Searching Parameters for the Polynomial Arithmetic Analogue of Hickernell Sequences

by

Nicholas A. Mc Neilly

A research paper
presented to the University of Waterloo
in partial fulfillment of the
requirement for the degree of
Master of Mathematics
in
Computational Mathematics

Supervisor: Prof. Christiane Lemieux

Waterloo, Ontario, Canada, 2011

Abstract

The use of polynomial arithmetic has proven to be quite useful in several areas of mathematics. In 2002, Shu Tezuka presented a method for building low-discrepancy point sets by uniting polynomial arithmetic, finite fields and the basis for Hickernell sequences. The method produces a class of digital (t, s)-sequences which is a family of constructions that has been shown to be successful when used for computing numerical integrals of selected multi-dimensional problems. This success is closely related to the quality of the points and a known technique for measuring quality is described. This technique is exploited in conjunction with the leveraging of polynomial properties to search for parameters that result in high quality point sets for this implementation.

Acknowledgements

I wish to thank my supervisor, Professor Christiane Lemieux for her understanding, patience and guidance in conducting this research.

I would also like to acknowledge my family and friends for their unconditional commitment to support me and for providing much needed motivation and inspiration.

Dedication

To the Bethel Methodist Church in St. Paul's, Grenada.

Table of Contents

Chapter 1

Introduction

There are various constructions of deterministic multi-dimensional sequences that belong to the collection of quasi-Monte Carlo methods. The polynomial arithmetic analogue of Hickernell sequences (polynomial Hickernell sequences) is one such method and is particularly attractive because its quality can be readily investigated. We use an existing technique to investigate parameters resulting in high quality sequences and examine the performance when applied to problems in finance. The material is presented with the intention to always build on previously introduced content but there are a few areas where the reader may elect to skip ahead for additional information on a particular topic. The analysis of polynomial arithmetic provides us with an avenue to effectively assess the quality of polynomial Hickernell sequences, and in addition to the manner in which this can be achieved, we show the resulting performance.

The polynomial Hickernell sequence was presented by Shu Tezuka [7] in 2002 and it constitutes a class of digital (t, s)-sequences. Hickernell sequences were introduced before polynomial Hickernell sequences but used integer operations and devising a method for measuring quality proved to be very challenging. The nature of the analysis of polynomial arithmetic operations is far simpler than its integer equivalent and this is what propelled Tezuka to explore this construction. This topic was also considered by Niederreiter independently (see [7] and included references). With this more welcome analysis, it is easier to investigate desirable initialisation parameters for polynomial Hickernell sequences. These parameters are of interest because quasi-Monte Carlo constructions have been known to accelerate the process of numerical integration for selected problems in high dimensions. Our goal is to find good sequences that may have a meaningful impact in this regard.

The method used for evaluating quality, taken from [4], is based on linear algebra and incurs a great deal of computational overhead. Consequently, the C programming language was chosen for this project. It has superior processing capabilities which are needed to speedily perform extensive polynomial arithmetic and linear algebra operations. We build a command line interface that gives a user the ability to input the dimension of the desired sequence and subsequently, provides a list containing the top options emerging from a hybrid random search. The user will then be allowed to choose from these options and have the corresponding set of points printed to file. We use this interface to generate points for real applications and give a comparative overview of the output.

This paper has six sections and is structured in such a way that the material from the earlier ones is necessary to have a full grasp of the content that comes later. Chapter 2 contains the required background information for understanding polynomial Hickernell sequences. There are some involved definitions and the reader is not required to internalise all of these, but they serve as a good reference and will be needed at times. The critical terms will be clearly identified. The definition and algorithm for polynomial Hickernell sequences is given in Chapter 3. Searching parameters, the focal point of this paper, appears in Chapter 4. We give a detailed description of the parameters, the technique used to examine quality and the method for executing the search. The results are applied to a mortgage-backed security and an Asian option in Chapter 5, where the comparison between our implementation and other known constructions is illustrated graphically. Final remarks are given in Chapter 6 and this includes highlights of areas that are candidates for further exploration.

Chapter 2

Preliminaries

2.1 Monte Carlo Method

The polynomial Hickernell sequence is a method that falls under the umbrella of Quasi-Monte Carlo methods. Quasi-Monte Carlo methods are different from Monte Carlo methods and this difference will be explained in the next section. For now, we will state what is meant by the Monte Carlo method through referencing [3].

> **Monte Carlo method**: The use of random sampling as a tool to produce observations on which statistical inference can be performed to extract information about a system.

Essentially, Monte Carlo methods seek to use the results of random sampling to help solve problems within various disciplines such as the physical sciences, finance and others.

A common application of the Monte Carlo method is called *Monte Carlo integration*. Since this is one of the key reasons for investigating and implementing the polynomial Hickernell sequence, we will introduce the term as it appears in [3]:

> **Monte Carlo Integration**: Special use of the Monte Carlo method where we randomly sample uniformly over some domain $V \subseteq \mathbb{R}^s$ and use the produced sample $\{x_1, x_2, \dots, x_N\}$ to construct an estimator for an integral of the form $\int_V f(x)\,\mathrm{d}x$ where f is a real-valued function over V.

Example 2.1. If we wanted to compute $\int_0^1 f(x)\,\mathrm{d}x$ where $f(x) = \sin x$ then as an alternative to the usual analytical method, we can approximate this value using the Monte Carlo method as follows:

1. Generate N random values $x_1, x_2, \ldots, x_N$ from the standard uniform distribution $\big(U(0,1)\big)$
2. Compute $\overline{f} = \frac{1}{N}\sum_{i=1}^{N} f(x_i)$

The estimated values for Monte Carlo integration improves as the number of samples increases. A closed form solution for the integral described above is fairly straightforward but there are instances where this is not so. For these cases, Monte Carlo integration proves to be quite useful.

2.2 Quasi-Monte Carlo Method and Relevant Definitions

The quasi-Monte Carlo method is based on *low-discrepancy sampling* and in [3], an informal way to think of such a sample is given:

> A **low-discrepancy sample** is one whose points are distributed in a way that approximates the uniform distribution as closely as possible.

These low-discrepancy samples are used for multi-dimensional numerical integration with the half-open s-dimensional unit cube $I^s = [0,1)^s$ for $s \geq 1$. We focus on the unit cube because other domains can usually be appropriately normalised to utilise these points.

In [7], Tezuka recalls the definition of what is known as the *star discrepancy* and uses this to formally define a low-discrepancy sequence:

Definition 2.1. For a point set $P_N = \{X_0, X_1, \ldots, X_{N-1}\}$ of N points in $I^s = [0,1)^s$ and a subinterval J of I_s, we define $E(J,N) = \frac{A(J,N)}{N} - V(J)$ where $A(J,N)$ is the number of n, $0 \leq n \leq N-1$, with $X_n \in J$ and $V(J)$ is the volume of J. Then the **(star) discrepancy** of P_N is defined by

$$D_N^{(s)}(P_N) = \sup_J |E(J,N)|$$

where the supremum is taken over all subintervals J of the form $\prod_{i=1}^{s}[0, y_i)$. For an infinite sequence, X, of points in I^s, let $D_N^{(s)}(X)$ denote the discrepancy of the point set of its first N points.

A **low-discrepancy** sequence is an infinite sequence X such that

$$D_N^{(s)}(X) \leq C_s \frac{(\log N)^s}{N}$$

where $N \geq 2$ and C_s is an absolute constant depending only on the dimension.

This definition has been included to illustrate that there is a precise way to identify a low-discrepancy sequence.

A key difference between the quasi-Monte Carlo method and the Monte Carlo method is that the points for the former need not be independently distributed. In fact, it is often the case that these points are computed in a deterministic fashion.

The quasi-Monte Carlo method is achieved by replacing the pure random sampling component of the Monte Carlo method with low-discrepancy samples. This exchange is particularly useful when applied to Monte Carlo integration. The convergence rate for low-discrepancy constructions is superior to that of the Monte Carlo method for certain types of functions and Definition 2.1 can be used to support this.

There is some terminology that will be referenced throughout this paper and as such we will introduce some key definitions by following [5]:

Definition 2.2. Let $b \geq 2$, $s \geq 1$, and $0 \leq t \leq m$ be integers. Then a point set consisting of b^m points of I^s forms a $(\mathbf{t}, \mathbf{m}, \mathbf{s})$**-net in base** b if every subinterval $J = \prod_{i=1}^{s}[a_i b^{-d_i}, (a_i + 1)b^{-d_i})$ of I^s with integers $d_i \geq 0$ and $0 \leq a_i < b^{d_i}$ for $1 \leq i \leq s$ (a so called elementary interval) and of volume b^{t-m} contains exactly b^t points of the point set.

