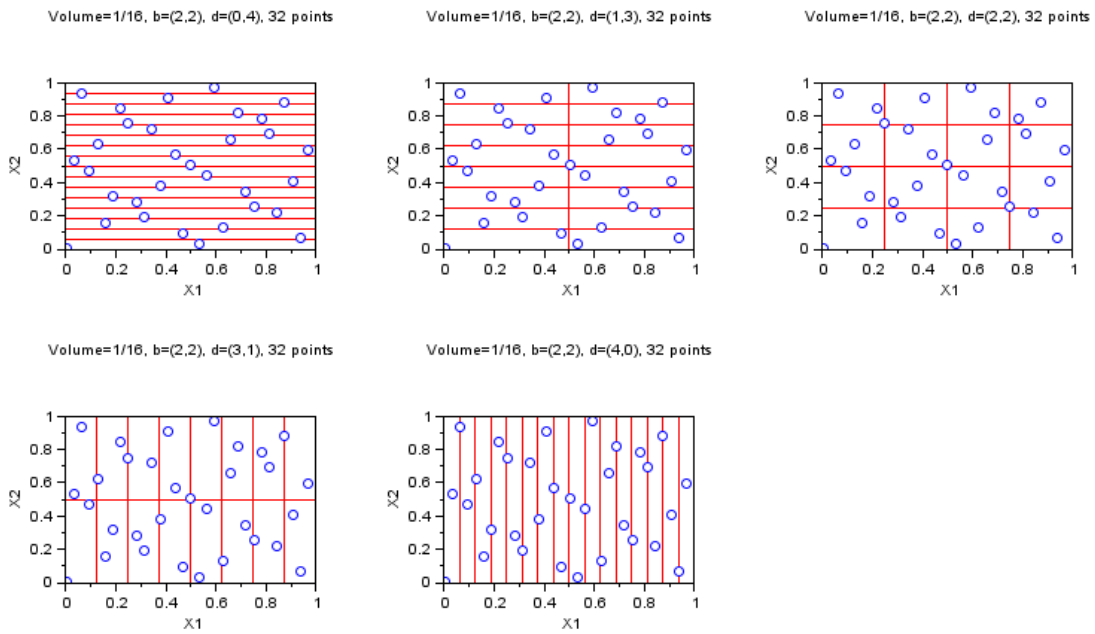

Figure 2.8: The Niederreiter sequence : 32 points in 2 dimensions, with elementary intervals of area 1/16.

```
u = [0,0;u];
h = scf();
lowdisc_plotbmbox(2,4,u);
```

The previous script produces the figure 2.10.

Sobol is a (0,5,2)-net in base 2. In order to check this, we consider $2^6 = 64$ Sobol points in two dimensions and plot elementary intervals with area 1/32: there are two points by elementary interval. The elementary intervals tse 2^3 subdivisions for X_1 and 2^2 subdivisions for X_2.

```
u=lowdisc_ldgen ( 2^6-1 , 2 , "sobol" );
u = [0,0;u];
h = scf();
plot(u(1:2^5,1),u(1:2^5,2),"bo")
plot(u(2^5+1:2^6,1),u(2^5+1:2^6,2),"g*")
legend(["Points 1-32","Points 33-64"]);
lowdisc_plotelembox(2,[3 2]);
xtitle("Sobol, Volume=1/32, 64 Points");
```

The previous script produces the figure 2.11.

2.9 Leaped sequences and (t,m,s)-nets

A common practice is to use a leaped sequence to improve the correlations in high dimensions. However, we may notice that a leaped sequence may lead to a set which is not a (t,m,s)-net anymore.

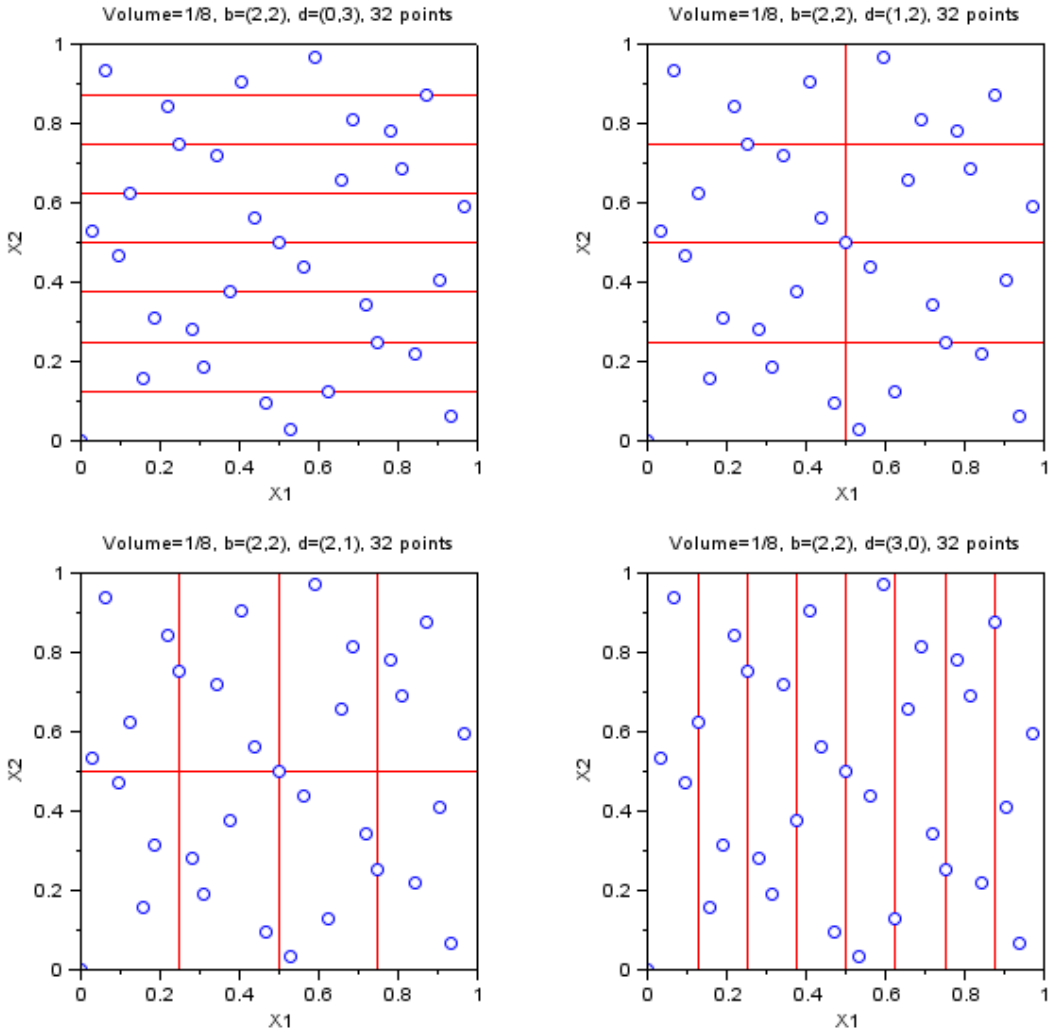

Figure 2.9: The Niederreiter sequence : 32 points in 2 dimensions, with elementary intervals of area 1/8.

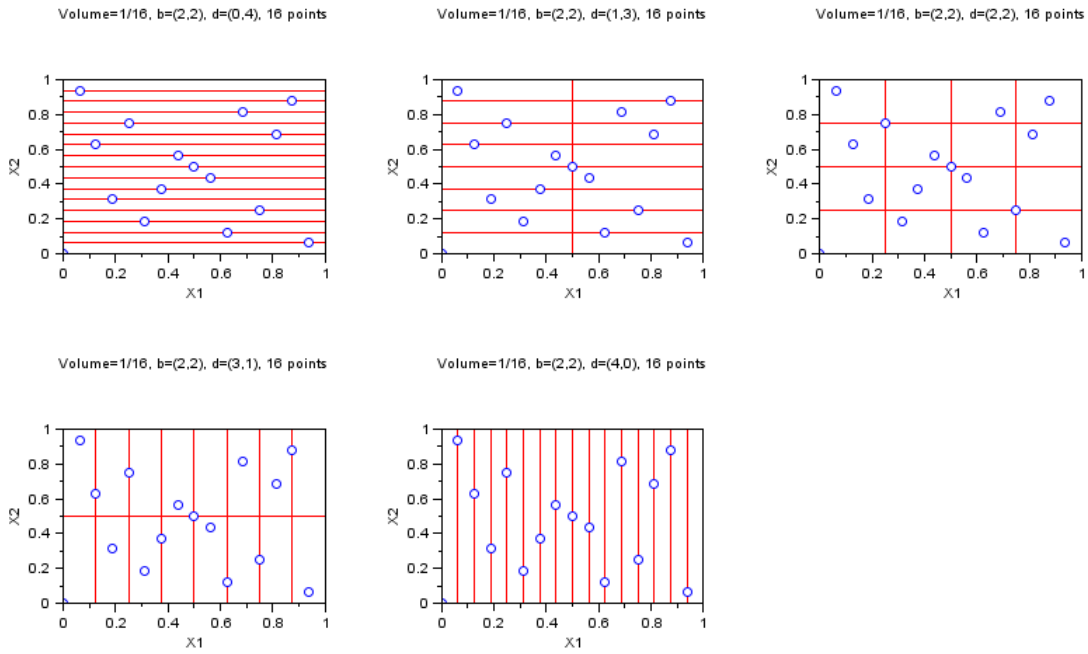

Figure 2.10: The Sobol sequence : 16 points in 2 dimensions, with elementary intervals of area 1/16.

In [8] (p. 274), Kocis and Whiten suggest to use leaped sequences, as presented in the figure 2.12. In such a Halton leaped sequence, we use every (L+1)-th vector, subject to the condition that $L+1$ is a prime different from all bases. Hence, when $L=0$, all elements are considered while when $L=1$, only one element every two numbers is used.

Kocis and Whiten use test functions with the goal of finding the leaps which minimizes the integration error. They do this experiment with s in the range from 1 to 400, a number of points from 10 to 10^5 and a leap parameter in the range from 3 to 3000. They find that the best values for $L+1$ are 31, 61, 149, 409, and 1949. This suggests to use L+1=409 for dimensions less than 400.

For example, let's see the first 12 points of a leaped Halton sequence in two dimensions, and consider elementary intervals with area 1/12.

```
u=lowdisc_ldgen ( 11 , 2 , "halton" , %f );
u = [0,0;u];
//
scf();
subplot(1,2,1);
plot(u(:,1),u(:,2),"bo")
lowdisc_plotelembox([2 3],[2 1])
xtitle("Halton, Volume=1/12, 12 Points");
h = gca();
h.isoview="off";
//
subplot(1,2,2);
```

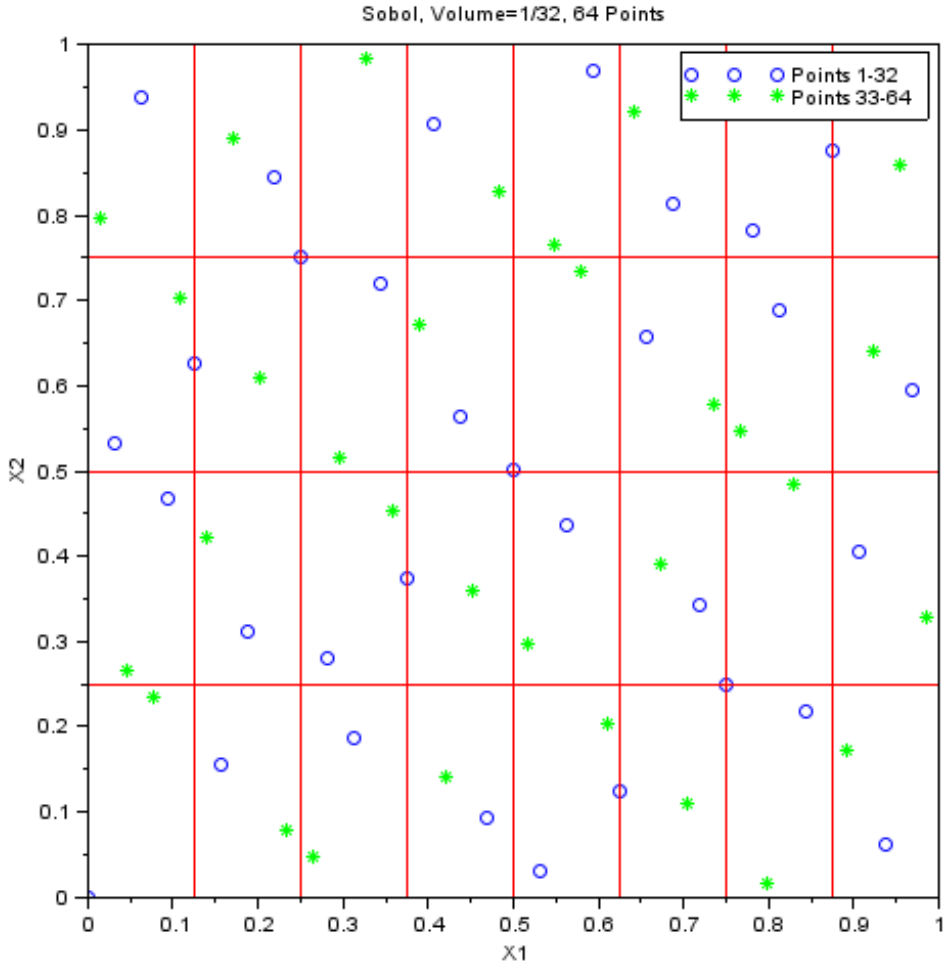

Figure 2.11: The Sobol sequence : 64 points in 2 dimensions, with elementary intervals of area 1/32.

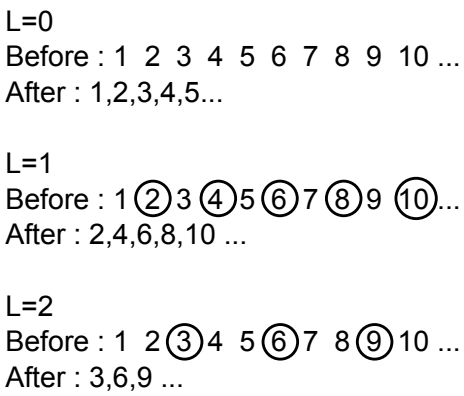

Figure 2.12: Principle of the leap parameter.

```
plot(u(:,1),u(:,2),"bo")
lowdisc_plotelembox([2 3],[1 1])
xtitle("Halton, Volume=1/6, 12 Points");
h = gca();
h.isoview="off";
```

The previous script produces the figure 2.13. In this experiment, we use $L + 1 = 5$, i.e. the third prime number. This is because the primes $b_1 = 2$ and $b_2 = 3$ are used for the dimensions 1 and 2. Although the unleaped Halton sequence had exactly 1 point by elementary interval, the leaped Halton sequence is not as regular : some intervals are empty (the top left on the left figure), some intervals have more points than expected (the bottom left on the left figure).

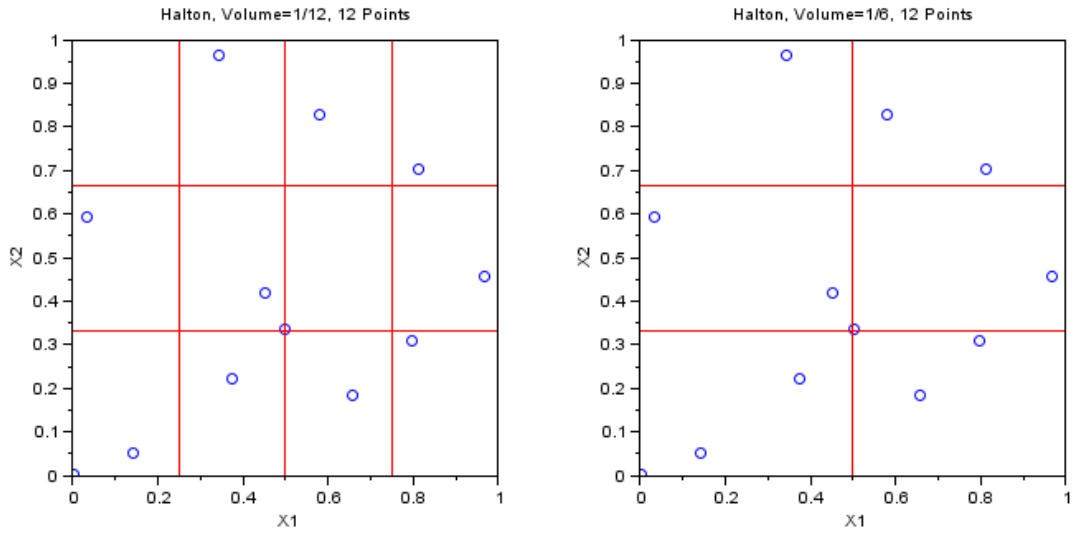

Figure 2.13: The leaped Halton sequence : 12 points in 2 dimensions, with elementary intervals of area 1/6.

The same phenomenon may happen with a leaped Faure sequence. In the following script, we consider $2^4 = 16$ points in 2 dimensions, with elementary intervals of area 1/16. We use the leap

parameter $L = 5$.

```
lds = lowdisc_new("faure");
lds = lowdisc_configure(lds,"-dimension",2);
lds = lowdisc_configure(lds,"-leap",5);
lds = lowdisc_startup (lds);
[lds,u] = lowdisc_next (lds,2^4-1);
lds = lowdisc_destroy(lds);
u = [0,0;u];
scf();
lowdisc_plotbmbox(2,4,u);
```

The previous script produces the figure 2.14.