Definition 2.3. Let $b \geq 2$, $s \geq 1$, and $t \geq 0$ be integers. Then a sequence $y_0, y_1, y_2, \ldots$ of points in I^s is a $(\mathbf{t}, \mathbf{s})$**-sequence in base** b if for all $k \geq 0$ and $m \geq t$ the point set consisting of y_n with $kb^m \leq n < (k+1)b^m$ forms a (t, m, s)-net in base b.

Definitions 2.2 and 2.3 refer to three very important parameters:

t : an integer indicator representing the quality of a point set. Small values of t signify better quality sets. For (t, m, s)-nets, $0 \leq t \leq m$ whereas for (t, s)-sequences, $t \geq 0$

m : an integer used when describing nets over base b to indicate the size of the point set. This size corresponds to a set of b^m points

s : the number of dimensions for the point set

These terms will be used frequently and the reader is encouraged to pay close attention to their meanings.

In the definition that follows, we present a fairly general description for the construction of a *digital net* over an arbitrary finite field $\mathbb{F}_q$ where q is a prime power. In this paper, our q will be prime and the finite field used will be $\mathbb{Z}_q$. However, the definition is given in terms of an arbitrary finite field for completeness.

Definition 2.4. Let q be a prime-power and let $s \geq 1$ and $m \geq 1$ be integers. We consider the following construction principle for point sets P consisting of q^m points in I^s. We choose:

1. bijections $\psi_r : \mathbb{Z}_q = \{0, 1, \dots, q-1\} \to \mathbb{F}_q$ for $0 \leq r \leq m-1$
2. bijections $\eta_j^{(i)} : \mathbb{F}_q \to \mathbb{Z}_q$ for $1 \leq i \leq s$ and $1 \leq j \leq m$
3. elements $c_{jr}^{(i)} \in \mathbb{F}_q$ for $1 \leq i \leq s$, $1 \leq j \leq m$ and $0 \leq r \leq m-1$

For $n = 0, 1, \dots, q^m - 1$ let

$$n = \sum_{r=0}^{m-1} a_r(n) q^r \quad \text{with all}\, a_r(n) \in \mathbb{Z}_q$$

be the digit expansion of n in base q. We put

$$x_n^{(i)} = \sum_{j=1}^{m} y_{nj}^{(i)} q^{-j} \quad \text{for } 0 \leq n < q^m \text{ and } 1 \leq i \leq s$$

with

$$y_{nj}^{(i)} = \eta_j^{(i)} \left(\sum_{r=0}^{m-1} c_{jr}^{(i)} \psi_r(a_r(n)) \right) \in \mathbb{Z}_q \quad \text{for } 0 \leq n < q^m,\ 1 \leq i \leq s, \text{ and } 1 \leq j \leq m$$

Then the point set

$$x_n = (x_n^{(1)}, x_n^{(2)}, \ldots, x_n^{(s)}) \in I^s \quad \text{for } n = 0, 1, 2, \ldots, q^m - 1$$

is called a **digital net** constructed over $\mathbb{F}_q$.

Each dimension of the construction defined above is described by a *generator matrix.* i.e. for $1 \leq i \leq s$, let $C^{(i)}$ be the $m \times m$ matrix over $\mathbb{F}_q$ with rows

$$c_j^{(i)} = (c_{j,0}^{(i)},\ c_{j,1}^{(i)},\ \ldots,\ c_{j,m-1}^{(i)}) \quad \text{for } j = 1, 2, \ldots, m$$

It is helpful to consider these matrices as a two parameter system

$$C = \{c_j^{(i)} \in \mathbb{F}_q^m : 1 \leq i \leq s,\ 1 \leq j \leq m\}$$

of vectors in $\mathbb{F}_q^m$.

Definition 2.5. A digital net P provided by matrices $C^{(1)}, C^{(2)}, \ldots, C^{(s)}$ is called a **digital $(\mathbf{t}, \mathbf{m}, \mathbf{s})$-net over** $\mathbb{F}_q$ if P is a (t, m, s)-net in base q.

The definition for *digital sequences* is very similar to that of a digital net and as such, it will not be included here. The reader may refer to [5] for additional details. The main difference is that a sequence can have an infinite number of points. Therefore, an infinite number of bijections ψ_r and $\eta_j^{(i)}$ will be needed and each generator matrix $C^{(i)}$ will be of order $\infty \times \infty$.

Definition 2.6. A digital sequence provided by matrices $C^{(1)}, C^{(2)}, \ldots, C^{(s)}$ is called a **digital $(\mathbf{t}, \mathbf{s})$-sequence over** $\mathbb{F}_q$ if for all integers $m > t$ the s left upper matrices $C_m^{(i)}$ of $C^{(i)}$, $1 \leq i \leq s$, provide a digital (t, m, s)-net in base q.

The definitions given in relation to digital constructions is somewhat complex. We will now describe a function, which will be followed by an example to illustrate the fundamental idea of this type of construction in a more concrete manner.

Consider the following method for generating a point, u_i:

Point	**Expansion**	$\boldsymbol{\phi_3(i-1)}$	**Point**	**Expansion**	$\boldsymbol{\phi_3(i-1)}$
u_1	$0 = 0$	0	u_6	$5 = 2 \times 3^0 + 1 \times 3^1$	$7/9$
u_2	$1 = 1 \times 3^0$	$1/3$	u_7	$6 = 2 \times 3^1$	$2/9$
u_3	$2 = 2 \times 3^0$	$2/3$	u_8	$7 = 1 \times 3^0 + 2 \times 3^1$	$5/9$
u_4	$3 = 1 \times 3^1$	$1/9$	u_9	$8 = 2 \times 3^0 + 2 \times 3^1$	$8/9$
u_5	$4 = 1 \times 3^0 + 1 \times 3^1$	$4/9$	u_{10}	$9 = 1 \times 3^2$	$1/27$

Table 2.1: The first 10 points of the van de Corput sequence in base 3

1. Expand $i-1$ over base b where $b \geq 2$. That is, we first compute the $a(i)$'s of the expansion $i-1 = \sum_{l=0}^{\infty} a_l(i)b^l$ where it is assumed that such an expansion will have infinitely many zeros

2. Define a function $\phi_b : \mathbb{N} \to [0,1)$ known as the **radical inverse function in base** b as follows:

$$\phi_b(i-1) = \sum_{l=0}^{\infty} a_l(i)b^{-l-1} \tag{2.1}$$

   Compute $u_i = \phi_b(i-1)$

It can be observed that the right hand side of Equation (2.1) contains successive inverse powers of b where $a_l(i) < b, \forall\, l \geq 0$. Hence, $u_i = \phi_b(i-1) \in [0,1), \forall i \geq 1$.

The resulting sequence of the radical inverse function in base b is called the **van de Corput sequence in base** b.

Example 2.2. By reviewing the sequence in Table 2.1 we see that the first 3 points split the interval $[0,1)$ into 3 equal parts. The next point u_4 fills the space between u_1 and u_2. u_5 fills the space between u_2 and u_3 with other such spaces being filled as the sequence continues by repeatedly cycling through the interval $[0,1)$.

Example 2.2 is a special case of Definition 2.4 in a single dimension where $\mathbb{F}_q = \mathbb{Z}_3$ and the bijections ψ_r and $\eta_j^{(i)}$ are simply the identity mapping for $0 \leq r \leq 2$, and $1 \leq j \leq 3$. Also, the generator matrix $C^{(1)}$, is the 3×3 identity matrix.

Such a sequence can be extended to higher dimensions by choosing appropriate linear transformations for the $a(i)$'s before applying the radical inverse function. This essentially refers to choosing generator matrices $C^{(1)}, C^{(2)}, \ldots, C^{(s)}$ that result in good quality point sets. It turns out that this is exactly the concept presented by *Sobol'* [6] in 1967 to define his well known LP_τ-sequence which uses base 2. Later on, we will see how the polynomial Hickernell sequence compares against this and other known methods.

2.3 Formal Laurent Series

The formal Laurent series is used extensively for the implementation of polynomial Hickernell sequences and we give the definition here as it is presented in [3]:

> The field of **formal Laurent series** over $\mathbb{F}_b[z]$ is defined as the set $\mathbb{F}_b((z^{-1}))$ of elements L of the form
> $$L = \sum_{r=w}^{\infty} a_r z^{-r}$$
> where the coefficients a_r are in $\mathbb{F}_b$.

Rational functions of the form $g(z)/p(z)$ can be represented by a formal Laurent series. For now, we will assume that $p(z)$ is monic. We will see later why this assumption can be made.

We are interested in computing the a_i coefficients for the following expansion:

$$\frac{g(z)}{p(z)} = \sum_{r=w}^{\infty} a_r z^{-r}$$

Let $g(z) = b_m z^m + b_{m-1} z^{m-1} + \ldots + b_1 z + b_0$ and $p(z) = z^n + c_{n-1} z^{n-1} + \ldots + c_1 z + c_0$.

$$\begin{aligned}
\implies \quad b_m z^m + b_{m-1} z^{m-1} + \ldots + b_1 z + b_0 \quad &= a_w z^{n-w} + a_{w+1} z^{n-w+1} + a_{w+2} z^{n-w+2} + \ldots + \\
& a_w c_{n-1} z^{n-w-1} + a_{w+1} c_{n-1} z^{n-w-2} + \ldots + \\
& \ldots + \\
& a_w c_1 z^{1-w} + a_{w+1} c_1 z^{-w} + a_{w+2} c_1 z^{-w-1} + \ldots + \\
& a_w c_0 z^{-w} + a_{w+1} c_0 z^{-w-1} + a_{w+2} c_0 z^{-w-2} + \ldots \\
\implies \quad b_m z^m + b_{m-1} z^{m-1} + \ldots + b_1 z + b_0 \quad &= a_w z^{n-w} \\
& + (a_{w+1} + a_w c_{n-1}) z^{n-w-1} \\
& + (a_{w+2} + a_{w+1} c_{n-1} + a_w c_{n-2}) z^{n-w-2} \\
& + \ldots \\
& + (a_{w+n-1} + a_{w+n-2} c_{n-1} + \ldots + a_{w+1} c_2 + a_w c_1) z^{1-w} \\
& + (a_{w+n} + a_{w+n-1} c_{n-1} + \ldots + a_{w+1} c_1 + a_w c_0) z^{-w} \\
& + (a_{w+n+1} + a_{w+n} c_{n-1} + \ldots + a_{w+2} c_1 + a_{w+1} c_0) z^{-w-1} \\
& + \ldots
\end{aligned}$$

We let $n - w$ be equal to m and by equating coefficients of powers of z, we have that

$$
\begin{aligned}
&a_w = b_m \\
&a_{w+i} = b_{m-i} - \sum_{j=0}^{i-1} a_{w+j} c_{n-i+j} \quad \text{for } i = 1, \ldots, n-1 \\
&a_{w+i} = b_{m-i} - \sum_{j=0}^{n-1} a_{w+i-n+j} c_j \quad \text{for } i = n, \ldots, m \\
&a_{w+i} = - \sum_{j=0}^{n-1} a_{w+i-n+j} c_j \quad \text{for } i = m+1, \ldots
\end{aligned}
$$

These coefficients will be important since the algorithm for the polynomial Hickernell sequence makes use of the fact that rational functions can be expressed as a formal Laurent series. The reader may also notice that the structure of the Laurent series is similar to the expansion of $x_n^{(i)}$ in Definition 2.4.

2.4 Fundamental Algorithms

There are a few key basic algorithms that are incorporated into searching parameters for the polynomial Hickernell sequence. Three of the important ones are combinations, testing reducibility and a variant of Gaussian elimination.

2.4.1 Combinations

The combinations described are of the form $\binom{n}{r}$. These enumerations are needed to evaluate low-dimensional projections and to appropriately partition sub-systems of the generator matrices for quality testing. Both of which, will be explained later. The method is given in Algorithm 2.1.

2.4.2 Testing Reducibility

One of the parameters to be described is the g parameter. This corresponds to a polynomial that will be used along with a deliberately chosen irreducible polynomial to generate s other polynomials. There is an advantage to be gained by using an irreducible polynomial and

Algorithm 2.1 COMBINATIONS(n, r)

```
1: $A[1..r] \leftarrow$ 0's
2: $k \leftarrow 1$
3: while $k \geq 1$ do
4:     while $A[k] < n$ do
5:         $A[k] \leftarrow A[k] + 1$
6:         if k = r then
7:             A contains combination!
8:         else
9:             $k \leftarrow k + 1$
10:            $A[k] \leftarrow A[k-1]$
11:        end if
12:    end while
13:    $k \leftarrow k - 1$
14: end while
```

because of this, we refer the reader to [8] by Gathen and Gerhard for algorithms on testing reducibility.

2.4.3 Gaussian Elimination Variant

Gaussian elimination is a well known algorithm and it is used heavily here. However, for our purposes we have made a slight modification to improve efficiency. Usually, the pivot element is selected in a specific order but we have chosen to ignore this. Our main goal is simply to determine whether or not an $m \times n$ system is linearly dependent, where $m \leq n$. Consequently, we traverse from row 1 to row m using the first non-zero element in each row as the pivot. If no such position exists then the system is dependent. The steps are presented in Algorithm 2.2.

2.5 Polynomial Analogue of the Radical Inverse Function

Recall that on page 8 we introduced the radical inverse function in base b. The polynomial analogue is very similar to this. Let $\mathbb{F}_b[z] \times (\mathbb{F}_b[z] - \{0\})$ refer to the set of rational functions in z over $F_b[z]$.

Algorithm 2.2 IsDependent(A)

```
 1: $m \leftarrow$ RowDimension($A$)
 2: $n \leftarrow$ ColumnDimension($A$)
 3: for $i = 1$ to $m$ do
 4:   $pos \leftarrow$ FirstNonZero($A[i][1..n]$)
 5:   if $\nexists\, pos$ then
 6:     return true
 7:   end if
 8:   for $j = i + 1$ to $m$ do
 9:     $mp = A[j][pos] \times (A[i][pos])^{-1}$
10:     $A[j][1..n] \leftarrow A[j][1..n] - (mp \times A[i][1..n])$
11:   end for
12: end for
13: return false
```

Example 2.3. The rational functions $\frac{z^2+1}{1}$, $\frac{z^2+1}{z}$, $\frac{z}{z^3+z+1}$ all belong to $\mathbb{F}_b[z] \times (\mathbb{F}_b[z] - \{0\})$

Consider the following:

1. Expand $g(z)$ over base $p(z)$ where degree$(p(z)) > 0$. That is, compute r's of the expansion $g(z) = \sum_{l=0}^{\infty} r_l(g(z))p(z)^l$ where it is assumed that such an expansion will have infinitely many zeros.

2. Define a function $\phi_{p(z)} : \mathbb{F}_b[z] \to \mathbb{F}_b[z] \times (\mathbb{F}_b[z] - \{0\})$ known as the **polynomial analogue of the radical inverse function in base** $p(z)$ as follows:

$$\phi_{p(z)}(g(z)) = \sum_{l=0}^{\infty} r_l(g(z))p(z)^{-l-1}$$

This polynomial version of the radical inverse function is introduced since it is used in the implementation of the polynomial Hickernell sequence.

2.6 Hickernell Sequences

Once again, we will use the radical inverse function presented on page 8. This time we will use it to define the Hickernell sequence as it is found in [7]:

Point	$\boldsymbol{\phi_3(i-1)}$	$\mathbf{2\times\phi_3(i-1)}$	$\mathbf{3\times\phi_3(i-1)}$	$\mathbf{5\times\phi_3(i-1)}$
u_1	0	0	0	0
u_2	$1/3$	$2/3$	1	$2\,2/3$
u_3	$2/3$	$1\,1/3$	2	$3\,1/3$
u_4	$1/9$	$2/9$	$1/3$	$5/9$
u_5	$4/9$	$8/9$	$1\,1/3$	$2\,2/9$

Table 2.2: Computing the first 5 points of Hickernell sequence in base 3

Definition 2.7. The **Hickernell sequence** is defined as

$$\big(\{g_1\phi_b(n)\},\{g_2\phi_b(n)\},\ldots,\{g_s\phi_b(n)\}\big)\quad\text{for } n=0,1,2,\ldots$$

where the integers $g_1, g_2, \ldots, g_s$ are suitably chosen and $\{\alpha\}$ denotes the fractional part of the real α.