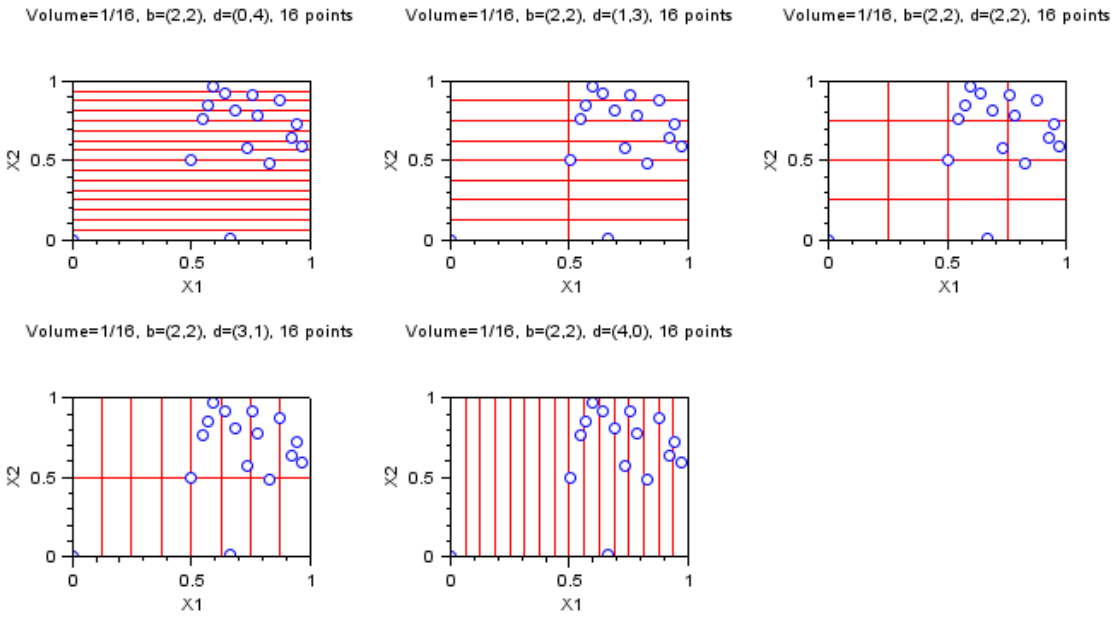

Figure 2.14: The leaped Faure sequence : $2^4 = 16$ points in 2 dimensions, with elementary intervals of area 1/16.

2.10 Notes and references

Most of the content of the section 2.1 is based on Niederreiter's [11]. This is also partly based on Glasserman's [4], especially section 5.1.4 "Nets and sequences", and Fig. 5.3, page 292.

Some of the (t, m, s)-nets plots of the section 2.6 can be seen in [12] or in [14] (see Fig 4.7).

The table 2.3 is the table 4.2 in [14] (Section 4.6 "Les (t,s)-suites et les (t,m,s)-reseaux. and Fig 4.7).

The table 2.2 presents the quality parameters t of two (t,s)-sequences. For Sobol sequence, this is the table in section 3.4 of [2], extended with personal computations. For the Niederreiter sequence in base 2, this is the table II in [10].

Chapter 3

The Van Der Corput sequence

3.1 Introduction

Let $k \geq 0$ be an integer and let b be a prime number. We decompose k into base b:

$$k = a_{r-1}(k)b^{r-1} + ... + a_2(k)b^2 + a_1(k)b + a_0(k)$$

where r is the number of digits required in the decomposition.

We can also write this decomposition as the sum:

$$k = \sum_{i=0}^{r-1} a_i(k)b^i.$$

In order to write the decomposition of an integer in base b, some authors write:

$$k = (a_{r-1}...a_2a_1a_0)_b$$

The radical inverse function is

$$\psi_b(k) = \sum_{i=0}^{r-1} a_i(k)b^{-i-1}.$$

This can be expanded in:

$$\psi_b(k) = a_0(k)b^{-1} + a_1(k)b^{-2} + .. + a_{r-1}(k)b^{-r}.$$

This can also be written:

$$k = (0.a_0a_1a_2...a_{r-1})_b$$

In this function, the least significant digits of k (with small values of i) are in the most significant digits of the radical inverse (with small values of powers of 1/b).

The base-b Van Der Corput sequence[15] is

$$x_i = \psi_b(i), \qquad i \geq 0.$$

Notice that

$$\psi_b(0) = 0.$$

3.2 Van Der Corput sequence in base 2

The Van Der Corput sequence in base 2 is just the first component of the Halton sequence. The following script considers the numbers k from zero to 9, decompose them in base 2 and compute the k-th element of the Van Der Corput sequence in base 2.

```
b=2;
n=b^4;
u=lowdisc_ldgen(n,1,"halton");
for k=1:n
    d=number_tobary(k,b);
    dstr=strcat(string(d),"");
    drev=strcat(string(d($:-1:1)),"");
    mprintf("k=%d=(%s)_%d, r=(0.%s)_%d=%f\n",..
    k,dstr,b,drev,b,u(k,1))
end
```

The previous script produces the following output.

```
k=1=(1)_2, r=(0.1)_2=0.500000
k=2=(10)_2, r=(0.01)_2=0.250000
k=3=(11)_2, r=(0.11)_2=0.750000
k=4=(100)_2, r=(0.001)_2=0.125000
k=5=(101)_2, r=(0.101)_2=0.625000
k=6=(110)_2, r=(0.011)_2=0.375000
k=7=(111)_2, r=(0.111)_2=0.875000
k=8=(1000)_2, r=(0.0001)_2=0.062500
k=9=(1001)_2, r=(0.1001)_2=0.562500
k=10=(1010)_2, r=(0.0101)_2=0.312500
k=11=(1011)_2, r=(0.1101)_2=0.812500
k=12=(1100)_2, r=(0.0011)_2=0.187500
k=13=(1101)_2, r=(0.1011)_2=0.687500
k=14=(1110)_2, r=(0.0111)_2=0.437500
k=15=(1111)_2, r=(0.1111)_2=0.937500
k=16=(10000)_2, r=(0.00001)_2=0.031250
```

For example, the integer k=6 decomposes as 110 in base 2, because $2^2 + 2^1 = 4 + 2 = 6$. Its radical inverse is 0.011 in base 2, which is associated to the radical inverse value $0/2+1/4+1/8 = 0.375$.

The figure 3.1 plots the points of the Van Der Corput sequence in base 2.

In base 2, the lowest order digits are alternatively 0, 1, 0, etc... The most significant digit of the radical inverse function is alternatively 0, 1, 0, etc... Hence, the radical inverse in base 2 is alternatively lower and greater than 1/2. This can be seen in the plot, where the even points are on the left of the figure, while the odd points are on the right of the figure.

Another property which is made clear from the figure is that the interval [0,1] is progressively filled block-by-block, where each block of points has an increasing size which is a power of 2. Indeed, the first block has size 1 (the point 1), the second block has size 2 (the points 2 and 3), the third block has size 4 (the points 4 to 7), and so on. Therefore, when we use a point set which is a power of two, we have a regular discretization of the [0,1] interval.

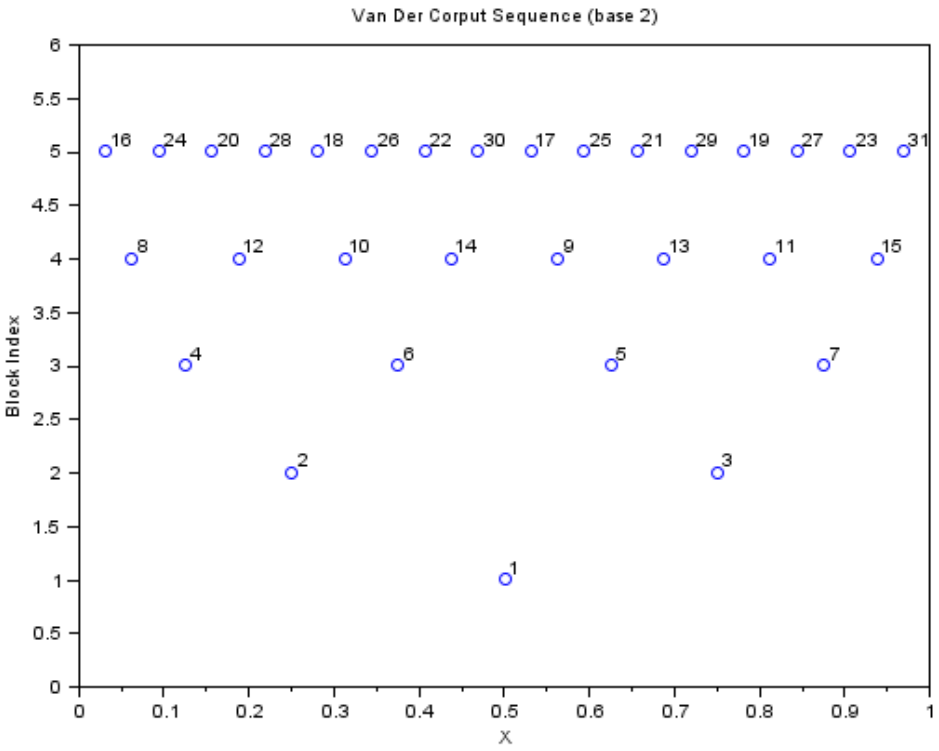

Figure 3.1: The Van Der Corput sequence in base 2.

3.3 Van Der Corput sequence in base 3

To get the Van Der Corput sequence in base 3, we just use the second dimension of the Halton sequence in two dimensions.

```
b=3;
n=b^3;
u=lowdisc_ldgen(n,2,"halton");
for k=1:n
    d=number_tobary(k,b);
    dstr=strcat(string(d),"");
    drev=strcat(string(d($:-1:1)),"");
    mprintf("k=%d=(%s)_%d, r=(0.%s)_%d=%f\n",..
    k,dstr,b,drev,b,u(k,2))
end
```

The previous script produces the following output.

```
k=1=(1)_3, r=(0.1)_3=0.333333
k=2=(2)_3, r=(0.2)_3=0.666667
k=3=(10)_3, r=(0.01)_3=0.111111
k=4=(11)_3, r=(0.11)_3=0.444444
k=5=(12)_3, r=(0.21)_3=0.777778
k=6=(20)_3, r=(0.02)_3=0.222222
k=7=(21)_3, r=(0.12)_3=0.555556
k=8=(22)_3, r=(0.22)_3=0.888889
k=9=(100)_3, r=(0.001)_3=0.037037
k=10=(101)_3, r=(0.101)_3=0.370370
k=11=(102)_3, r=(0.201)_3=0.703704
k=12=(110)_3, r=(0.011)_3=0.148148
k=13=(111)_3, r=(0.111)_3=0.481481
k=14=(112)_3, r=(0.211)_3=0.814815
k=15=(120)_3, r=(0.021)_3=0.259259
k=16=(121)_3, r=(0.121)_3=0.592593
k=17=(122)_3, r=(0.221)_3=0.925926
k=18=(200)_3, r=(0.002)_3=0.074074
k=19=(201)_3, r=(0.102)_3=0.407407
k=20=(202)_3, r=(0.202)_3=0.740741
k=21=(210)_3, r=(0.012)_3=0.185185
k=22=(211)_3, r=(0.112)_3=0.518519
k=23=(212)_3, r=(0.212)_3=0.851852
k=24=(220)_3, r=(0.022)_3=0.296296
k=25=(221)_3, r=(0.122)_3=0.629630
k=26=(222)_3, r=(0.222)_3=0.962963
k=27=(1000)_3, r=(0.0001)_3=0.012346
```

For example, the integer k=6 decomposes as 20 in base 3. Its radical inverse is 0.02 in base 3, which is associated to the radical inverse value 2/9=0.2222...

The figure 3.2 plots the points of the Van Der Corput sequence in base 3.

We see that the [0,1] interval is regularily discretized when we consider a number of points which is either 3-1=2, 9-1=8 or 27-1=26 points. In other words, when the number of points is a

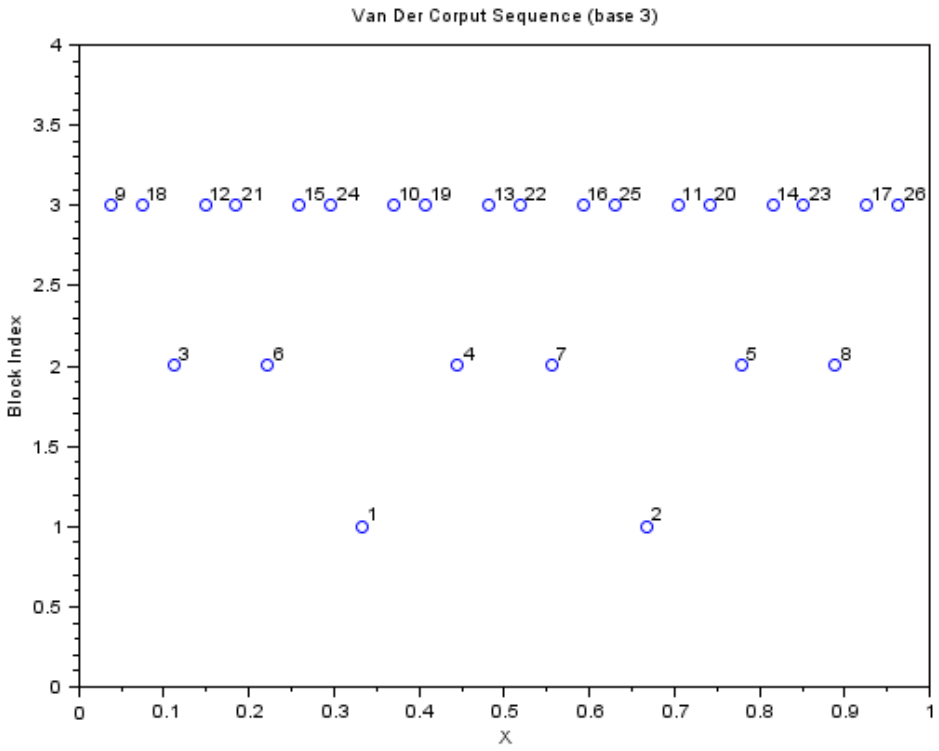

Figure 3.2: The Van Der Corput sequence in base 3.

power of the base (which is here equal to 3), then the interval is filled more regularly.

The figure 3.3 plots the points of the Van Der Corput sequence in base 5.

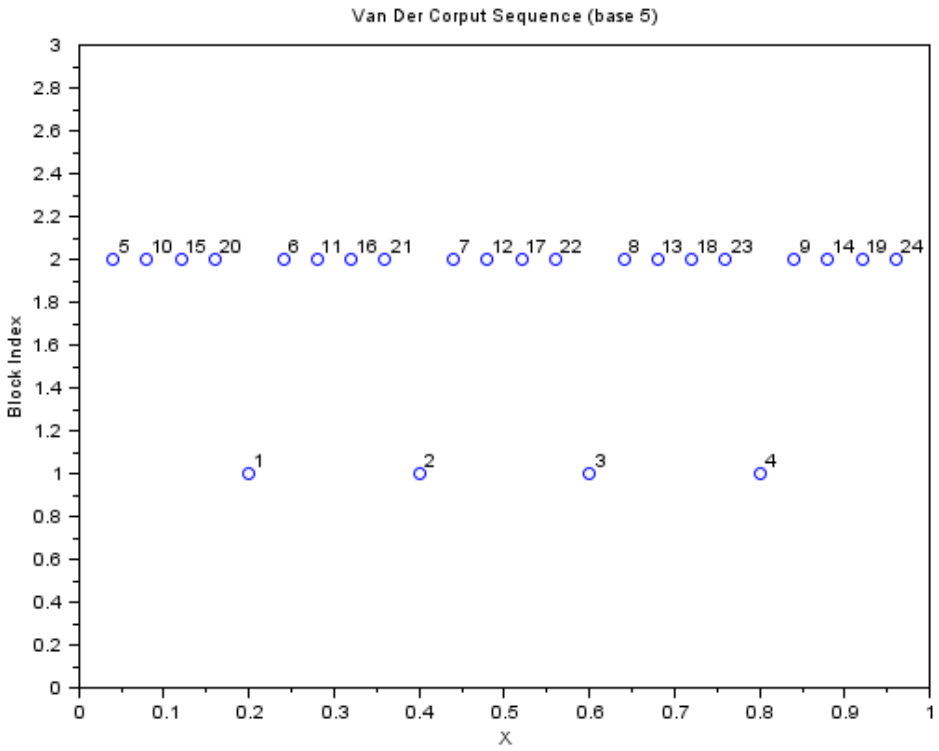

Figure 3.3: The Van Der Corput sequence in base 5.

When the base increases, the number of points before the [0,1] interval is regularly filled increases rapidly (as a power of b).

Moreover, consider the points 5 to 9, then 10 to 14 : they progressively go from left to right, taking b=5 iterations to make a partial circle. More generally, the subsequences take more iterations to go from lower values (near 0) to larger values (near 1). This is because it takes b iterations for the the least significant digits to go from 0 to b-1.