Example 2.4. Using the first 5 points from Example 2.2 and the integers $g_1 = 2, g_2 = 3, g_3 = 5$ we can obtain the first 5 points of the associated Hickernell sequence by doing the computations shown in Table 2.2. The resulting point set is

$$\big\{(0,0,0),(2/3,0,2/3),(1/3,0,1/3),(2/9,1/3,5/9),(8/9,1/3,2/9)\big\}$$

Finding suitable g's that result in a sequence that satisfies the definition of low-discrepancy sequences is no trivial undertaking. However, some authors have shown that for some values of b, there exists g's that result in Hickernell sequences that fulfill these requirements (see [7] and references therein).

The analysis of polynomial arithmetic over finite fields is simpler than its integer arithmetic counterpart and this has fueled the drive to explore the polynomial Hickernell sequence. As a result, there will be less complicated avenues for evaluating the quality of point sets and good initialisation parameters can be found in a more speedy fashion.

Chapter 3

The Polynomial Arithmetic Analogue of Hickernell Sequences

3.1 Polynomial Version of Hickernell Sequences

In the same way that the radical inverse function was used for the definition of Hickernell sequences, we will use the polynomial analogue of the radical inverse function to define the *polynomial version of Hickernell sequences.* In the Hickernell sequence, when we generated a point u_i we found a's such that $i-1 = \sum_{l=0}^{\infty} a_l(i)b^l$. However, for the polynomial version, we proceed to create a polynomial $v_{i-1}(z)$ using the coefficients $a_0, a_1, a_2, \ldots$ such that

$$v_{i-1}(z) = \sum_{l=0}^{\infty} a_l(i)z^l$$

The following comes from [7]:

Definition 3.1. The **polynomial version of Hickernell sequences** is defined as

$$\big(\{g_1(z)\phi_{p(z)}(v_i(z))\}, \{g_2(z)\phi_{p(z)}(v_i(z))\}, \ldots, \{g_s(z)\phi_{p(z)}(v_i(z))\}\big), \quad \text{for } i = 0, 1, 2, \ldots$$

where $g_1(z), g_2(z), \ldots, g_s(z)$ are suitably chosen polynomials over $\mathbb{F}_b$ and $\{\alpha(z)\}$ denotes the fractional part of the formal Laurent series $\alpha(z)$.

The definition given above is not sufficient to obtain the required points. It gives rise to points that are s-tuples of rational functions with indeterminate z. We need points

Point	base b	Polynomial	base $p(z)$	rational function s-tuples
u_1	0	0	0	$(0,0,0)$
u_2	1×2^0	1	$1 \times (z+1)^0$	$\left(\frac{1}{z+1}, \frac{z}{z+1}, \frac{z^2}{z+1}\right)$
u_3	1×2^1	z	$1 \times (z+1)^0 + 1 \times (z+1)$	$\left(\frac{z}{(z+1)^2}, \frac{z^2}{(z+1)^2}, \frac{z^3}{(z+1)^2}\right)$
u_4	$1 \times 2^0 + 1 \times 2^1$	$1+z$	$1 \times (z+1)$	$\left(\frac{1}{(z+1)^2}, \frac{z}{(z+1)^2}, \frac{z^2}{(z+1)^2}\right)$
u_5	1×2^2	z^2	$1 \times (z+1)^0 + 1 \times (z+1)^2$	$\left(\frac{z^2}{(z+1)^3}, \frac{z^3}{(z+1)^3}, \frac{z^4}{(z+1)^3}\right)$

Table 3.1: Computing the first 5 points of the polynomial Hickernell sequence

$\in [0,1)^s$ so we will introduce a function $\gamma : \mathbb{F}_b((z^{-1})) \to \mathbb{R}$ as follows:

$$\gamma\left(\sum_{r=w}^{\infty} a_r z^{-r}\right) = \sum_{r=w}^{\infty} a_r b^{-r}$$

By applying this function element-wise we will obtain the vector of real numbers. Of course, we will have to convert the rational function into its formal Laurent series equivalent before we can apply the mapping.

Example 3.1. Let us consider the first 5 points of the polynomial Hickernell sequence using the parameters $b = 2, g_1 = 1, g_2 = z, g_3 = z^2$ and $p(z) = z+1$. The breakdown is given in Table 3.1. By applying the γ function to the fractional part of the formal Laurent series equivalent of the 3-tuple elements, we will obtain the desired points from $[0,1)^3$.

3.2 Algorithm

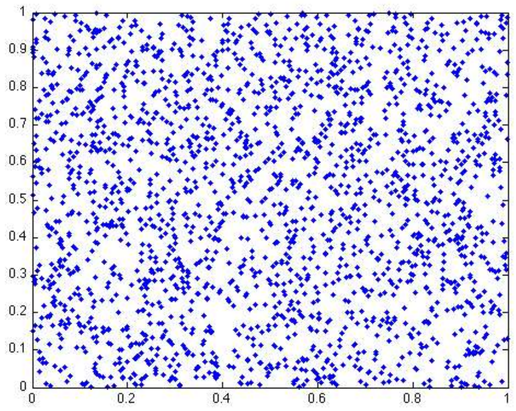
The algorithm originally comes from [7] but the method given in Algorithm 3.1 more closely resembles that which was given in [3] by Lemieux, since the latter was written more explicitly.

The first three lines of Algorithm 3.1 are straightforward. Lines 4-7 incorporate the use of a couple functions defined earlier and seems to be more involved but the implementation is not overly tricky.

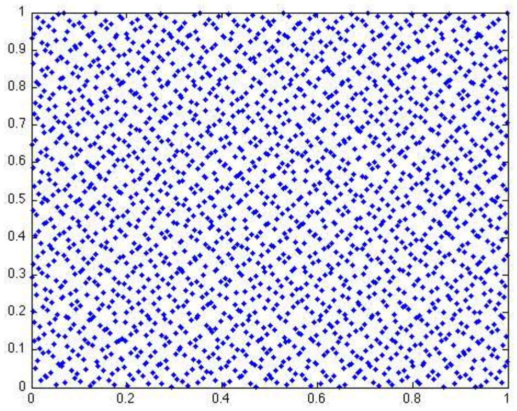
The method given only describes how to compute a single point. Usually, N points are required, so this will be repeated for $i = 1, 2, \ldots, N$.

Algorithm 3.1 PolyHickernellPoint($b, p(z), g(z)$'s$, i$)

```
Input:   $b$ - a base which is usually a prime power
         $p(z)$ - a monic polynomial of degree $n$
         $(g_1(z), \ldots, g_s(z))$ - an $s$-tuple of polynomials
         $i$ - a natural number

Output:  $u_i$ - the point $\in [0,1)^s$ corresponding to $i$

1: write $i - 1 = a_0(i) + a_1(i)b + \ldots + a_m(i)b^m$
2: construct $v_i(z) = a_0(i) + a_1(i)z + \ldots + a_m(i)z^m$
3: write $v_i(z) = r_0(v_i(z)) + r_1(v_i(z))p(z) + \ldots + r_h(v_i(z))p(z)^h$ {where $h = \lfloor m \div n \rfloor$}
4: use $r_0(v_i(z)), r_1(v_i(z)), \ldots, r_h(v_i(z))$ to compute $\phi_{p(z)}(v_i(z))$ as a rational function
5: for $j = 1$ to $s$ do
6:     $u_i[j] \leftarrow \gamma\left(\left\{\frac{g_j(s) \times \text{NUMERATOR}(\phi_{p(z)}(v_i(z)))}{\text{DENOMINATOR}(\phi_{p(z)}(v_i(z)))}\right\}\right)$
7: end for
8: return $u_i$
```



(a) 2048 (2^{11}) random points $\in [0,1)^2$



(b) Polynomial Hickernell digital $(0, 11, 2)$-net

Figure 3.1: 2-dimensional plots of 2048 points $\in [0,1)^2$

Example 3.2. In Figure 3.1, the parameters used for the polynomial Hickernell sequence are as follows:

$b = 2$
$p(z) = z - 1$
$g_1(z) = 1$
$g_2(z) = z^{17} + z^{15} + z^{14} + z^{13} + z^{10} + z^9 + z^6 + 1$
$i = 1, 2, \ldots, 2048$

By observing the plots it can be seen that the distribution of the points for the polynomial Hickernell sequence is more uniform and well-behaved. The random sampling plot has wider gaps are more erratic clustering. This example gives an informal graphical representation of the contrast between the Monte Carlo method and the quasi-Monte Carlo method.

Chapter 4

Searching Parameters

4.1 Parameter Overview

The algorithm for implementing the polynomial Hickernell sequence given in Chapter 3 is presented in a very general way. However, the method we elected to use here is in accordance with *Proposition 2* from [7] where b is prime and $p(z) = z - j$ for $0 \leq j < b$.

We have chosen to use either 2 or 3 for b. We use the two smallest primes since these necessitate generator matrices with more columns[1] and this results in better distinction for parameters of high quality.

The upper-triangular **Pascal matrix**, P, is an infinite matrix having the binomial coefficients as its entries in the upper triangle.

Example 4.1. The truncated 4×4 upper-triangular Pascal matrix is written as:

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

P can be used to change the base of a polynomial from z to $p(z)$. The transformation matrix corresponds to P^j where $p(z) = z - j$. Since the primes used are very small, we have chosen to fix $p(z) = z - 1$ and use the transformation matrix P. Thus, given

[1]The expansion of $i - 1$ has more terms for a small base b

$f = f_0 + f_1 z + \ldots + f_m z^m$ we can write $f = \hat{f}_0 + \hat{f}_1 p(z) + \ldots + \hat{f}_m p(z)^m$ by simply applying the transformation P

$$\hat{\mathbf{f}} = P\mathbf{f}$$

where $\hat{\mathbf{f}} = (\hat{f}_0, \hat{f}_1, \ldots, \hat{f}_m)^T$, $\mathbf{f} = (f_0, f_1, \ldots, f_m)^T$ and P is the truncated $(m+1) \times (m+1)$ upper-triangular Pascal matrix.

Example 4.2. Consider expressing the polynomial $f(z) = 1+z+z^3$ in terms of $p(z) = z-1$ over base 2. Since $f(z)$ is of degree 3, we will need a 4×4 Pascal matrix and will apply this transformation to $\mathbf{f}$ for the result.

$$P \times \mathbf{f} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} \implies f = 1 + (z-1)^2 + (z-1)^3$$

The reader may verify that f written in base $p(z)$ is equivalent to the original in base z.

Soon, we will see how linear algebra can be used to evaluate quality and where the selection of the g's affects this process. In [4], we see that the *linear recurring sequence* $\mathbf{g} = (1, g \bmod r, g^2 \bmod r, \ldots, g^{s-1} \bmod r)$ is used to select the g polynomials, for some irreducible polynomial r. The success of their method has inspired our technique. The equivalent of their r in our construction would be p. However, since p is restricted to having degree 1, this is a very poor choice for us. Instead, we use a polynomial r with degree corresponding to the size of the biggest point set for which the quality will be computed. Call this large value HP. We choose r to be the first irreducible polynomial of degree m_2, where $m_2 = \lceil \log_b(HP) \rceil$. The ordering is performed lexicographically, starting with the term of degree m_2.

Example 4.3. Samples showing polynomial ordering in base 2:

$1 + z^2 + z^5$	comes before	$1 + z^3 + z^5$
$1 + z^3 + z^5$	comes before	$1 + z + z^2 + z^3 + z^5$
$1 + z + z^3 + z^4 + z^5$	comes before	$1 + z^2 + z^3 + z^4 + z^5$

If $b = 2$ and $m_2 = 5$ then we would choose $1 + z^2 + z^5$ as our r.

Given a polynomial g, we choose $\mathbf{g} = (g_1, g_2, \ldots, g_s)$ to be the first s terms of the sequence $(g^i \bmod r)$, $i \geq 0$, where $(g^i \bmod r)$ has a constant term when expressed in terms of $p(z)$. If such a collection cannot be found quickly then an alternative g is selected. The irreducible r is an essential tool used to assist us with selecting g's and it has proven to be very effective.

Sometimes a polynomial parameter will be referred to as an integer. In this context, we are referencing the corresponding polynomial when the integer is written in base b.

Example 4.4. Using base 2, for $g = 37$, we have that $g = 1 + 2^2 + 2^5 \implies g = 1 + z^2 + z^5$.

In the search for good parameters, many values of g will be examined. Once the base has been selected, this will be the only varying parameter during the search.

4.2 Generator Matrices

In Section 2.2, we introduced the concept of generator matrices and mentioned that each dimension of a digital net construction could be characterised be a generator matrix.

Since the polynomial Hickernell sequence is a class of digital (t, s)-sequences, this principle also applies here. The generator matrices $C^{(1)}, C^{(2)}, \dots, C^{(s)}$ account for the linear transformation of the vector equivalent of the ith point in base b, to the coefficients used for the fractional part of the formal Laurent series in each respective dimension. The description of these matrices comes from [7]:

Let $g_k(z) = \hat{g}_0^{(k)} + \hat{g}_1^{(k)} p(z) + \hat{g}_2^{(k)} p(z)^2 + \dots$ for $k = 1, 2, \dots, s$. Then

$$\hat{G}^{(k)} = \begin{pmatrix} \hat{g}_0^{(k)} & \hat{g}_1^{(k)} & \hat{g}_2^{(k)} & \cdots \\ 0 & \hat{g}_0^{(k)} & \hat{g}_1^{(k)} & \cdots \\ 0 & 0 & \hat{g}_0^{(k)} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

The representation of the generator matrices for Tezuka's Proposition 2 is written as

$$C^{(k)} = P^T \hat{G}^{(k)} P \quad \text{for } k = 1, 2, \dots, s$$

where P is the upper-triangular Pascal matrix.

In Section 4.1, we explained that for a g^i in the recurring sequence to be chosen, then the g^i expressed in base $p(z)$, had to have a non-zero constant term. The reason for this is to ensure that the matrices $C^{(k)}$ for $k = 1, 2, \dots, s$ are all non-singular[2]. This results in each dimension being better distributed. That is, having a corresponding t-value of 0.

[2]The product of 3 square triangular matrices with non-zero elements on each of the diagonals is invertible

For the purposes of evaluating quality, we are forced to restrict the size of the generator matrices. Since the maximum number of points considered is HP, it is appropriate to use square matrices of size m_2, where m_2 is as previously defined. Hence, $C^{(k)}$ is an $m_2 \times m_2$ matrix for $k = 1, 2, \ldots, s$.

4.3 Projections

Our introduction of *projections* is based on [5]. We consider an s-dimensional digital net. Let V be a subset of $\{1, 2, \ldots, s\}$ having cardinality u. Then a **u-dimensional projection** refers to the u-dimensional digital net containing the corresponding values for the columns specified by V. Consequently, if $C^{(1)}, C^{(2)}, \ldots, C^{(s)}$ are the matrices that provide the s-dimensional digital net then the respective V matrices will provide the u-dimensional digital net.

Example 4.5. If we use the points from Example 2.4, then the 2-dimensional projection corresponding to $V = \{2, 3\}$ would be the following point set:

$$\big\{(0,0), (0, {}^{2}\!/\!_{3}), (0, {}^{1}\!/\!_{3}), ({}^{1}\!/\!_{3}, {}^{5}\!/\!_{9}), ({}^{1}\!/\!_{3}, {}^{2}\!/\!_{9})\big\}$$

Similarly, the 1-dimensional projection corresponding to $V = \{1\}$ is:

$$\big\{0, {}^{2}\!/\!_{3}, {}^{1}\!/\!_{3}, {}^{2}\!/\!_{9}, {}^{8}\!/\!_{9}\big\}$$

Projections are important because there are some practitioners using digital constructions who require that the lower dimensional projections be as well distributed as possible (see [5] and references therein). They are also important because it has been shown that some multi-dimensional numerical integrals can be closely approximated by point sets having good quality low-dimensional projections. In this regard, we have chosen to compute quality only for 2, 3 and 4-dimensional projections. There is no need to consider 1-dimensional projections since the generator matrices are non-singular. By only considering these low-dimensional projections we significantly reduce the computational burden of evaluating quality. This will be better understood when the quality algorithm is given in the next section.