3.4 Properties

The Van Der Corput sequence is a (0,1)-sequence. Hence, for any $m \geq 0$, consider the point set with indices $kb^m, ..., (k+1)b^m - 1$, for k=0,1,...,b-1. Then, any elementary interval of length $1/b^m$ contains exactly one point. For m=0, this just implies that the points are in the [0,1] interval.

In the following script, we consider the Van Der Corput point sets for the base b=5. This is just the third dimension of the Halton sequence. For m=1, we consider the previously defined point sets and plot the limits of the intervals of length $1/b^m$. Then we plot the point sets, for k=0,1,...,b-1.

```
s=3;
```

```
b=5;
m=1;
u=lowdisc_ldgen(b^(m+1),s,"halton");
u=[zeros(1,s);u];
u=u(:,s);
scf();
// Plot the limits
for i=1:b^m
    x=i/b^m;
    plot([x x],[0 b-1],"r-")
end
// Plot the point sets
for k=0:b-1
    indices=k*b^m+1:(k+1)*b^m;
    plot(u(indices)',k,"bo")
    xstring(u(indices)',k,string(indices))
end
a = gca();
a.data_bounds=[
0 -0.5
1 b-0.5
];
a.isoview="off";
st=msprintf("Van␣Der␣Corput␣Sequence,␣Base␣%d,␣%d␣Points",b,b^(m+1));
xtitle(st,"X","Block␣Index")
```

The previous script produces the figure 3.4. We check that each point set has exactly one point in each elementary interval, with the convention that the intervals are closed on the left and open on the right.

When we set $s = 1$ and $b = 2$, we get the Van Der Corput sequence in base b=2, which is presented in the figures 3.5, 3.6 and 3.7.

3.5 Example

In this section, we present an exemple which shows how particular segments of the Van Der Corput behave. These segments are chosen so that they match the properties of (t, s)-sequences.

Let $m, k \geq 0$ be two integers, let b be a prime number and consider the elements on the Van Der Corput sequence in base b with indices $i = kb^m, ..., (k+1)b^m - 1$. The following script presents the Van Der Corput sequence in base $b = 2$ and considers $m = 5$ and $k = 13$. We first decompose the integer i into base $b = 2$.

```
b=2;
m=5;
u=lowdisc_ldgen(1000,1,"halton");
k=13;
for i=k*b^m:(k+1)*b^m-1
    d=number_tobary(i,b)';
    mprintf("i=%d=(%s)\n",i,strcat(string(d),","))
end
```

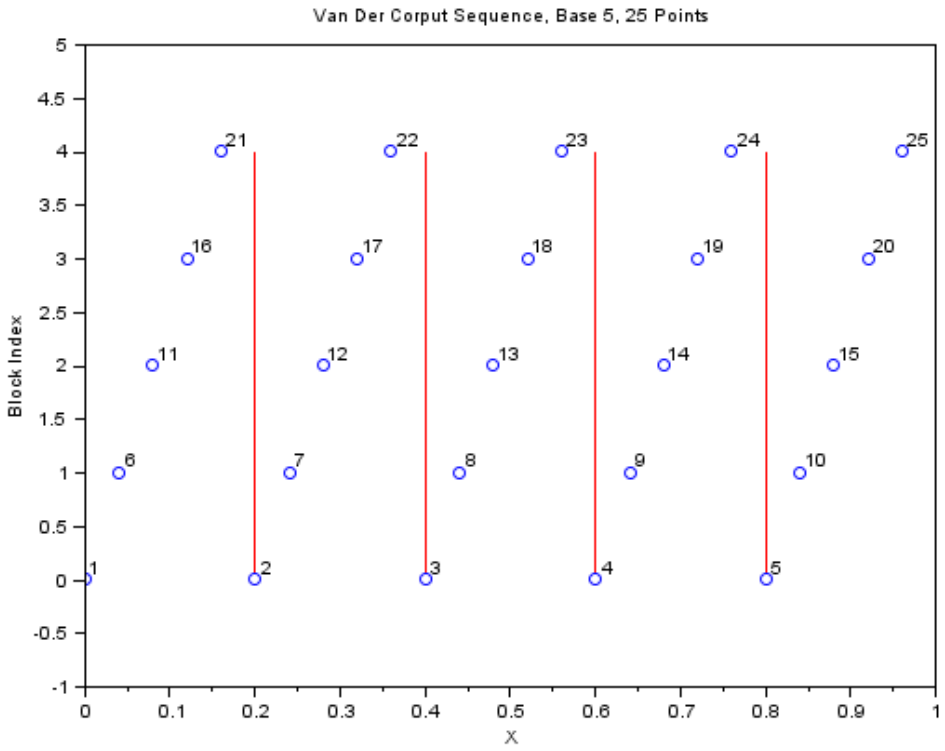

Figure 3.4: The Van Der Corput sequence in base 5 - 25 points.

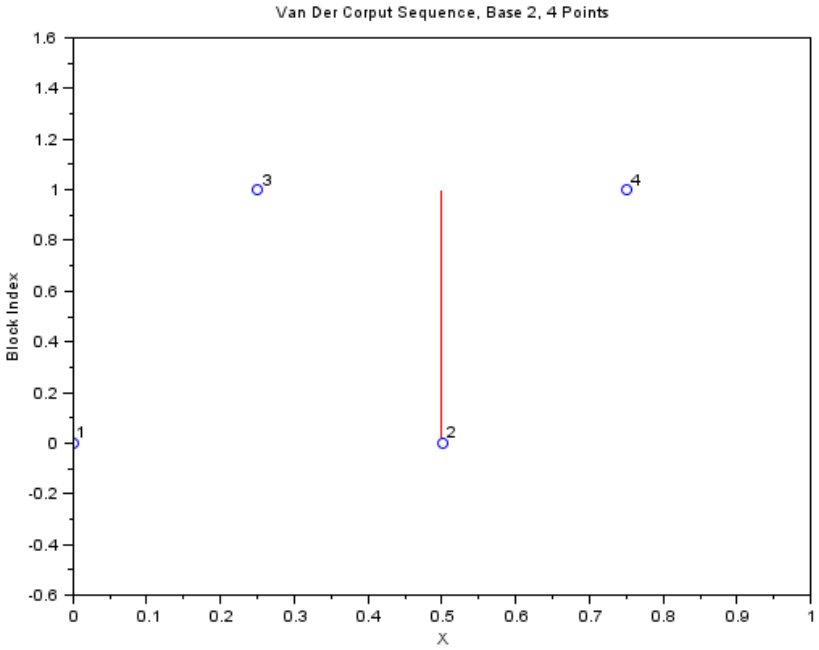

Figure 3.5: The Van Der Corput sequence in base 2 - 4 points.

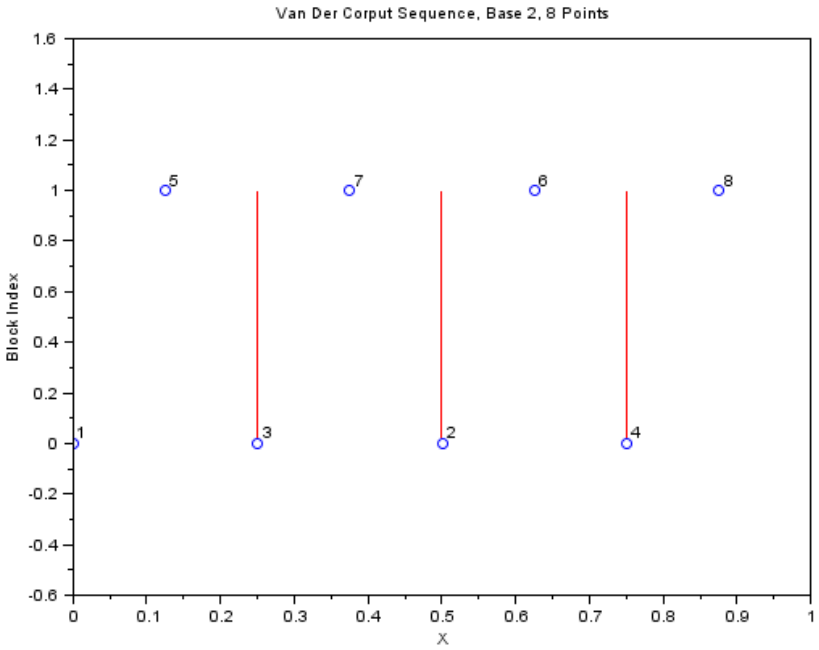

Figure 3.6: The Van Der Corput sequence in base 2 - 8 points.

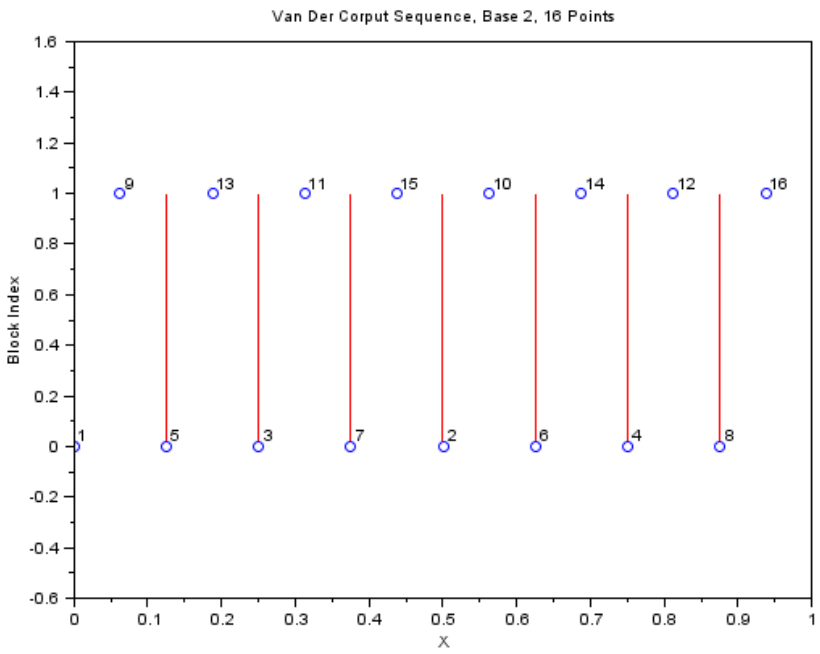

Figure 3.7: The Van Der Corput sequence in base 2 - 16 points.

The previous script produces the following output.

```
i=416=(1,1,0,1,0,0,0,0,0)
i=417=(1,1,0,1,0,0,0,0,1)
i=418=(1,1,0,1,0,0,0,1,0)
i=419=(1,1,0,1,0,0,0,1,1)
i=420=(1,1,0,1,0,0,1,0,0)
[...]
i=443=(1,1,0,1,1,1,0,1,1)
i=444=(1,1,0,1,1,1,1,0,0)
i=445=(1,1,0,1,1,1,1,0,1)
i=446=(1,1,0,1,1,1,1,1,0)
i=447=(1,1,0,1,1,1,1,1,1)
```

We see that the 9 bits are required to decompose these integers into base 2. The four first bits are $(1,1,0,1)$, which is explained by the fact that $k = 13 = 2^3 + 2^2 + 2^0$. The bits from 5 to 9 are all the possible bits in the set $\{0,1\}$.

Then we decompose the i-th element of the Van Der Corput sequence `u(i)` into base $b = 2$, printing the first 10 digits (the remaining low order digits are all zero).

```
for i=k*b^m:(k+1)*b^m-1
    d=flps_tobary(u(i),b,10)';
    mprintf("u(i)=%f=(%s)\n",u(i),strcat(string(d),","))
end
```

The previous script produces the following output.

```
u(416)=0.021484375=(0,0,0,0,0,0,1,0,1,1)
u(417)=0.521484375=(0,1,0,0,0,0,1,0,1,1)
u(418)=0.271484375=(0,0,1,0,0,0,1,0,1,1)
u(419)=0.771484375=(0,1,1,0,0,0,1,0,1,1)
[...]
u(444)=0.240234375=(0,0,0,1,1,1,1,0,1,1)
u(445)=0.740234375=(0,1,0,1,1,1,1,0,1,1)
u(446)=0.490234375=(0,0,1,1,1,1,1,0,1,1)
u(447)=0.990234375=(0,1,1,1,1,1,1,0,1,1)
```

For example, the decomposition of 0.521484375 is $1/2 + 1/2^6 + 1/2^8 + 1/2^9$.

We see that the 4 low order bits are $(1, 0, 1, 1)$, which are the reverse bits of the decomposition of k. The bits from 2 to 6 are all the possible bits in the set $\{0, 1\}$.

3.6 Property

Proposition 3.6.1. *The Van Der Corput sequence is a $(0, 1)$-sequence.*

Proof. Let a be an integer in the set $\{0, 1, ..., b^m - 1\}$. Let $I = [a/b^m, (a+1)/b^m)$ be an elementary interval. Its length is $1/b^m$. Let $k \geq 0$ be an integer. Let us prove that there is a unique i in the set $\{kb^m, ..., (k + 1)b^m - 1\}$ such that $\phi(i) \in I$, where $\phi(i)$ is the i-th element of the Van Der Corput sequence in base b. Let $a(0), a(1), ..., a(m - 1) \in \{0, 1, ..., b - 1\}$ be the m digits of the base-b decomposition of a:

$$a = a(0)b^{m-1} + ... + a(m - 2)b + a(m - 1). \tag{3.1}$$

Notice that the digits are ordered from the highest to the lowest significant digits. Let $a(m), a(m+1), ..., a(m + r - 1) \in \{0, 1, ..., b - 1\}$ be the r digits of the base-b decomposition of k in base b:

$$k = a(m + r - 1)b^{r-1} + ... + a(m + 1)b + a(m). \tag{3.2}$$

Notice that the digits are ordered from the lowest to the highest significant digits. Let us introduce:

$$\begin{aligned} t &= \frac{a}{b^m} && (3.3)\\ &= \frac{a(0)}{b} + \frac{a(1)}{b^2} + ... + \frac{a(m-1)}{b^m} && (3.4)\\ & && (3.5) \end{aligned}$$

and

$$x = \frac{a(m)}{b} + \frac{a(m+1)}{b^2} + ... + \frac{a(m+r-1)}{b^r}. \tag{3.6}$$

Notice that t is uniquely defined from the digits arising from the base-b decomposition of a while x is uniquely defined from the digits arising from the base-b decomposition of k. Consider the number

$$\begin{aligned} y &= t + \frac{x}{b^m} && (3.7)\\ &= \frac{a(0)}{b} + \frac{a(1)}{b^2} + ... + \frac{a(m-1)}{b^m} \\ &\quad + \frac{a(m)}{b^{m+1}} + \frac{a(m+1)}{b^{m+2}} + ... + \frac{a(m+r-1)}{b^{m+r}} && (3.8) \end{aligned}$$

Obviously $x \geq 0$. Moreover,

$$\begin{aligned} x &\leq \frac{b-1}{b} + \frac{b-1}{b^2} + ... + \frac{b-1}{b^r} && (3.9) \\ &< 1. && (3.10) \end{aligned}$$

Indeed, this is a geometric series:

$$\begin{aligned} \frac{b-1}{b} + \frac{b-1}{b^2} + ... + \frac{b-1}{b^r} &= \frac{b-1}{b}\left(1 + \frac{1}{b} + \frac{1}{b^2} + ... + \frac{1}{b^{r-1}}\right) && (3.11) \\ &= \frac{b-1}{b}\frac{1-\frac{1}{b^r}}{1-\frac{1}{b}} && (3.12) \\ &= \frac{b-1}{b}\frac{1-\frac{1}{b^r}}{\frac{b-1}{b}} && (3.13) \\ &= 1 - \frac{1}{b^r} && (3.14) \\ &< 1. && (3.15) \end{aligned}$$

Therefore, $x \in [0, 1)$. Hence, $a + x \in [a, a + 1)$ so that $a/b^m + x/b^m \in I$. But $t = \frac{a}{b^m}$, which implies $t + x/b^m = y \in I$. In order to conclude the proof, we recognize that y is the i-th element of the Van Der Corput sequence in base b, for a particular value of i. To see this value of i, we just reverse the digits of the definition of y from the equation 3.8. This leads to

$$\begin{aligned} i &= a(m+r-1)b^{m+r-1} + ... + a(m+1)b^{m+1} + a(m)b^m \\ &\quad a(m-1)b^{m-1} + ... + a(1)b + a(0) && (3.16) \\ &= \left(a(m+r-1)b^{r-1} + ... + a(m+1)b + a(m)\right)b^m \\ &\quad a(m-1)b^{m-1} + ... + a(1)b + a(0) && (3.17) \\ &= kb^m + a(m-1)b^{m-1} + ... + a(1)b + a(0), && (3.18) \end{aligned}$$

where the last equality is implied by the equation 3.2. Moreover,

$$0 \leq a(m-1)b^{m-1} + ... + a(1)b + a(0) \leq b^m - 1.$$

Hence

$$kb^m \leq i \leq kb^m + b^m - 1 = (k+1)b^m - 1.$$

We have computed the unique i such that $y = \phi(i)$ is in I, which concludes the proof. □

Chapter 4

The Halton sequence

4.1 Introduction

The Hammersley sequence (1960) uses the Van Der Corput sequence in order to create a low discrepancy point set [6]. However, the number of points in the Hammersley sequence must be know in advance, which is sometimes an issue. The Halton sequence (1964) uses the Van Der Corput sequence without this limitation [5], and this is why it is used more often.

The Halton sequence in s dimensions uses s different bases which must be relatively primes. We usually use the s first prime numbers: 2, 3, 5, 7, 11, etc...

Denote by $p(i)$ the i-th prime number. The dimension i of the Halton sequence is the Van Der Corput in base $p(i)$. Hence, the k-th element of the Halton sequence is

$$x_k = (\psi_{p_1}(k), \psi_{p_2}(k), ..., \psi_{p_s}(k))$$

for $k \geq 0$.

The following script produces 10 points of the Halton sequence in 4 dimensions. The prime number used here are: 2, 3, 5, 7.

```
-->u=lowdisc_ldgen(9,4,"halton")
 u  =
    0.5         0.3333333    0.2      0.1428571
    0.25        0.6666667    0.4      0.2857143
    0.75        0.1111111    0.6      0.4285714
    0.125       0.4444444    0.8      0.5714286
    0.625       0.7777778    0.04     0.7142857
    0.375       0.2222222    0.24     0.8571429
    0.875       0.5555556    0.44     0.0204082
    0.0625      0.8888889    0.64     0.1632653
    0.5625      0.0370370    0.84     0.3061224
```

The lowest components of the Halton sequence make use of smaller primes while higher components make use of the larger primes. Moreover, we know that the size of the cycle which allows the sequence to fill the space depends on the value of the corresponding prime. Hence, if the user of a Halton sequence orders the variables so that the most important variables come first, then the Halton sequence can potentially lead to better results.

4.2 Scrambled Halton Sequence

In high dimensions, the Halton sequence poorly fills the space, as can be seen in low dimensionnal projections. In order to improve this, Braaten and Weller (1979) suggest to scramble the digits in the Halton sequence, in order to break correlations[1].

The permuted radical inverse function is

$$\psi_b(k) = \sum_{i=0}^{r-1} \frac{\sigma(a_i(k))}{b^{i+1}}.$$

where σ is a given permutation.

For a given base b, the permutation does not change the values of the points in the sequence: only the ordering of the points changes. However, since each dimension has a specific base b, the multidimensionnal point set of a scrambled sequence is, in general, different from the unscrambled point set.

The following function computes the scrambled Van Der Corput radical inverse function.

```
function r = scrambleVDC(index,b,sigma)
    current = index
    ib = 1.0 / b
    r = 0.0
    while (current>0)
        digit = modulo ( current , b )
        digit=sigma(digit+1)
        current = int ( current / b )
        r = r + digit * ib
        ib = ib / b
    end
endfunction
```

For example, the base-3 Van Der Corput sequence can be associated with the permutation [0,2,1]. The following script computes the first ten points of this scrambled sequence.

```
sigma=[0,2,1];
b=3;
for index=1:10
    r = scrambleVDC(index,b,sigma);
    mprintf("index=%d,␣r=%f\n",index,r)
end
```

The previous script produces the following output.

```
index=1, r=0.666667
index=2, r=0.333333
index=3, r=0.222222
index=4, r=0.888889
index=5, r=0.555556
index=6, r=0.111111
index=7, r=0.777778
index=8, r=0.444444
index=9, r=0.074074
index=10, r=0.740741
```

In their paper, Braaten and Weller provide the permutations for dimensions up to 16.

```
2 (0 1)
3 (0 2 1)
5 (0 3 1 4 2)
7 (0 4 2 6 1 5 3)
11 (0 5 8 2 10 3 6 1 9 7 4)
13 (0 6 10 2 8 4 12 1 9 5 11 3 7)
etc...
```

The issue here is that the limited size of the table provided by Braaten and Weller limits the practical use of the algorithm. Tuffin (1996) provides improved algorithms to expand the table (and to improve the permutations as well), but this work has two limitations. First, the permutations are computed by taking all dimensions into account, and not dimension-by-dimension as in Braaten and Weller. This has the advantage of improving the permutations, but has the drawback of making the table larger. Secondly, Tuffin published the algorithm, but not the actual permutations.

Kocis and Whiten (1979) provide a simple algorithm to generate permutations for the Halton sequence [8]. Their RR2 algorithm uses the Van Der Corput sequence in base 2 and reverses the digits, removing the integers that are too large. A more detailed description is presented in Lemieux [9]. This algorithm has no strong theoretical basis, but works efficiently.

The following function computes the RR2 scrambling for the i-th dimension of a point set with `s` dimensions, using the bases in the matrix `b`.

```
function sigma=RR2Scrambling(s, i, b)
    ns=ceil(log(b(s))/log(2))
    sigma=[];
    u=lowdisc_ldgen(2^ns,1,"halton")
    u=[0;u]
    for k=1:2^ns
        sigmak=u(k)*2^ns
        if (sigmak<b(i)) then
            sigma($+1)=sigmak;
        end
    end
endfunction
```

The following script prints the RR2 scrambling used for a point set in 3 dimensions.

```
s=3;
primemat=number_primes100();
b=primemat(1:s);
for i=1:s
    sigma=RR2Scrambling(s,i,b);
    sigmastr=strcat(string(sigma),",");
    mprintf("i=%d, [%s]\n",i,sigmastr);
end
```

The previous script produces the following output.

```
i=1, [0,1]
i=2, [0,2,1]
i=3, [0,4,2,1,3]
```

This feature is provided by the "-scrambling" option of the `lowdisc_configure` function.

Chapter 5

The Faure sequence

5.1 Principles

Let s be the number of dimensions. The Faure sequence uses the smallest prime number greater or equal than b. For example, if $s = 6$, then the prime number b used by the Faure sequence is 7.

Let k be an integer and let $a_i(k)$ be the coefficients of the base-b decomposition of k:

$$k = \sum_{i=0}^{r-1} a_i(k) b^i.$$

Let i be an integer in the set $\{1, 2, ..., s\}$. The i-th coordinate of the Faure sequence makes use of a r-by-r upper triangular matrix, called the generator matrix. For the Faure sequence, the generator matrix is based on the Pascal matrix defined as:

$$C^{(i)}(m, n) = \binom{n-1}{m-1} i^{n-m},$$

where

$$\binom{n-1}{m-1} = \frac{m!}{n!(m-n)!}$$

for

$$n \geq m$$

and zero otherwise.

By convention $C^{(0)}$ is identity matrix.

Consider the column vector $a(k)$ defined by ordering the digits from the least significant to the most significant:

$$a(k) = (a_0(k), a_1(k), ..., a_{r-1}(k))^T.$$

We introduce the column vector $y(k)$ defined by the matrix-vector product:

$$y^{(i)}(k) = C^{(i-1)} a(k) \pmod b$$

The i-th component of the Faure sequence is:

$$x_k^{(i)} = \sum_{i=0}^{r-1} y^{(i)}(k) b^{-i-1}.$$

5.2 Properties

The matrices $C^{(i)}$ are so that

$$C^{(i)} a(k) \pmod b$$

is a permutation of the vector $a(k)$ for the set of points $k = jb^r, ..., (j+1)b^r - 1$, for any $j = 0, 1, ..., b-1$. Hence, for this set of points, each component $x_k^{(i)}$ is a permutation of the Van Der Corput sequence $\psi_b(k)$.

The first component of the Faure sequence is the Van Der Corput sequence in base b. Other components of the Faure sequence are permutations of sub-sequences of the Van Der Corput sequence in base b.

Faure sequences are (0,s)-sequences in base b. Therefore, Faure sequences have the lowest possible value of the quality parameter t in base b. On the other hand, the base increases with the dimension s, although the base for the Faure sequence increases much slower than the largest base of the Halton sequence.

5.3 Examples

The following function computes the Faure matrix, given the number of digits r and the dimension index i.

```
function c = fauregmatrix ( r , i )
    c = zeros(r,r)
    for m = 1:r
        b=specfun_nchoosek ((m:r)-1 , m-1)
        c(m,m:r) = i.^((m:r)-m) .* b
    end
endfunction
```

For r=1, the Faure matrix is equal to one. The following script prints the Faure matrices for increasing values of r and i.

```
for r=2:3
    for i=1:3
        c = fauregmatrix ( r , i );
        mprintf("r=%d, C(%d)\n",r,i)
        disp(c)
    end
end
```

The previous script produces the following output.

```
r=2, C(1)
    1.    1.
    0.    1.
r=2, C(2)
```

```
    1.    2.
    0.    1.
r=2, C(3)
    1.    3.
    0.    1.
r=3, C(1)
    1.    1.    1.
    0.    1.    2.
    0.    0.    1.
r=3, C(2)
    1.    2.    4.
    0.    1.    4.
    0.    0.    1.
r=3, C(3)
    1.    3.    9.
    0.    1.    6.
    0.    0.    1.
```

The following script produces the first 10 points of the Faure sequence in s=3 dimensions. For this value of s, the Faure sequence uses the base b=3. We use the `number_tobary` function in order to decompose the integer k into base b. The "bigendian" option reverses the order of the digits, so that the most significant digit comes last. Then we compute the matrix-vector product `C*a` and compute the remainder modulo b. Finally, we evaluate the point by using powers of the inverse of b.

```
b=3;
s=3;
for k=1:10
    a=number_tobary(k,b,"bigendian");
    r=size(a,"*");
    bpow=b.^(1:r)';
    for i=1:s
        C=faurematrix ( r , i-1 );
        y=modulo(C*a,b);
        x(k,i)=sum(y./bpow);
    end
end
disp(x)
```

The previous script produces the following output.

```
-->disp(x)
    0.3333333    0.3333333    0.3333333
    0.6666667    0.6666667    0.6666667
    0.1111111    0.4444444    0.7777778
    0.4444444    0.7777778    0.1111111
    0.7777778    0.1111111    0.4444444
    0.2222222    0.8888889    0.5555556
    0.5555556    0.2222222    0.8888889
    0.8888889    0.5555556    0.2222222
    0.0370370    0.5925926    0.4814815
```

```
    0.3703704    0.9259259    0.8148148
```

We can compare with the second component of the Halton sequence in s=2 dimensions, which uses the prime number b=3 as the base.

```
u=lowdisc_ldgen(10,2,"halton");
u(:,2)
```

The previous script produces the following output.

```
-->u(:,2)
 ans  =
    0.3333333
    0.6666667
    0.1111111
    0.4444444
    0.7777778
    0.2222222
    0.5555556
    0.8888889
    0.0370370
    0.3703704
```

5.4 Properties of 1D projections

The following script computes the elements of the first 27 points of the Faure sequence in 3 dimensions. We plot the first component at the top, the second component in the middle and the third component at the bottom. We see that all the points in each coordinate are at the same positions, but come in a different order. This point set is made, indeed, of permuted point sets from the Van Der Corput sequence in base b=3.

```
s=3;
b=3;
scf();
lds = lowdisc_new("faure");
lds = lowdisc_configure(lds,"-dimension",s);
lds = lowdisc_startup (lds);
k=1;
imax=3;
for i=1:imax
    npoints=(b-1)*b^(i-1);
    [lds,u]=lowdisc_next(lds,npoints);
    for j=1:s
        subplot(s,1,j)
        plot(u(:,j)',i,"bo")
        xstring(u(:,j)',i,string(k:k+npoints-1))
    end
    k=k+npoints;
end
lds = lowdisc_destroy (lds);
```

```
for j=1:s
    subplot(s,1,j)
    a = gca();
    a.data_bounds=[
    0 0
    1 imax+1
    ];
    xtitle("Faure␣Sequence","X"+string(j),"Block␣Index")
end
```

The previous script produces the figure 5.1.

We can also look at the distribution of sub-sequences of the Faure point set in s=3 dimensions. In the following script, we consider a set of $3^4 = 81$ points, organized in 3 consecutive blocks of $3^3 = 27$ points. We plot the first component at the top, the second component in the middle and the third component at the bottom. The goal of this experiment is to visualize the point sets with indices $ib^r, ..., (i+1)b^r - 1$, for $i = 0, 1, ..., b-1$ with r=2. We can see these particular sub-sequences of consecutive 27 points are evenly filling the one dimensionnal interval $[0, 1]$, so that the sub-sequences are translated by $1/27$ each time a new block i is considered.

```
b=3;
s=3;
u=lowdisc_ldgen(3^3,3,"faure");
u=[zeros(1,s);u];
scf();
npoints=b^3;
for i=0:b-1
    for j=1:s
        subplot(s,1,j)
        indices=i*b^2+1:(i+1)*b^2
        plot(u(indices,j)',i,"bo")
        xstring(u(indices,j)',i,string(indices))
    end
end
for j=1:s
    subplot(s,1,j)
    a = gca();
    a.data_bounds=[
    0 -1
    1 b
    ];
    xtitle("Faure␣Sequence","X"+string(j),"Block␣Index")
end
```

The previous script produces the figure 5.2.

5.5 Value of the prime base

One potential advantage of the Faure sequence over the Halton sequence is that the prime base is much smaller than the largest prime used in the Halton sequence. This is the prime number used

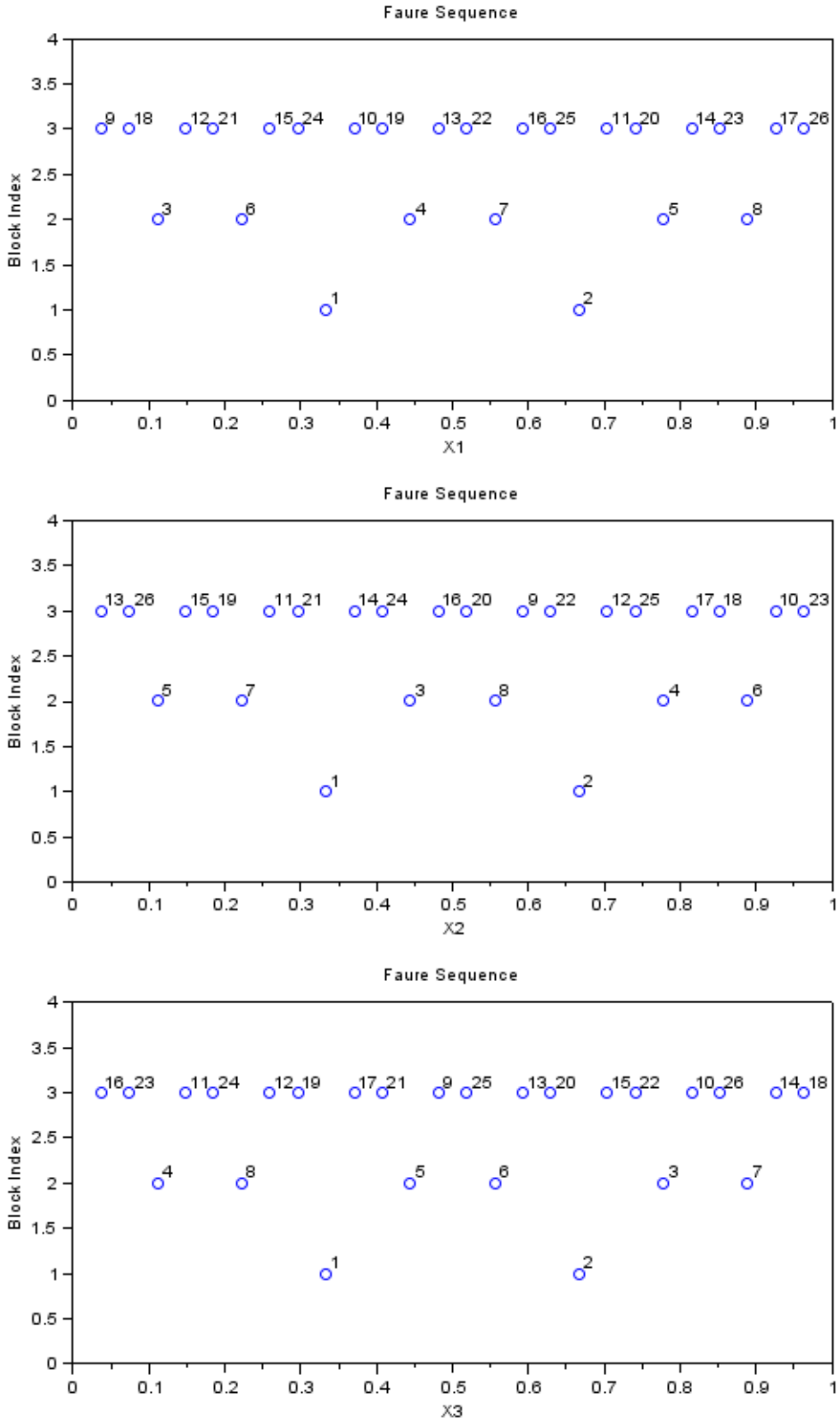

Figure 5.1: The Faure sequence - Projections of n=27 points in s=3 dimensions.