4.4 Quality Evaluation

4.4.1 Method

The t parameter of digital (t, m, s)-nets and (t, s)-sequences is used to indicate the quality of the respective constructions, where lower values correspond to better quality. In Section 2.2, we saw that the definition of digital (t, s)-sequences is dependent on that of (t, m, s)-nets. This allows us to focus on (t, m, s)-nets for the purposes of examining quality. We give a definition and a lemma from [4], that highlight how linear algebra can be used for quality testing:

Definition 4.1. Let d be an integer with $0 \leq d \leq m$. The system $\{\mathbf{c}_j^{(i)} \in \mathbb{F}_q^m : 1 \leq j \leq m, 1 \leq i \leq s\}$ of vectors is called a $(\mathbf{d}, \mathbf{m}, \mathbf{s})$**-system over** $\mathbb{F}_q$ if for non-negative integers $d_1, d_2, \ldots, d_s$ with $\sum_{i=1}^{s} d_i = d$ the vectors $\mathbf{c}_j^{(i)}, 1 \leq j \leq d_i, 1 \leq i \leq s$, are linearly independent over $\mathbb{F}_q$.

Lemma 4.1. *The $m \times m$ matrices $C^{(1)}, C^{(2)}, \ldots, C^{(s)}$ provide a digital (m-d,m,s)-net constructed over $\mathbb{F}_q$ if and only if the system $\{\mathbf{c}_j^{(i)} : 1 \leq j \leq m, 1 \leq i \leq s\}$ of their row vectors is a (d, m, s)-system over $\mathbb{F}_q$.*

Lemma 4.1 provides us with an algorithm for computing t. That is, use the generator matrices to determine the largest value d, where $1 \leq d \leq m$, such that the digital net forms a (d, m, s)-system. Once this is found, we have that $t = m - d$.

4.4.2 Quality Vectors

The method outlined for computing quality is to be applied to digital nets. However, having a good quality digital net is not sufficient for our goals, since a sequential subset starting at the first point, may have very low quality. Having a sequence of points that maintain high quality from start to finish is what we seek.

We will use HP from Section 4.1 as the finish point and introduce a new term, LP, as the start point, where LP is a small integer indicating a minimal number of points. Let $m_1 = \lceil \log_b LP \rceil$. We compute t for each of $m = m_1, m_1 + 1, \ldots, m_2$ and call the resulting vector of t values a **quality vector**, written $\mathbf{t}_g^u$, where g corresponds to the g's used and u indicates the magnitude of the low-dimensional projections considered. The average of these t values, denoted $\bar{t}$, is used to assess the uniformity of the digital construction and naturally, lower averages are preferred.

Example 4.6. Suppose $b = 2, LP = 100, HP = 1000$. Then $m_1 = 7$ and $m_2 = 10$. Therefore, t will be computed for each of $m = 7, 8, 9, 10$ and the average of these 4 will be used for quality assessment.

4.4.3 Algorithm

The core of the method given in Algorithm 4.1 is based on Lemma 4.1 and comes from [4] but we have made two additions. Computing quality as a vector and the inclusion of low-dimensional projections. The former is done by adding an outer loop from m_1 to m_2 but the latter is not as straightforward. To account for projections, we have to consider partitions of d into s parts where exactly u parts are non-zero. We accomplish this by first considering combinations of s into u parts and then partitions of d into the same u parts. These separate selections are incorporated to produce the desired partition of d into s parts.

4.4.4 Complexity

Standard school algorithms have been used to conduct polynomial arithmetic for this implementation and as such, the associated cost of computing will reflect this.

The cost of modular arithmetic over $\mathbb{Z}_b$ has been left out for simplicity purposes. We will be considering the worse-case scenario for Algorithm 4.1.

Line 1:	$O(sm_2^2)$	Since we do polynomial multiplication and division with remainder
Line 2:	$O(sm_2^3)$	Since we perform matrix multiplication
Lines 4-17:	$O\Big((m_2 - m_1)\, m_2^4 \binom{s}{u} \binom{d+u-1}{u-1}\Big)$	Since there are four loops and at the innermost level there is a linear dependence evaluation

By taking all the steps into consideration, we have that the total cost of Algorithm 4.1 is

$$O\left(sm_2^2 + sm_2^3 + (m_2 - m_1)\, m_2^4 \binom{s}{u} \binom{d+u-1}{u-1}\right)$$

This upper bound is somewhat harsh since there will be many instances when the row vector subset is dependent. For these cases, the next iteration will start at the first loop.

Algorithm 4.1 FindQuality(s, b, u, g, m_1, m_2, r)

```
Input:   $s$ - the number of dimensions for the point set
         $b$ - the prime base (either 2 or 3)
         $u$ - the magnitude of the low-dimensional projections (either 2, 3 or 4)
         $g$ - the integer representation of the $g$ polynomial used
         $m_1$ - $\lceil \log_b LP \rceil$
         $m_2$ - $\lceil \log_b HP \rceil$
         $r$ - the first irreducible polynomial of degree $m_2$

Output:  $\mathbf{t}_g^u$ - the quality vector corresponding to the polynomial $g$ and having its
         $u$-dimensional projections evaluated

1: $\mathbf{g} \leftarrow$ sequence of $s$ vectors generated by $g$ and $r$
2: $\mathbf{C} \leftarrow s$ generator matrices of size $m_2 \times m_2$ based on $\mathbf{g}$
3: $\mathbf{t}_g^u[1..m_2 - m_1 + 1] \leftarrow$ 0's
4: for $i = m_1$ to $m_2$ do
5:   for $d = u$ to $i$ do
6:     for all combinations of $s$ into $u$ parts do
7:       for all partitions of $d$ into $u$ parts do
8:         use both selections to construct partition of $d$ into $s$ parts
9:         use the first $d_i$ rows of matrix $C^{(i)}$ for $i = 1, 2, \ldots, s$ to compose a subset of
           $\{v_1, v_2, \ldots, v_d\}$ vectors
10:        if subset is linearly dependent then
11:          $\mathbf{t}_g^u[i - m_1 + 1] \leftarrow (i - d + 1)$
12:          continue from loop over $i$
13:        end if
14:      end for
15:    end for
16:  end for
17: end for
18: return $\mathbf{t}_g^u$
```

4.5 Searching Scheme

Searching for good quality initialisation parameters basically boils down to finding a g that has a low $\bar{t}$ value. Since the elements of $\mathbf{g}$ are each computed modulo r, this constrains the degree of g to being $< m_2$. However, it is not practical to explore all such g's for a reasonable choice of HP in higher dimensions. Initially, we used a uniform random search of g values within the possibility space[3], executed a specific number of times, called $TRIALS$. This gave fair results but we also examined searching g values sequentially, from some starting point $TRIALS$ times and this consistently gave better results. Consequently, it was determined that an appropriate way to search for good g's is to create different regions, denoted $REGIONS$, in the possibility space by randomly selecting a g value and then performing a sequential search $TRIALS$ times for each of these regions. The top results based on $\bar{t}$ are returned and displayed with their associated quality vectors. The user may choose any of these to initialise a sequence of points. Algorithm 4.2 has the method.

Algorithm 4.2 SEARCHGS(s, b, u, m_1, m_2, r)