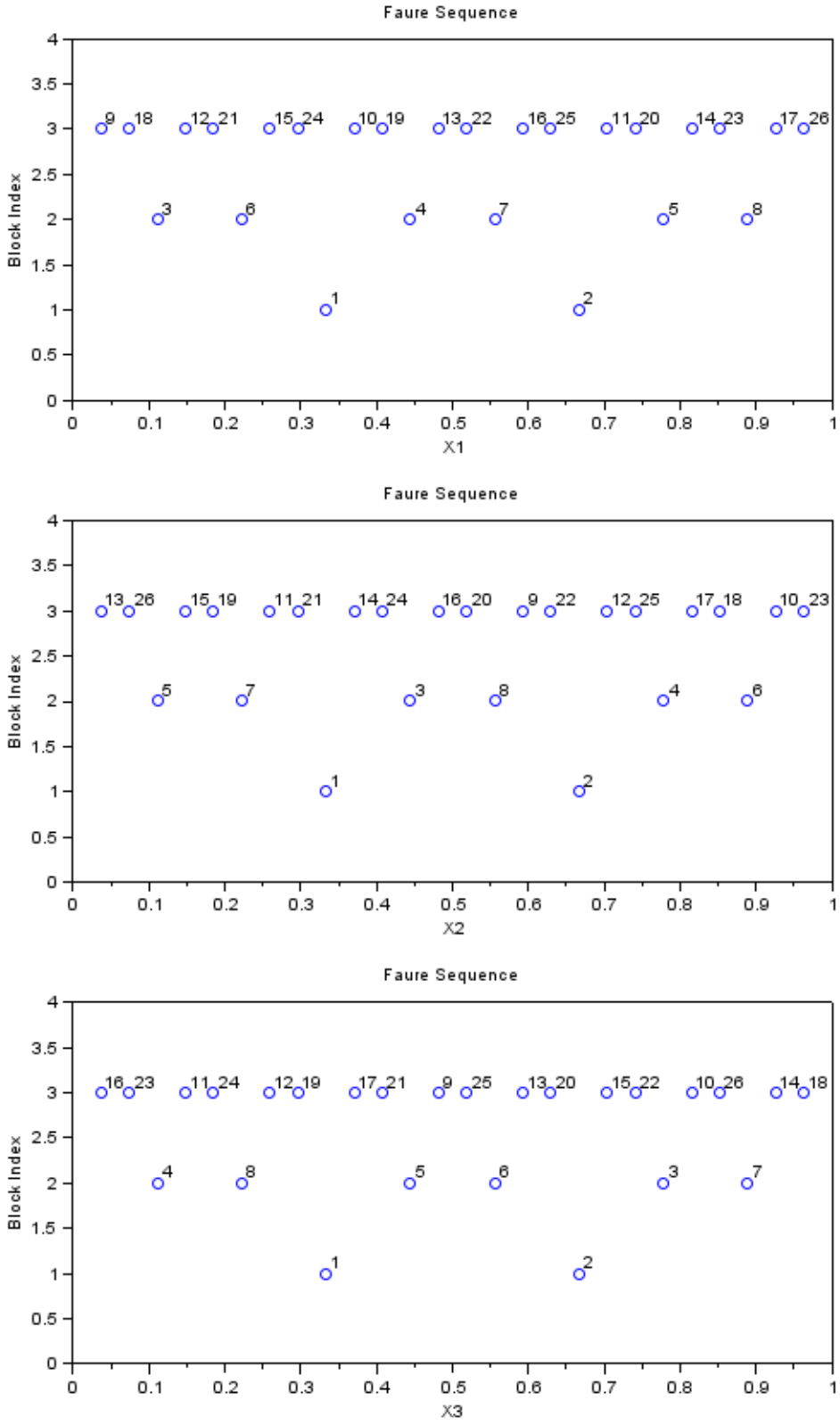

Figure 5.2: The Faure sequence - Projections of n=27 points in s=3 dimensions.

by the s-th component of the sequence. This is because, in s dimensions, the Halton sequence uses the prime numbers $p(1), p(2), ..., p(s)$, while the Faure sequence only uses the smallest prime larger or equal to n.

The following script compares the base used by the Faure sequence for $s = 1, 2, ..., 100$ and compare it to the largest base used by the Halton sequence.

```
p=number_primes100();
scf();
plot(1:100,p,"r-")
b=[];
for n=1:100
    k=find(p>n,1);
    b(n)=p(k);
end
plot(1:100,b,"b-")
legend(["Halton Largest Base" "Faure Base"]);
xtitle("","Number of Dimensions","Prime Base")
```

We can see in the figure 5.3 that the Halton sequence can make use of much larger bases, which potentially leads to much larger number cycles.

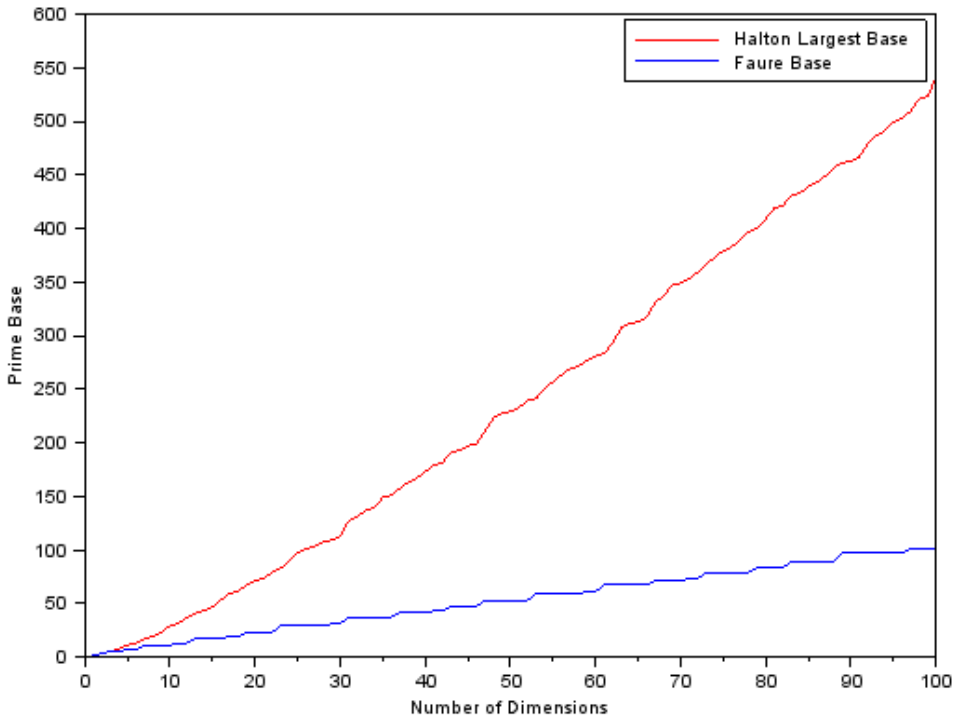

Figure 5.3: The base used in the Faure sequence compared with the largest base used in the Halton sequence.

However, the lowest components of the Halton sequence may make use of potentially much

smaller bases than the equivalent Faure sequence. Indeed, if the user of a Halton sequence orders the variables so that the most important variables come first, then the Halton sequence can potentially lead to better results. On the other hand, the Faure sequence does not favor any component, which may be safer in the general situation where we do not know in advance which variable may be important.

5.6 Notes and references

Most of the theoretical content of this chapter is based on Glasserman's [4].

Chapter 6

Efficiency

6.1 Error bounds

In this example, we show how the Quasi-Monte-Carlo (QMC) integration outperforms the Monte-Carlo integration on a typical example.

In general, the deterministic absolute error of QMC depends on

$$\frac{(\log n)^s}{n}$$

where n is the number of points and s is the number of dimensions.

In general, the random absolute error of Monte-Carlo simulation depends on

$$\frac{1}{\sqrt{n}}$$

Hence, if s is moderate (i.e. from 1 to 20), the asymptotic rate of convergence of QMC is much faster than the rate of Monte-Carlo.

Furthermore, it is known that, when the multi-dimensionnal function is purely additive, i.e. when it is the sum of low-dimensionnal functions, then QMC can reach its peak performance, which is very near 1/n when s is moderate.

The following script plots the Monte-Carlo and Quasi-Monte-Carlo convergence rates, for increasing values of n and s=1,2 and 4.

```
scf();
n=2.^(1:30);
mcconv=1 ./sqrt(n);
qmcconv1=log(n).^1 ./n;
qmcconv2=log(n).^2 ./n;
qmcconv4=log(n).^4 ./n;
plot(n,mcconv,"k-");
plot(n,qmcconv1,"r-");
plot(n,qmcconv2,"g-");
plot(n,qmcconv4,"b-");
legend(["Monte-Carlo" ..
"Quasi-Monte-Carlo␣s=1" ..
"Quasi-Monte-Carlo␣s=2" ..
```

```
"Quasi-Monte-Carlo s=4" ..
]);
a=gca();
a.log_flags="lln";
xtitle("Convergence rate","Number of Simulations",..
"Absolute Error");
```

The previous script produces the figure 6.1.

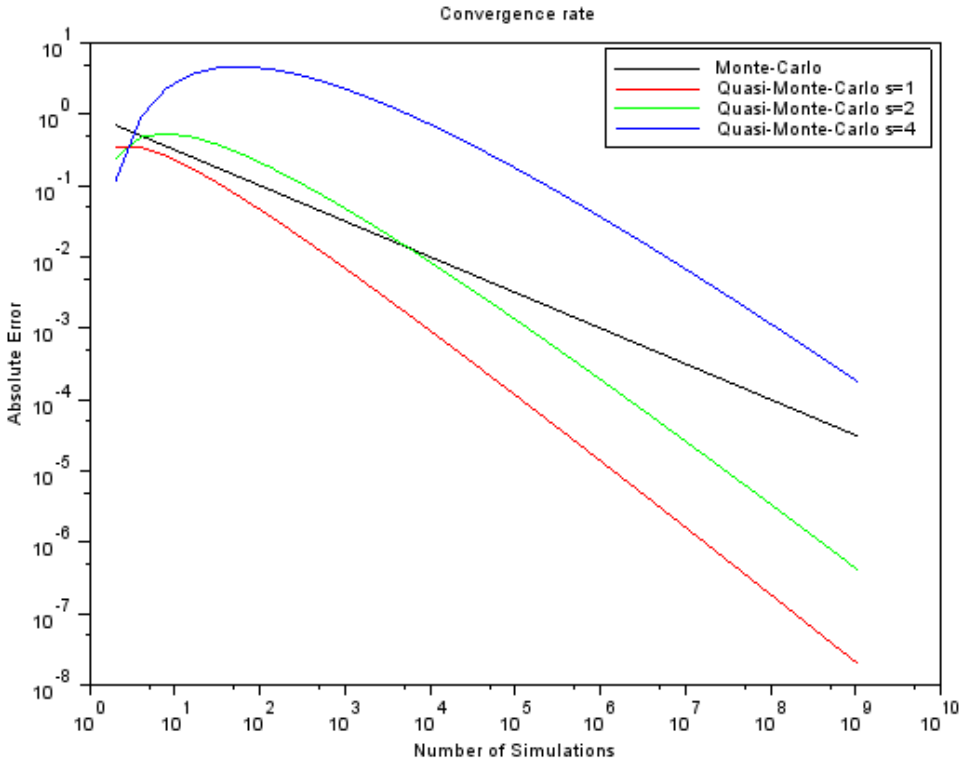

Figure 6.1: Theoretical convergence of Monte Carlo and Quasi-Monte Carlo.

We can see that the theoretical QMC convergence rate first increases, then decreases. It can be shown that the rate $\log(n)^s/n$ is increasing for n lower than exp(s), and then decreasing. For s=3, this is $n = \exp(3)$, which is almost equal to n=20, after which the asymptotic rate of QMC sets in. Hence, we see that the number of points n after which the asymptotic convergence sets in is larger when s increases. Moreover, we see that the accuracy of the integral as predicted by the QMC error bound is much larger when s increases. Fortunately, QMC can be, in some cases, much more accurate than the accuracy predicted by the error bound.

6.2 The Ishigami example

We consider the Ishigami [1] example, which is often used as a test case for sensitivity analysis. This function depends on 3 input variables in the interval $[-\pi, \pi]$. It is not purely additive, but

depends on interactions between x_1 and x_3. Nevertheless, more that 75% of the variance of f can be explained by x_1 and x_2 alone, as can be explained by the first order sensitivity indices computed after sensitivity analysis.

```
function y=ishigami(x)
  a = 7
  b = 0.1
  s1=sin(x(:,1))
  s2=sin(x(:,2))
  x34 = x(:,3).^4
  y(:,1) = s1 + a.*s2.^2 + b.*x34.*s1
endfunction
```

In the following script, we use an increasing number of simulation points and compare the absolute error between Monte-Carlo and QMC.

```
imax=18;
s=3;
integralExact=3.5;
// See the error with Monte-Carlo
errRand=[];
for i=1:imax
    npoints=2^i;
    x=distfun_unifrnd(-%pi,%pi,npoints,s);
    y=ishigami(x);
    integralApprox=mean(y);
    errRand(i)=abs(integralApprox-integralExact);
end
// See the error with Quasi Monte-Carlo
errQMC=[];
for i=1:imax
    npoints=2^i;
    u=lowdisc_ldgen(npoints,s);
    x=2*%pi*u-%pi;
    y=ishigami(x);
    integralApprox=mean(y);
    errQMC(i)=abs(integralApprox-integralExact);
end
scf();
n=2.^(1:imax);
mcconv=10 ./sqrt(n);
qmcconv=0.1*log(n).^s./n;
plot(n,errRand,"bo-");
plot(n,mcconv,"b-");
plot(n,errQMC,"ro-");
plot(n,qmcconv,"r-");
a=gca();
a.log_flags="lln";
legend(["Monte-Carlo" "MC␣Rate" ..
"Quasi-Monte-Carlo" "QMC␣rate"]);
xtitle("Quasi␣Monte-Carlo␣versus␣Monte-Carlo",..
```

```
"Number of simulations","Absolute Error");
```

In the previous script, the constants 10 and 0.1 in `mcconv` and `qmcconv` are only there to be able to make the difference between the lines: only the slopes are meaningful in this convergence plot. The previous script produces the figure 6.2.

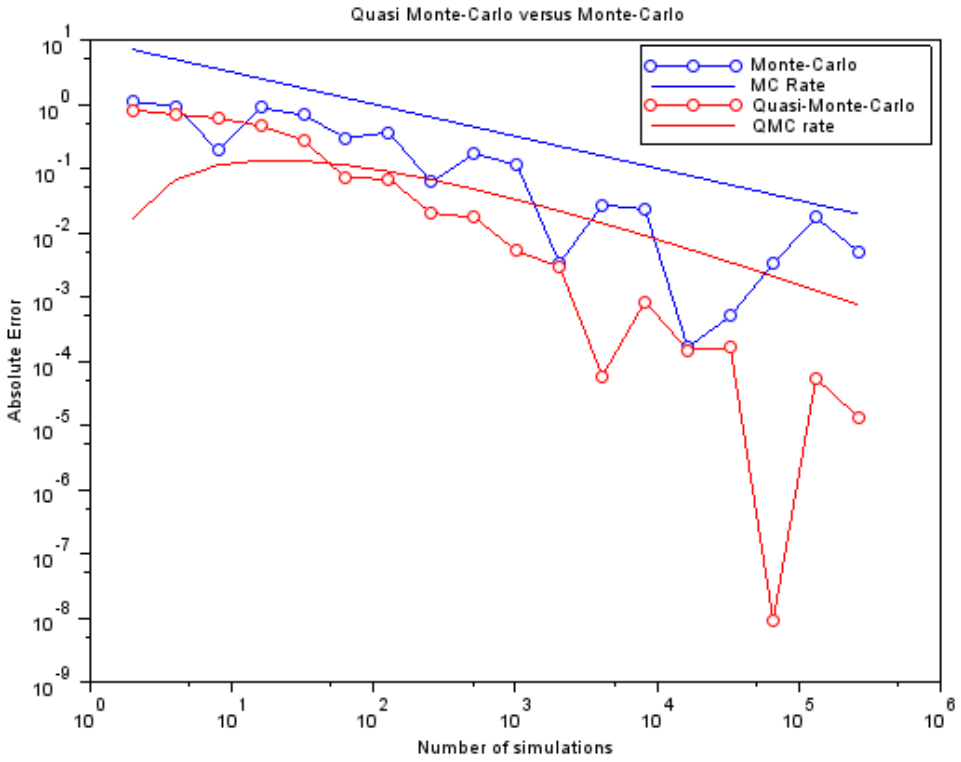

Figure 6.2: Practical convergence of Monte Carlo and Quasi-Monte Carlo with the Ishigami test function.

We can see that QMC outperforms MC in at least two senses. First, with 100 000 simulations, QMC can identify almost 5 significant digits, while Monte-Carlo has only from 1 to 3 significant digits. Second, the practical convergence rate of QMC is better than Monte-Carlo, and better than expected. We can see that it is almost equal to 1/n. This is a common situation, because the asymptotic bound given for low discrepancy sequences is just an upper bound. In general, the function is smoother than required by the low discrepancy sequence, and the achieved convergence is better than predicted by the theory.

The Ishigami is somewhere between difficult cases (where both QMC and MC fail) and easy cases (where QMC gives great results).

6.3 Efficiency of Quasi Monte-Carlo

The actual efficiency of quasi-Monte-Carlo methods over crude Monte-Carlo depends on the nature of the function to be integrated, the number of variables n and the number of dimensions s.

A typical failure of QMC involve functions which are highly oscillatory or discontinuous.

Another case of failure of QMC is for functions which are very different from an additive function. This happens when the function has high-order interations.

QMC can perform differently depending on the structure of the interactions between input variables, especially when s is large.

In [13], Sobol suggests the following. If all the variables are equally important and s is large (say, s>15), then there is no advantage in switching to quasi-Monte-Carlo. However, if all the variables are independent or if the dependence on xi decreases as i increases (in other words, the initial coordinates are the leading ones), one can expect a considerable benefit for QMC, even if s is large (from s=10 to s=100, may be 1000).