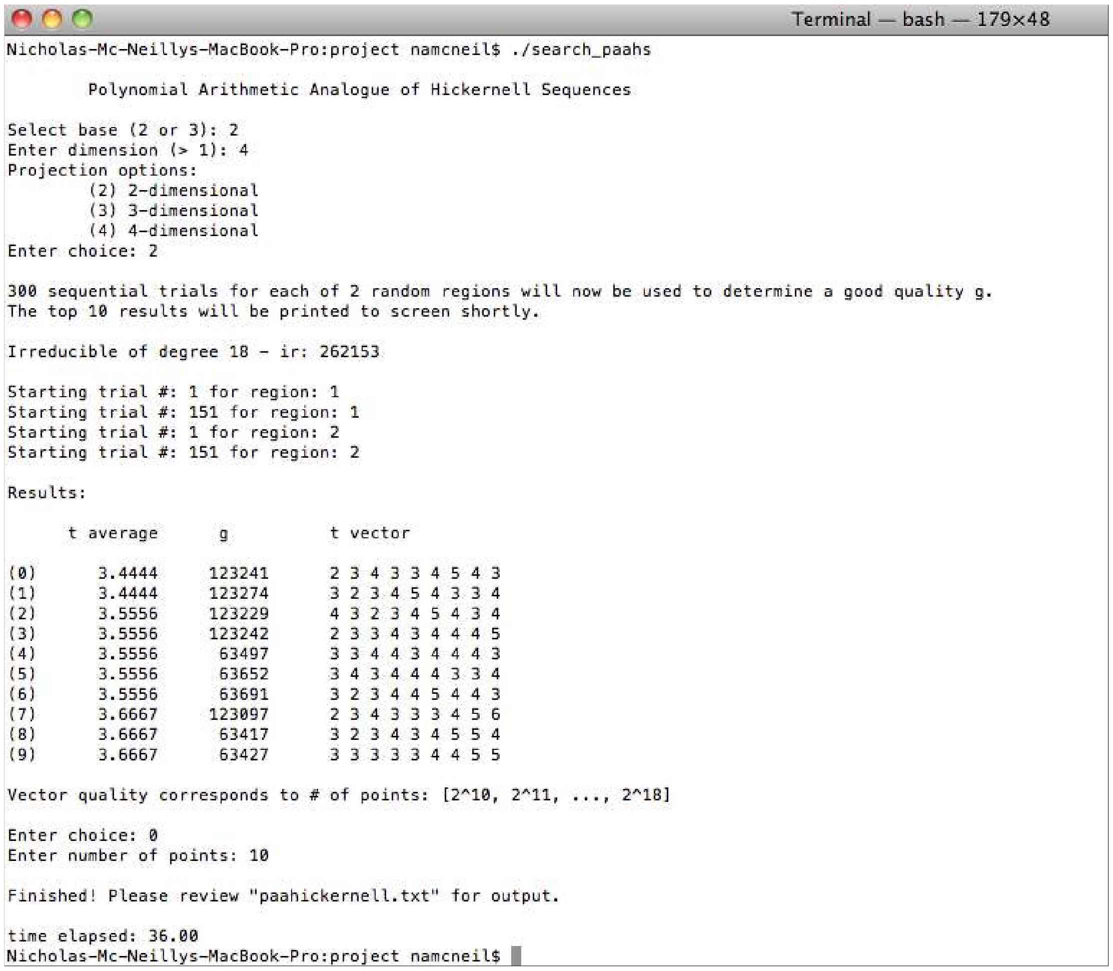
Input:	the input parameters are the same as that of the FINDQUALITY algorithm except that g is excluded
Output:	R - a structure used to store the top $DISPLAY$ results sorted by $\bar{t}$ with the associated g and $\mathbf{t}_g^u$ also being recorded

```
1: $bound \leftarrow b^{m_2}$
2: $R[1..DISPLAY] \leftarrow$ empty
3: **for** $i = 1$ to $REGIONS$ **do**
4:     $pivot \leftarrow$ RANDOM$(1, bound)$
5:     **for** $j = 0$ to $TRIALS - 1$ **do**
6:         $g \leftarrow pivot + j$
7:         $\mathbf{t}_g^u \leftarrow$ FINDQUALITY$(s, b, u, g, m_1, m_2, r)$
8:         $\bar{t} \leftarrow$ AVERAGE$(\mathbf{t}_g^u)$
9:         INSERTRESULTS$(R, g, \mathbf{t}_g^u, \bar{t})$
10:     **end for**
11: **end for**
12: **return** $R$
```

[3] g integers corresponding to polynomials having degree $< m_2$

4.6 Producing Points

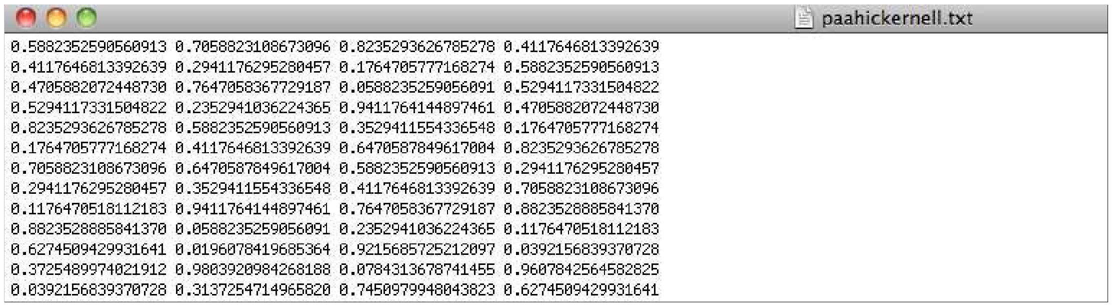
A command line interface is used to enable the user to perform the search and to select a g to generate the required number of points. Currently, the parameters HP,LP,$TRIALS$, $REGIONS$ and $DISPLAY$ are hard-coded into the application. When the program is started, the user is first prompted for three parameters: the base, the dimension and the magnitude of the low-dimensional projections. At this stage, a message is printed indicating that the search has begun and as the search goes on, an update is written to screen at appropriate intervals describing the progression. After the search is completed, the top $DISPLAY$ results will be shown and the user will be prompted to select an option. This will be followed by the number of points needed. The points will then be printed to a named text file with each point being separated by a new line character and within the point, each dimension is separated by white space. Screenshots of the interface and the points file are given in Figures 4.1 and 4.2 respectively.



Terminal — bash — 179×48

```
Nicholas-Mc-Neillys-MacBook-Pro:project namcneil$ ./search_paahs

        Polynomial Arithmetic Analogue of Hickernell Sequences

Select base (2 or 3): 2
Enter dimension (> 1): 4
Projection options:
        (2) 2-dimensional
        (3) 3-dimensional
        (4) 4-dimensional
Enter choice: 2

300 sequential trials for each of 2 random regions will now be used to determine a good quality g.
The top 10 results will be printed to screen shortly.

Irreducible of degree 18 - ir: 262153

Starting trial #: 1 for region: 1
Starting trial #: 151 for region: 1
Starting trial #: 1 for region: 2
Starting trial #: 151 for region: 2

Results:

      t average     g          t vector

(0)      3.4444     123241     2 3 4 3 3 4 5 4 3
(1)      3.4444     123274     3 2 3 4 5 4 3 3 4
(2)      3.5556     123229     4 3 2 3 4 5 4 3 4
(3)      3.5556     123242     2 3 3 4 3 4 4 4 5
(4)      3.5556      63497     3 3 4 4 3 4 4 4 3
(5)      3.5556      63652     3 4 3 4 4 4 3 3 4
(6)      3.5556      63691     3 2 3 4 4 5 4 4 3
(7)      3.6667     123097     2 3 4 3 3 3 4 5 6
(8)      3.6667      63417     3 2 3 4 3 4 5 5 4
(9)      3.6667      63427     3 3 3 3 3 4 4 5 5

Vector quality corresponds to # of points: [2^10, 2^11, ..., 2^18]

Enter choice: 0
Enter number of points: 10

Finished! Please review "paahickernell.txt" for output.

time elapsed: 36.00
Nicholas-Mc-Neillys-MacBook-Pro:project namcneil$
```

Figure 4.1: Screenshot of the command line interface



paahickernell.txt

```
0.5882352590560913 0.7058823108673096 0.8235293626785278 0.4117646813392639
0.4117646813392639 0.2941176295280457 0.1764705777168274 0.5882352590560913
0.4705882072448730 0.7647058367729187 0.0588235259056091 0.5294117331504822
0.5294117331504822 0.2352941036224365 0.9411764144897461 0.4705882072448730
0.8235293626785278 0.5882352590560913 0.3529411554336548 0.1764705777168274
0.1764705777168274 0.4117646813392639 0.6470587849617004 0.8235293626785278
0.7058823108673096 0.6470587849617004 0.5882352590560913 0.2941176295280457
0.2941176295280457 0.3529411554336548 0.4117646813392639 0.7058823108673096
0.1176470518112183 0.9411764144897461 0.7647058367729187 0.8823528885841370
0.8823528885841370 0.0588235259056091 0.2352941036224365 0.1176470518112183
0.6274509429931641 0.0196078419685364 0.9215685725212097 0.0392156839370728
0.3725489974021912 0.9803920984268188 0.0784313678741455 0.9607842564582825
0.0392156839370728 0.3137254714965820 0.7450979948043823 0.6274509429931641
```