Caflish, Morokoff and Owen defined the effective dimension of a function in the superposition or in the truncation sense. Both these definitions are making use of the functional ANOVA decomposition and lead to global sensitivity indices, which normalized versions were first defined by Sobol. Functions with a small effective dimension d are easy to integrate as long as the point set used has good properties for its projections over the first d coordinates. The original Halton sequence should be quite sensitive to the effective dimension d, since we know its projections deteriorate quickly as the dimension increases.

Chapter 7

Projections

The goal of this page is to gather experiments showing bad 2D projections for low disrepancy sequences.

7.1 Halton Sequence

In this section, we present some bad 2D projections of the Halton sequence. The Halton sequence is known for having bad 2D projections and several examples are presented in the bibliography.

In the following example, we consider the projection of dimensions (28,29) as in [3]. The paper contains an error : the faulty projection is (28,29) (primes 107,109) and not (27,28) (primes 103,107).

```
u=lowdisc_ldgen ( 4096 , 29 , "halton" );
scf();
lowdisc_proj2d ( u , [28 29] );
// Compute the correlation coefficient for dimensions (28,29)
corrcoef ( [u(:,28) u(:,29)] )
```

The previous script produces the following output.

```
-->corrcoef ( [u(:,28) u(:,29)] )
 ans  =
    1.          - 0.1210675
  - 0.1210675     1.
```

The previous script also produces the figure 7.1.

The figure 7.2 presents the leaped - Halton sequence. There is no obvious problem anymore.

```
u=lowdisc_ldgen ( 4096 , 29 , "halton-leaped");
scf();
lowdisc_proj2d ( u , [28 29] );
```

In the following example, we consider the projection of Halton sequence in dimensions (39,40) as suggested by Kocis and Whiten [8].

```
u=lowdisc_ldgen ( 2000 , 40 , "halton" );
scf();
lowdisc_proj2d ( u , [39 40] );
// Compute the correlation coefficient for dimensions (39,40)
corrcoef ( [u(:,39) u(:,40)] )
```

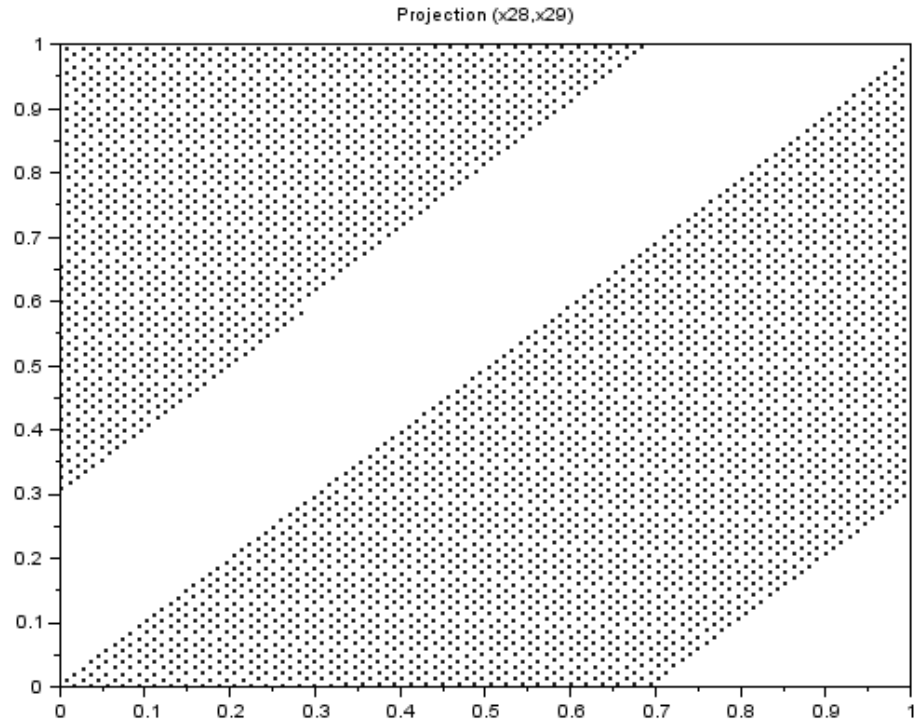

Figure 7.1: The Halton sequence - 4096 points in 29 dimensions. Projection (28,29).

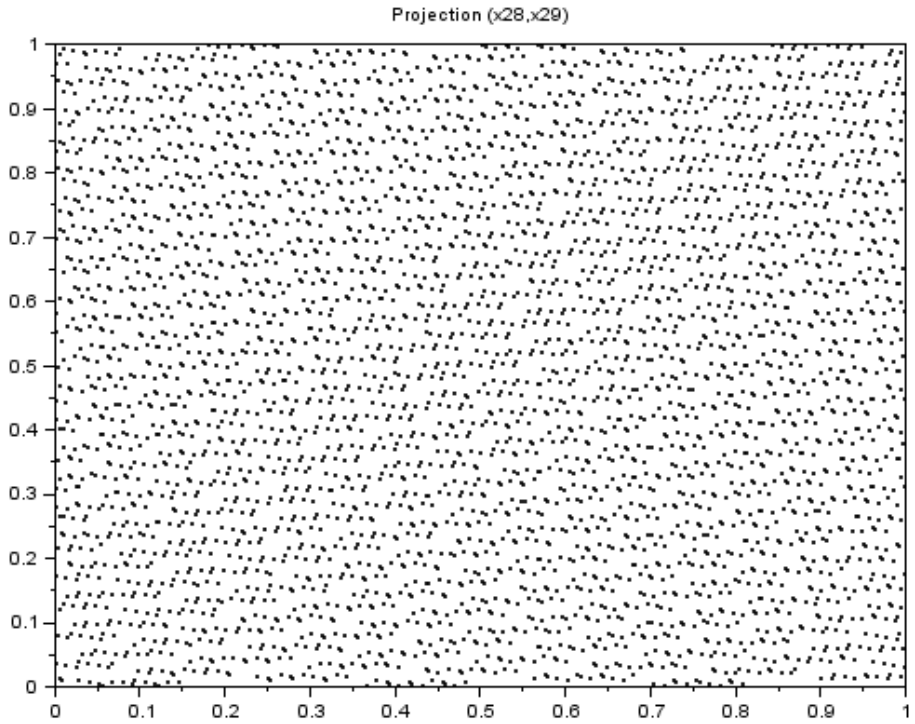

Figure 7.2: The leaped Halton sequence - 4096 points in 29 dimensions. Projection (28,29).

The previous script produces the following output.

```
-->corrcoef ( [u(:,39) u(:,40)] )
 ans  =
    1.           0.1048947
    0.1048947    1.
```

The previous script also produces the figure 7.3.

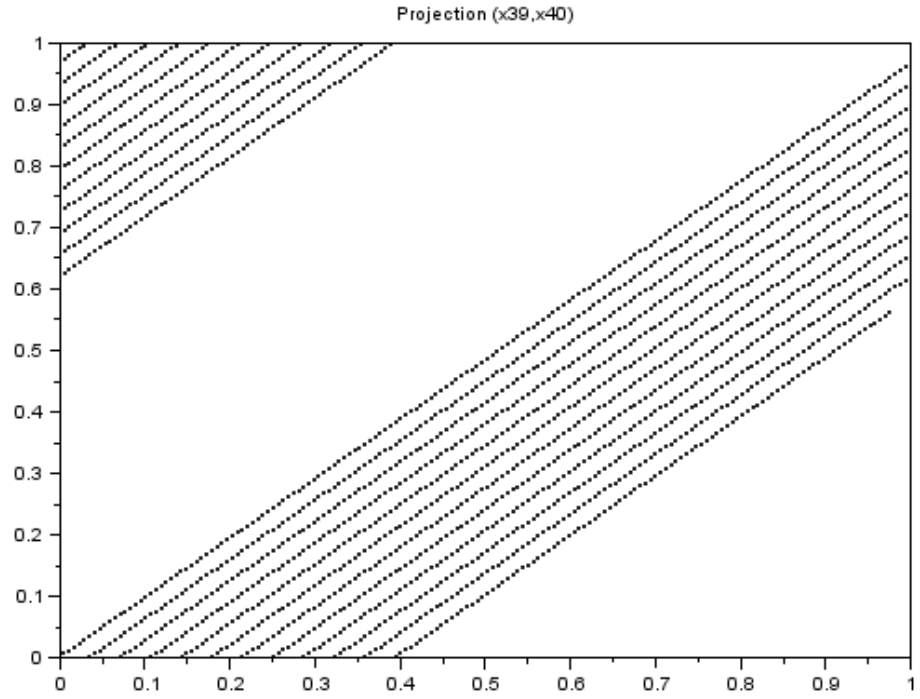

Figure 7.3: The Halton sequence - 2000 points in 40 dimensions. Projection (39,40).

The following script performs the same experiment with the leaped-Halton.

```
u=lowdisc_ldgen ( 2000 , 40 , "halton-leaped");
scf();
lowdisc_proj2d ( u , [39 40] );
```

The previous script produces the figure 7.4.

On the other hand, we emphasize that a leaped Faure sequence is not a (0,s)-sequence anymore. Therefore, there is no guarantee that the leaped Faure sequence is a low discrepancy sequence.

In the following example, we search the worst Halton projection for 2000 points in dimensions 40. In order to evaluate the projection, we compute the correlation coefficients. The projections which have the largest non-diagonal correlation coefficients are the worst.

```
u=lowdisc_ldgen ( 2000 , 40 , "halton" );
// Compute correlation coefficients
c = corrcoef ( u );
```

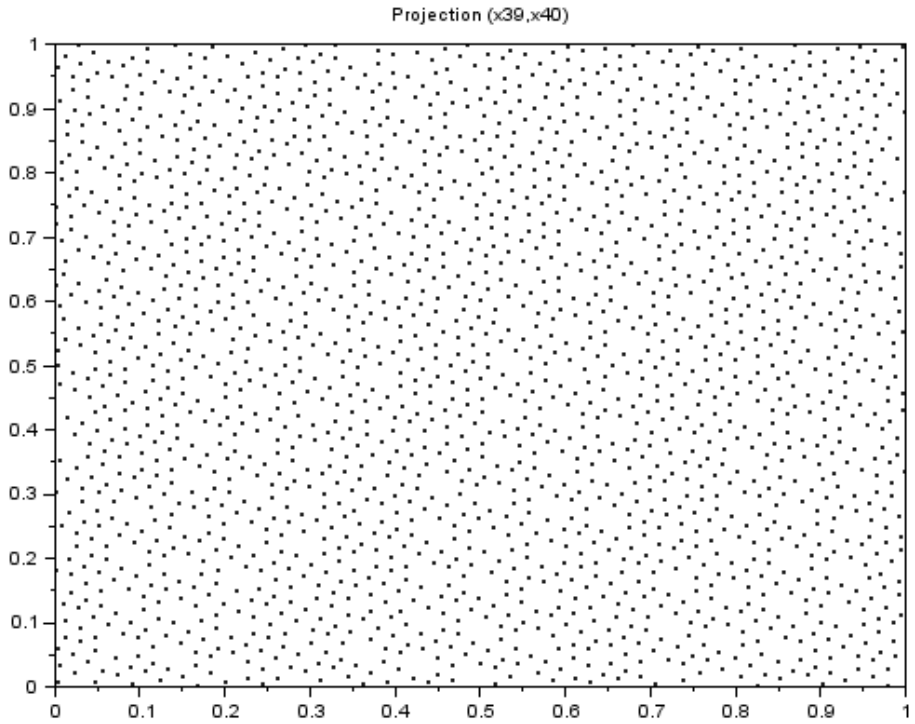

Figure 7.4: The leaped Halton sequence - 2000 points in 40 dimensions. Projection (39,40).

```
// Get the maximum correlation coefficient
// but remove the diagonal (unity) terms
[m,k] = max(abs(c-eye(c)))
// This is projection (35,36) :
k
// Get the correlation coefficient for this projection
c(k(1),k(2))
// Print this projection
scf();
lowdisc_proj2d ( u , k );
```

The previous script produces the figure 7.5.

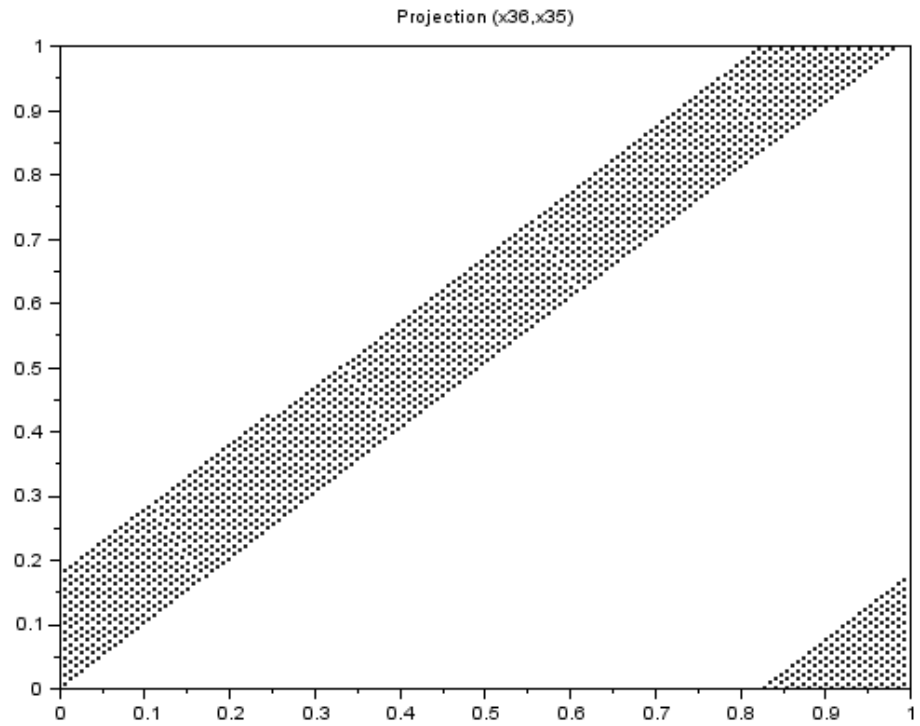

Figure 7.5: The Halton sequence - 2000 points in 40 dimensions. Projection (35,36).

7.2 Faure Sequence

In this section, we present some bad 2D projections of the Faure sequence.

In the following example, we search for the worst projection of 2000 points of the Faure sequence in dimension 40. The worst projection is relatively good, which indicates how Faure's sequence is much better than Halton's sequence.

```
u=lowdisc_ldgen ( 2000 , 40 , "faure" );
```

```
// Compute correlation coefficients
c = corrcoef ( u );
// Get the maximum correlation coefficient
// but remove the diagonal (unity) terms
[m,k] = max(abs(c-eye(c)))
// This is projection (20,21) :
k
// Get the correlation coefficient for this projection
c(k(1),k(2))
// Print this projection
scf();
lowdisc_proj2d ( u , k );
```

The previous script produces the figure 7.6.

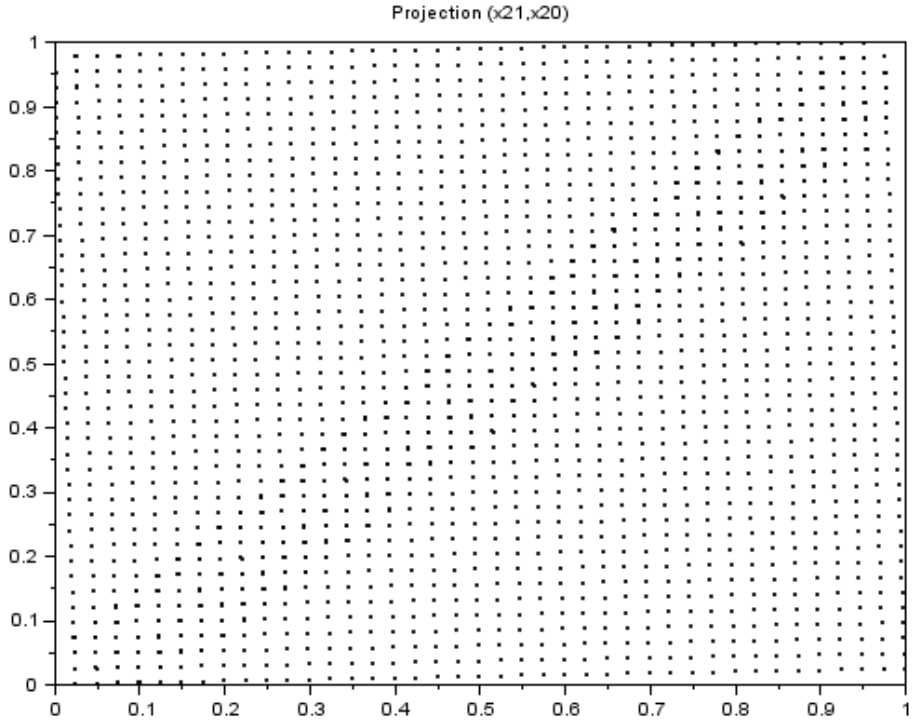

Figure 7.6: The Faure sequence - 2000 points in 40 dimensions. Projection (20,21).

This is consistent with the fact that, in s=40 dimensions, the Faure sequence uses the prime number b=41 as the base. Moreover, the Faure sequence in dimension s=40 is a (0,40)-sequence in base b=41. Consider now the projection (i,j), for i different from j. Consider elementary intervals from 41 subdivisions in dimension i, 41 subdivisions in dimension j, and 1 subdivision in the other dimensions. There are $41^2 = 1681$ intervals, each with volume 1/1681. Hence, if we consider a set of 1681 points, each such constructed elementary interval has one point. Since the previous figure has 2000>1681 points, all 2D projections of the Faure sequence in dimension s=40 are quite well

distributed.

In the following example, we consider 300 points of the Faure sequence in dimension 40, and consider the projection (39,40) as suggested by Kocis and Whiten [8].

```
u=lowdisc_ldgen ( 300 , 40 , "faure" );
scf();
lowdisc_proj2d ( u , [39 40] );
```

The previous script produce the figure 7.7.

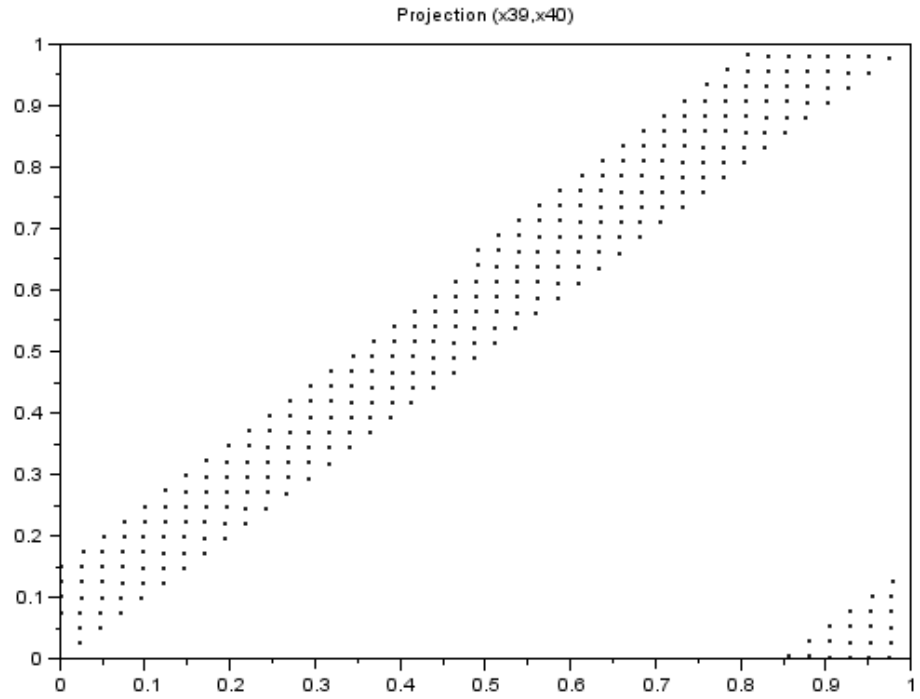

Figure 7.7: The Faure sequence - 300 points in 40 dimensions. Projection (39,40).

Since the number of points is 300 which is much smaller than $41^2 = 1681$, the Faure sequence cannot evenly distribute the points in the space.

We check what is the behavior with more favorable parameters.

```
dim = 40;
nsimmin = 300;
lds = lowdisc_new("faure");
lds = lowdisc_configure(lds,"-dimension",dim);
base = lowdisc_get(lds,"-faureprime");
[nsim,skip,leap] = lowdisc_fauresuggest ( dim , base , nsimmin );
lds = lowdisc_configure(lds,"-skip",skip);
lds = lowdisc_configure(lds,"-leap",leap);
lds = lowdisc_startup (lds);
[lds,u]=lowdisc_next(lds,nsim);
```

```
lds = lowdisc_destroy(lds);
scf();
lowdisc_proj2d ( u , [39 40] );
```

We see that the number of experiments has increased from 300 to 68921, which is significantly more.

```
-->size(u)
 ans  =
    68921.    40.
```

The figure 7.8 shows that there the correlation problem is greatly reduced, although the points seems to be located on lines.

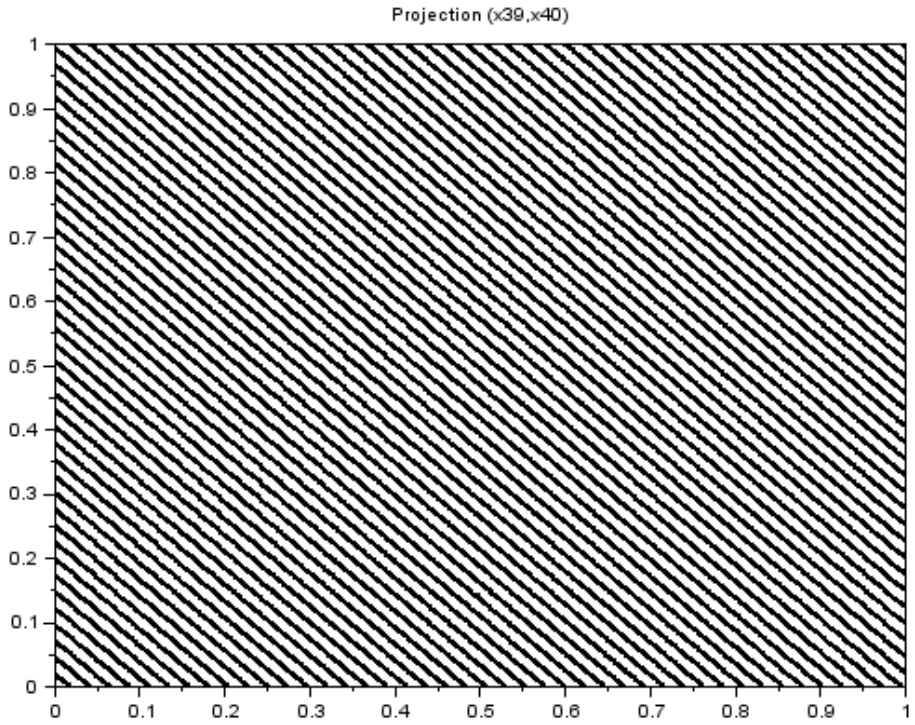

Figure 7.8: The Faure sequence - 68921 points in 40 dimensions. Projection (39,40).

The number of points $68921 = 41^3$, which guarantees good 3D projections.

In order to keep the number of experiments constant, we might be interested in using the `leap` option. Following Kocis and Whithen [8], we may try a leaped Faure sequence. We try the leap L=5.

```
lds = lowdisc_new("faure");
lds = lowdisc_configure(lds,"-dimension",40);
lds = lowdisc_configure(lds,"-leap",5);
lds = lowdisc_startup (lds);
[lds,u] = lowdisc_next (lds,300);
```

```
lds = lowdisc_destroy(lds);
scf();
lowdisc_proj2d ( u , [39 40] );
```

The previous script produces the figure 7.9.

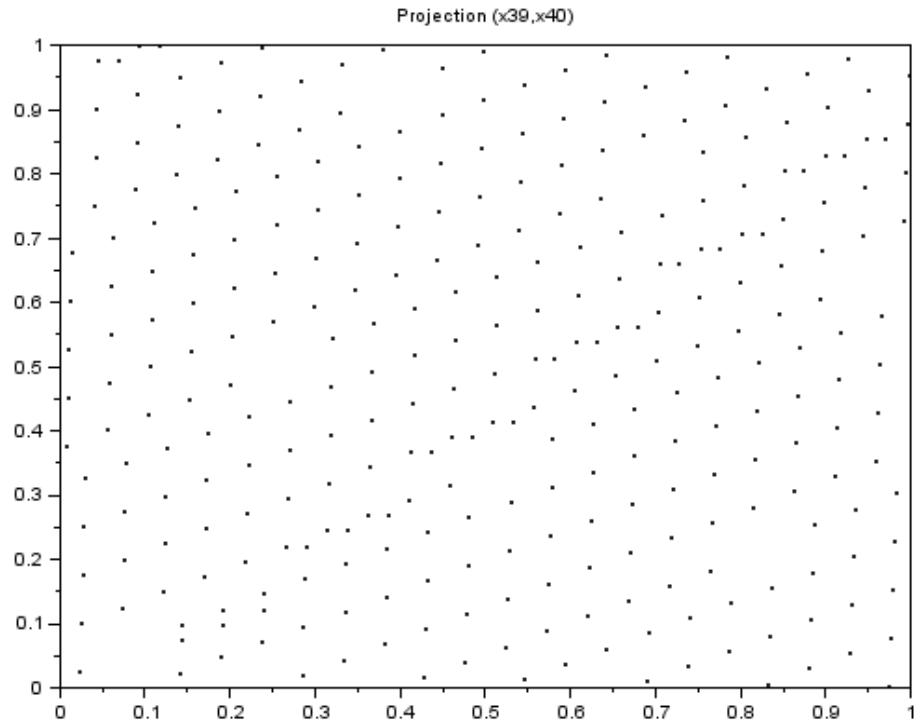

Figure 7.9: The Faure sequence - 300 points in 40 dimensions. Projection (39,40).

On the other hand, we emphasize that a leaped Faure sequence is not a (0,s)-sequence anymore. Therefore, there is no guarantee that the leaped Faure sequence is a low discrepancy sequence.

7.3 Sobol Sequence

In this section, we present some bad 2D projections of the Sobol sequence.

In the following example, we consider 1000 points of the Sobol sequence in 23 dimensions and check the projection (16,23).

```
u=lowdisc_ldgen(1000,30,"sobol");
scf();
lowdisc_proj2d(u,[16 23])
xtitle("Sobol : 1000 points","X16","X23")
```

The previous script produces the figure 7.10.

In the following example, we consider 1000 points of the Sobol sequence in 36 dimensions and check the projection (11,36).

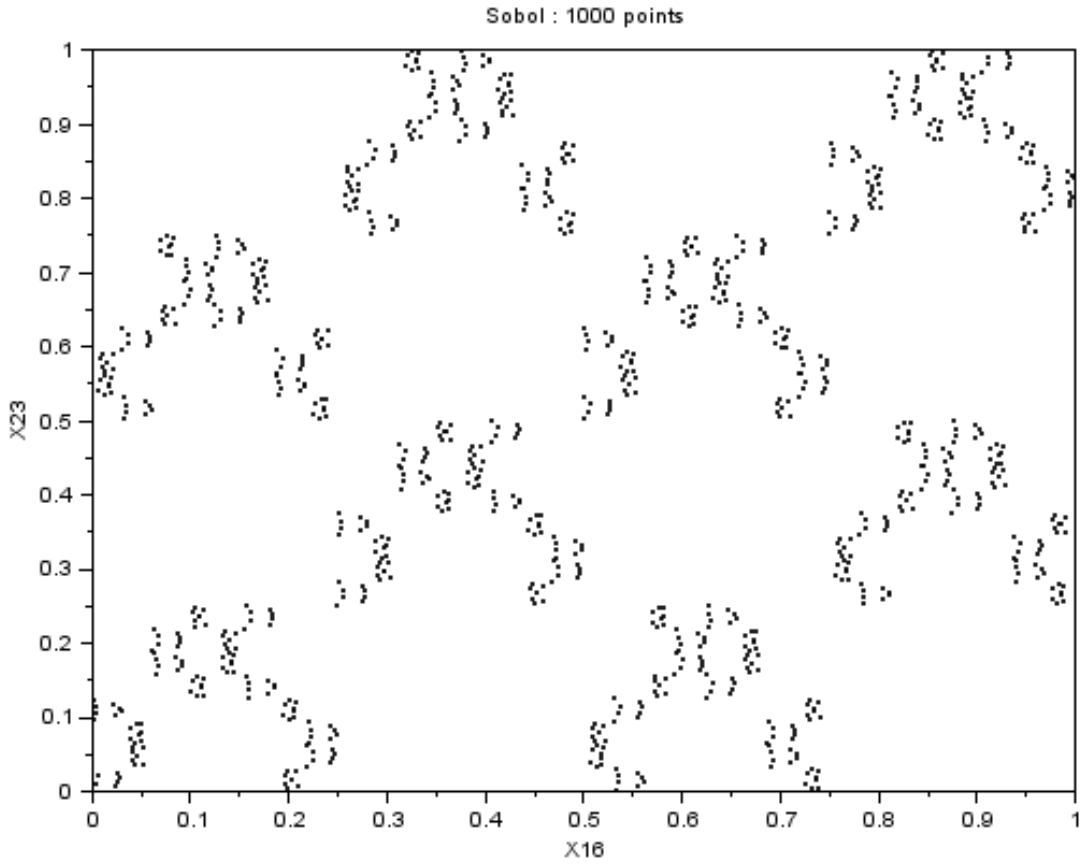

Figure 7.10: The Sobol sequence - 1000 points in 30 dimensions. Projection (16,23).

```
u=lowdisc_ldgen(1000,36,"sobol");
scf();
lowdisc_proj2d(u,[11 36])
xtitle("Sobol␣:␣1000␣points","X11","X36")
```

The previous script produces the figure 7.11.

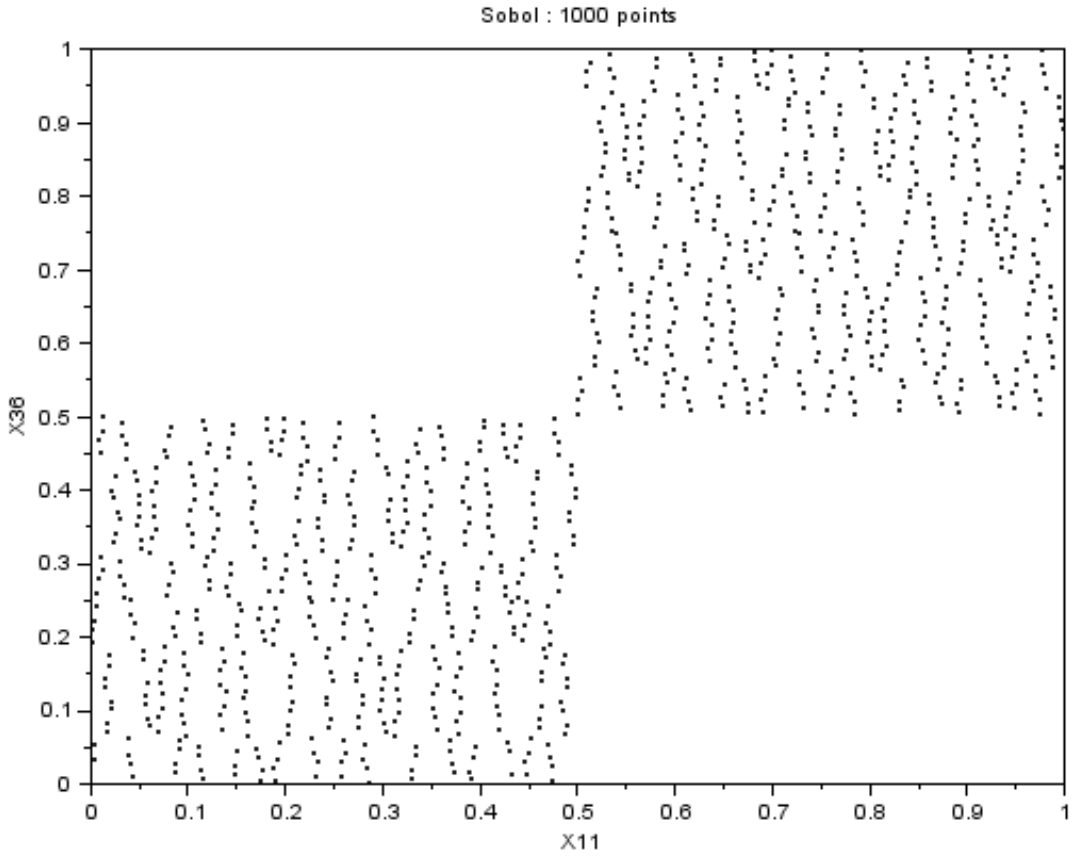

Figure 7.11: The Sobol sequence - 1000 points in 36 dimensions. Projection (11,36).

In the following example, we plot the projection (5,22) of 2^{12} and 2^{13} points of the Sobol sequence in s=22 dimensions. This example is presented in [7]. We consider $2^{m-t} = 2^4 \times 2^2 = 2^6$ elementary intervals, with 2^4 divisions in the dimension x_5 and 2^2 divisions in the dimension x_{22}. For this projection, the quality parameter t is 7. Hence, we need $2^m = 2^{6+7} = 2^{13}$ points to make so that each elementary interval contain $2^t = 2^7$ points.

```
h=scf();
subplot(2,1,1);
//
s=22;
m=12;
r = [4 2];
N=b^m;
u=lowdisc_ldgen(N-1,s,"sobol");
u=[zeros(1,s);u];
```

```
plot(u(:,5),u(:,22),"b.")
lowdisc_plotelembox([b b],r)
xtitle(msprintf("Volume=1/2^%d, %d Points",m,N),"X5","X22")
//
subplot(2,1,2);
m=13;
N=b^m;
u=lowdisc_ldgen(N-1,s,"sobol");
u=[zeros(1,s);u];
plot(u(:,5),u(:,22),"b.")
lowdisc_plotelembox([b b],r)
xtitle(msprintf("Volume=1/2^%d, %d Points",m,N),"X5","X22")
//
h.children(1).isoview="off";
h.children(2).isoview="off";
h.children(1).children($).children.mark_size=2;
h.children(2).children($).children.mark_size=2;
```

The previous script produces the figure 7.12.

7.4 Niederreiter (base 2) Sequence

In this section, we consider 1000 points of the Niederreiter (base 2) sequence in 26 dimensions, and consider the projection (25,26).

```
u=lowdisc_ldgen(1000,30,"niederreiter");
scf();
lowdisc_proj2d(u,[25 26])
```

The previous script produce the figure 7.13.

7.5 Two-dimensional correlations with increasing dimensions

In this section, we present how the linear correlation coefficient between dimensions of the sequence deteriorates when the number of dimensions increases. Given a number of dimensions and 1000 points, we are interested in the worst 2D projection, as defined by the largest correlation coefficient.

7.5.1 Numerical experiments

The following function test for low discrepancy sequences for dimensions s from 2 to smax. It generates a low discrepancy point set for n points and computes the correlation coefficient between columns of the matrix. Then the diagonal terms are set to zero and the largest entry is computed. We assume here that this identifies the worst 2D projection.

```
function m=worstCorrelation(n,smax,seqname)
    for s=2:smax
```

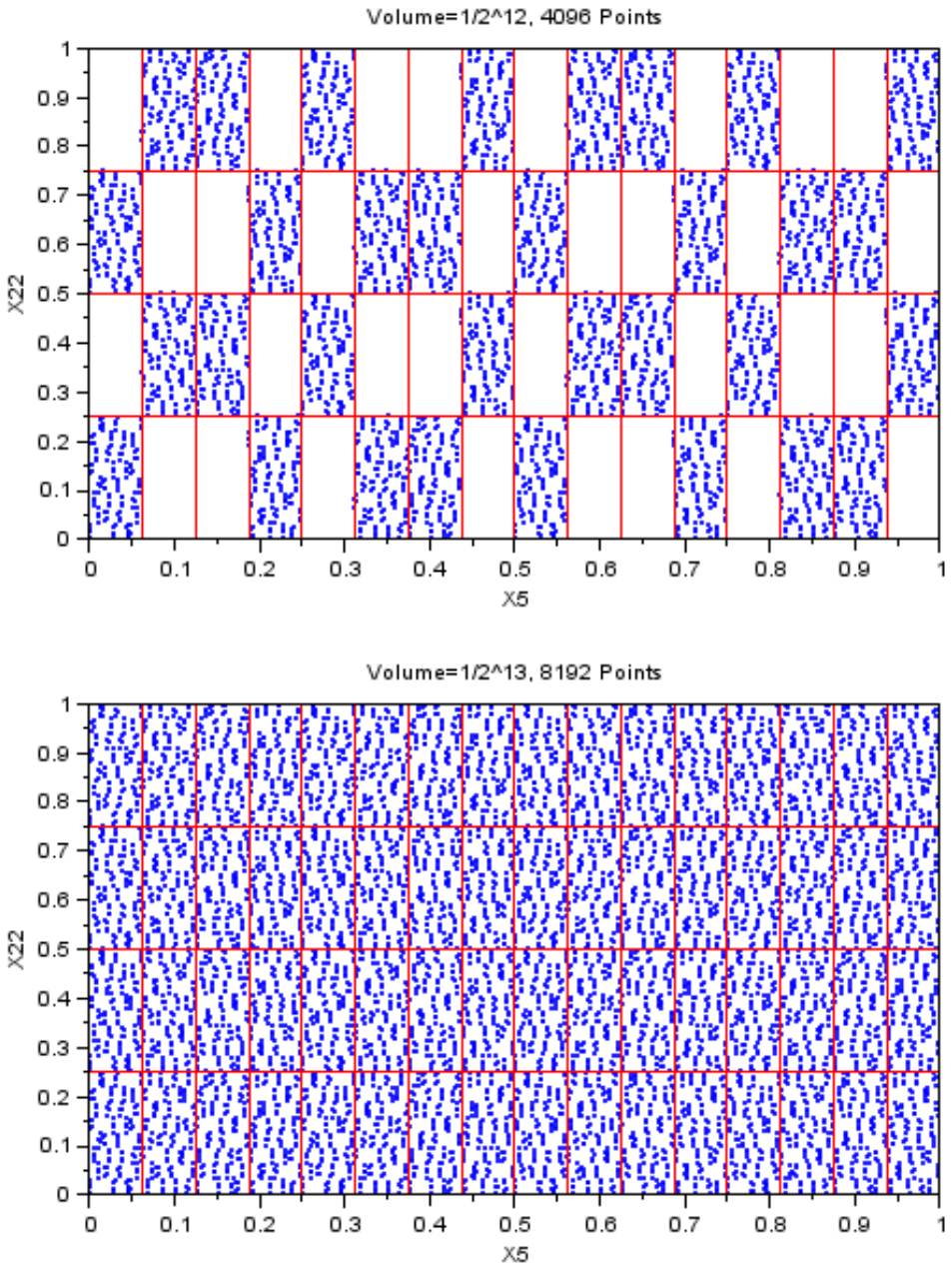

Figure 7.12: The Sobol sequence - 2^{12} (top) and 2^{13} (bottom) points in 22 dimensions. Projection (5,22).

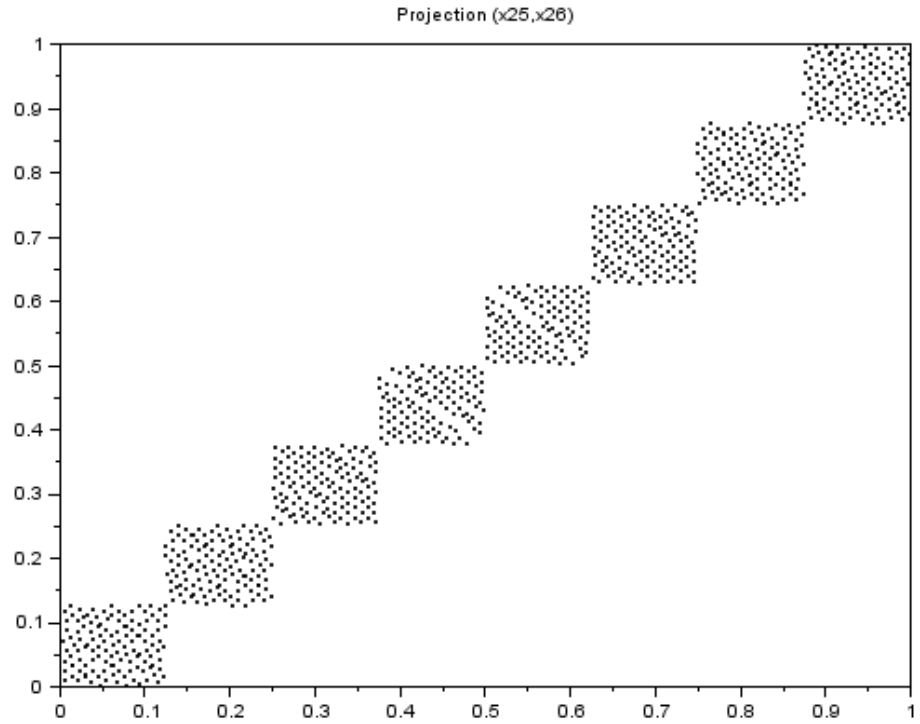

Figure 7.13: The Niederreiter (base 2) sequence - 1000 points in 30 dimensions. Projection (25,26).

```
        u=lowdisc_ldgen(n,s,seqname);
        c = corrcoef ( u );
        m(s-1) = max(abs(c-eye(c)));
    end
endfunction
```

The following script tests the Halton, Sobol, Faure and Niederreiter (optimal base) sequences.

```
n=1000;
smax=50;
scf();
m=worstCorrelation(n,smax,"halton");
plot(2:smax,m,"r-")
m=worstCorrelation(n,smax,"sobol");
plot(2:smax,m,"b-")
m=worstCorrelation(n,smax,"faure");
plot(2:smax,m,"g-")
m=worstCorrelation(n,smax,"niederreiter");
plot(2:smax,m,"k-")
legend(["Halton","Sobol","Faure",..
"Niederreiter"]);
a=gca();
a.children(1).legend_location="in_upper_left";
xtitle("1000␣Points","Number␣of␣dimensions",..
"Maximum␣Correlation␣Coefficient")
```

The previous script produces figure 7.14. This shows that, according to this test, the Niederreiter sequence in base 2 performs well for $s < 15$, but rapidely deteriorates for $s \geq 15$. Notice that that optimal base of Niederreiter sequence is known by our implementation only for $s < 13$. As expected, the Halton sequence has poor 2D projections, especially for $s \geq 20$ where the correlation coefficient becomes larger than 0.1. The Sobol sequence is better than Halton, but can have poor 2D projections, especially for $s \geq 35$. However, the projections for $s < 25$ are the best of this test. The Faure sequence appears to be the most robust sequence in this test for high dimensions.

The following script tests various Halton sequences.

```
n=1000;
smax=50;
scf();
m=worstCorrelation(n,smax,"halton");
plot(2:smax,m,"r-")
m=worstCorrelation(n,smax,"sobol");
plot(2:smax,m,"b-")
m=worstCorrelation(n,smax,"faure");
plot(2:smax,m,"g-")
m=worstCorrelation(n,smax,"niederreiter");
plot(2:smax,m,"k-")
legend(["Halton","Sobol","Faure",..
"Niederreiter"]);
a=gca();
a.children(1).legend_location="in_upper_left";
```

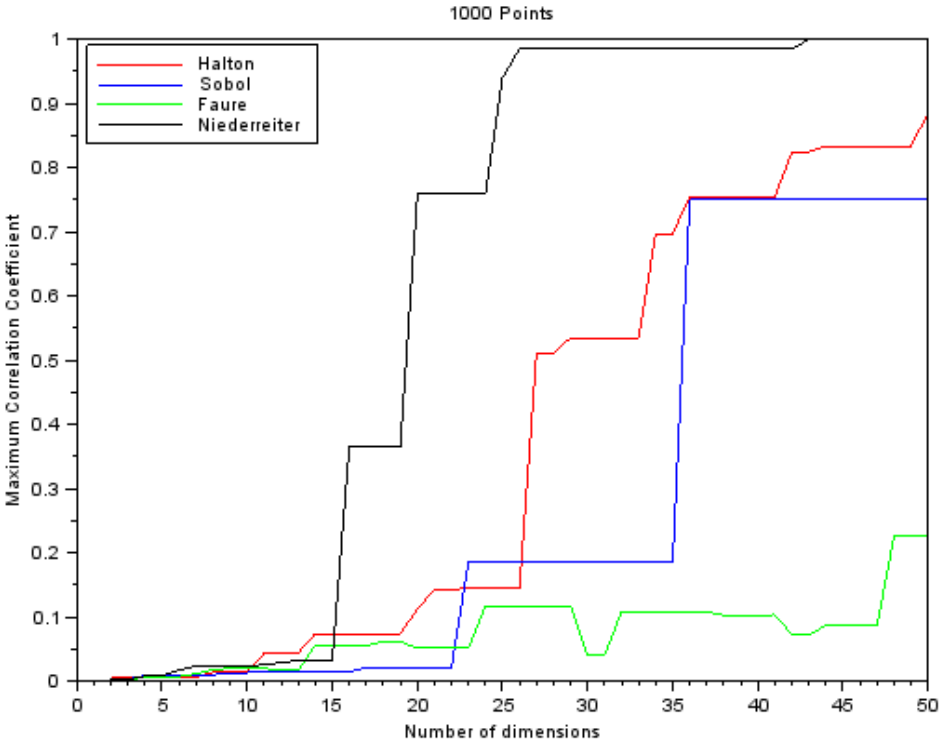

Figure 7.14: The maximum correlation coefficient vs the number of dimensions for various low discrepancy sequences.

```
xtitle("1000␣Points","Number␣of␣dimensions",..
"Maximum␣Correlation␣Coefficient")
```

The previous script produces figure 7.15. This shows that, according to this test, the reverse Halton sequence has almost the same poor 2D projections as the Halton sequences (the two red lines are almost the same). The leaped Halton sequence generally has better projections than Halton, but can also have larger correlations (s=16, 24, 42). The scrambled (Kocis-Whiten) Halton sequence is the most robust sequence in this test, at the exception of s=25, for which the correlation is slightly larger. These graphics suggest that the scrambled Halton sequence is the best of this set of tests.

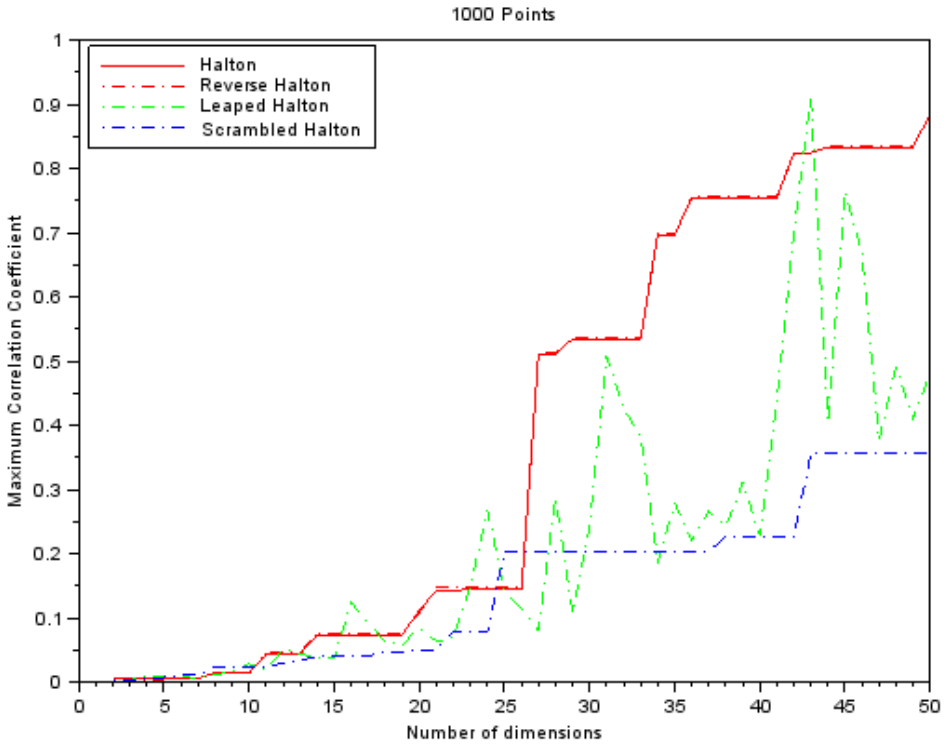

Figure 7.15: The maximum correlation coefficient vs the number of dimensions for various low discrepancy sequences.

7.5.2 Conclusions

These tests suggest that the Sobol sequence is better for small dimensions ($s < 25$) and the Faure sequence is better for large dimensions ($s > 25$). Weirdly, the Niederreiter sequence performs poorly in this test. However, this confirms the poor projections we have presented in the section 7.4.

We do not see any practical difference between the Reverse Halton sequence and the Halton sequence in terms of 2D projections. the scrambled On the other hand, the scrambled Halton sequence is generally better than a classical Halton sequence.

Chapter 8

Thanks

Many thanks to Allan Cornet and Pierre Marechal for their help during the development of the libray. We thank John Burkardt for his technical help on the source code we used. Thanks to Jean-Philippe Chancelier for finding bugs in the source code of a gateway.

Bibliography

[1] E. Braaten and G. Weller. An improved low-discrepancy sequence for multidimensional quasi-Monte-Carlo integration. *J. Comput. Phys.*, 33:249–258, 1979.

[2] P. Bratley and B. L. Fox. Algorithm 659: Implementing Sobol's quasirandom sequence generator. *ACM Transactions on Mathematical Software*, 14(1):88–100, 1988.

[3] H. Chi, M. Mascagni, and T. Warnock. On the optimal halton sequence. *Math. Comput. Simul.*, 70(1):9–21, September 2005.

[4] Paul Glasserman. *Monte-Carlo methods in Financial Engineering*. Springer, 2003.

[5] J. H. Halton. Algorithm 247: Radical-inverse quasi-random point sequence. *Commun. ACM*, 7(12):701–702, December 1964.

[6] J.M. Hammersley. Monte carlo methods for solving multivariate problems. *Annals of the New York Academy of Sciences*, 86:844–874, 1960.

[7] Stephen Joe and Frances Y. Kuo. Constructing sobol sequences with better two-dimensional projections. *SIAM J. Sci. Comput.*, 30(5):2635–2654, August 2008.

[8] Ladislav Kocis and William J. Whiten. Computational investigations of low-discrepancy sequences. *ACM Trans. Math. Softw.*, 23(2):266–294, June 1997.

[9] Christiane Lemieux. *Monte Carlo and Quasi-Monte Carlo Sampling*. Springer Series in Statistics, 2009.

[10] Harald Niederreiter. Low-discrepancy and low-dispersion sequences. *Journal of Number Theory*, 30:51–70, 1988.

[11] Harald Niederreiter. *Random number generation and quasi-Monte Carlo methods*, volume 63. CBMS-NSF Series in Applied Mathematics, SIAM, Philadelphia, 1992.

[12] Ken Seng, Tan Phelim, and P. Boyle. Applications of scrambled low discrepancy sequences to exotic options, 1997.

[13] Ilya M. Sobol'. *A Primer for the Monte Carlo Method*. CRC Press, 1994.

[14] Eric Thiémard. *Sur le calcul et la majoration de la discrepance a l'origine*. PhD thesis, Lausanne, EPFL, 2000.

[15] J. G. van der Corput. Verteilungsfunktionen. *Proc. Ned. Akad. v. Wet.*, 38:813–821, 1935.