Figure 4.2: Screenshot of the points file with $s = 4$

Chapter 5

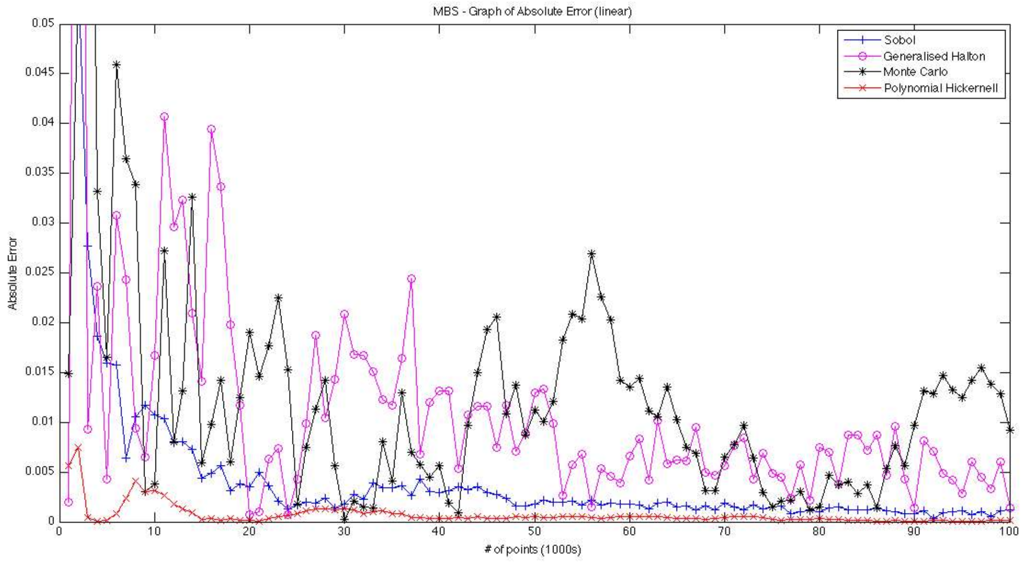
Applications

In this chapter, we give graphical comparisons using plots showing the absolute error of numerical integration for two types of financial problems. The comparisons are illustrated using the Sobol' sequence [6], the generalised Halton sequence [2], Monte Carlo points and the polynomial Hickernell sequence. The first two sequences are well known quasi-Monte Carlo constructions and were chosen as a gauge for evaluating our results. For each problem, the parameters used to initialise the polynomial Hickernell sequence will be given.

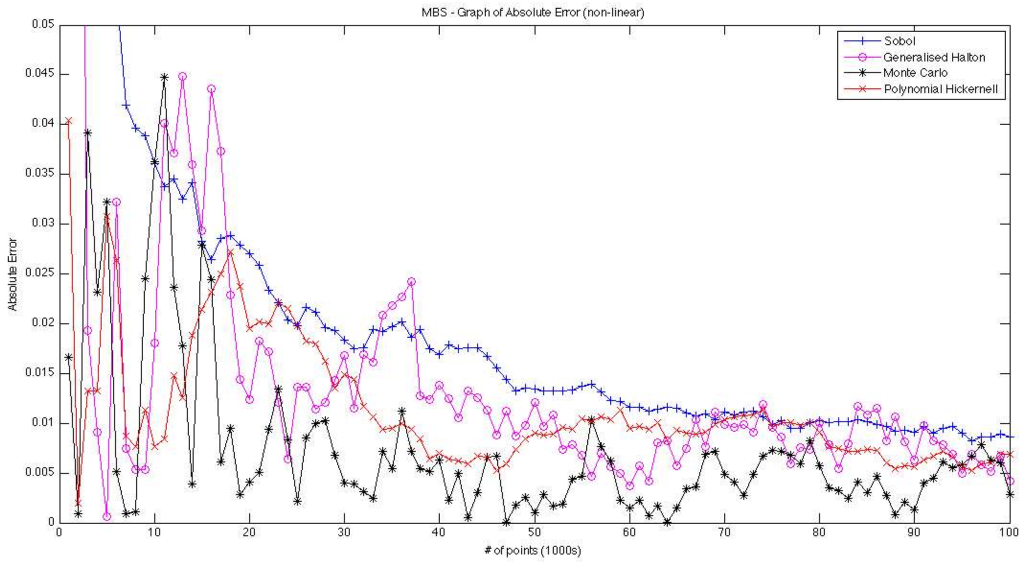
5.1 Mortgage-Backed Security

A *mortgage-backed security* (MBS) is a type of fixed income security where the cash flows are reliant on an underlying pool of mortgages. The pricing of this type of security is investigated in [1] by Caflisch, Morokoff and Owen for special initialisation conditions. They consider mortgages of length 30 years and this gives rise to a problem having 360 dimensions since there are 12 months in a year. We use two of their initialisation sets along with the corresponding solutions. These sets will be referred to as *linear* and *non-linear*. For our purposes, we can think of the linear set as being a basic case and the non-linear set as being a complex case. More information is available in [1]. The following parameters were used to search and initialise the polynomial Hickernell sequence:

HP	LP	u	r	$REGIONS$	$TRIALS$	b	$p(z)$	g	s
200000	1000	2	262153	5	1000	2	$z-1$	157474	360



(a) Absolute error for MBS (linear case)



(b) Absolute error for MBS (non-linear case)

Figure 5.1: Absolute error for MBS

In Figure 5.1a, the linear case, we see that the polynomial Hickernell sequence performs really well. It appears to be far more accurate than the others, with Sobol' being the closest. The Monte Carlo has the worst results.

For the non-linear case shown in Figure 5.1b, we see somewhat of a reverse from what appears in Figure 5.1a. Surprisingly, Monte Carlo seems to give the best results. This is followed by the generalised Halton sequence and the polynomial Hickernell sequence.

In both cases, we see that the polynomial Hickernell sequence is able to compete well with the other constructions and this is very encouraging news.

5.2 Asian Option

In the context of financial instruments, a *call option* is a contract which gives the holder the right, but not the obligation to buy an asset for an agreed price at a certain time T, called the *strike price*, if the asset value rises above the strike. A *put option* is similarly defined but for the case where the holder can sell the asset if the value falls below the strike. *Asian options* are based on this concept but with a more complex definition of the payoff at time T. These options can be priced by simulating a series of price paths and require the following values:

S - the initial asset price
K - the strike price
r - the risk free rate
T - the expiration time (usually in years)
σ - the asset volatility
N - the number of time steps

We consider pricing such a contract using $S = 50$, $r = 0.05$, $T = 1.0$, $\sigma = 0.3$ and $N = 40$ for $K = 45, 50, 55$. In this case, using $N = 40$ corresponds to a problem having 40 dimensions. The following parameters were used for searching and initialising the polynomial Hickernell sequence:

HP	LP	u	r	$REGIONS$	$TRIALS$	b	$p(z)$	g	s
200000	1000	3	262153	5	1000	2	$z-1$	42333	40

In Figure 5.2, where the strike price is below the asset price we see that there is close competition between the polynomial Hickernell sequence and the generalised Halton

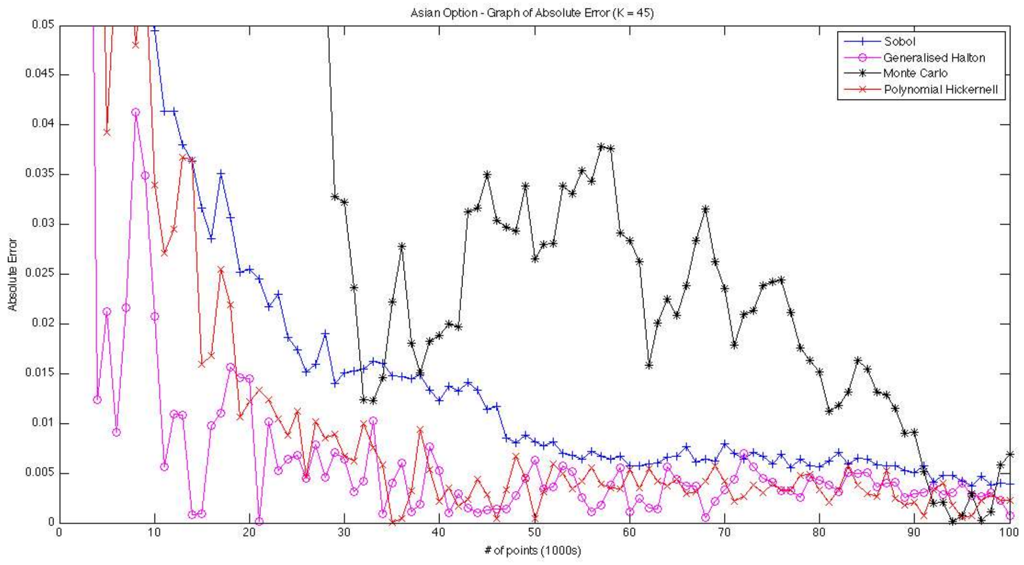


Figure 5.2: Absolute error for Asian Option ($K = 45$)

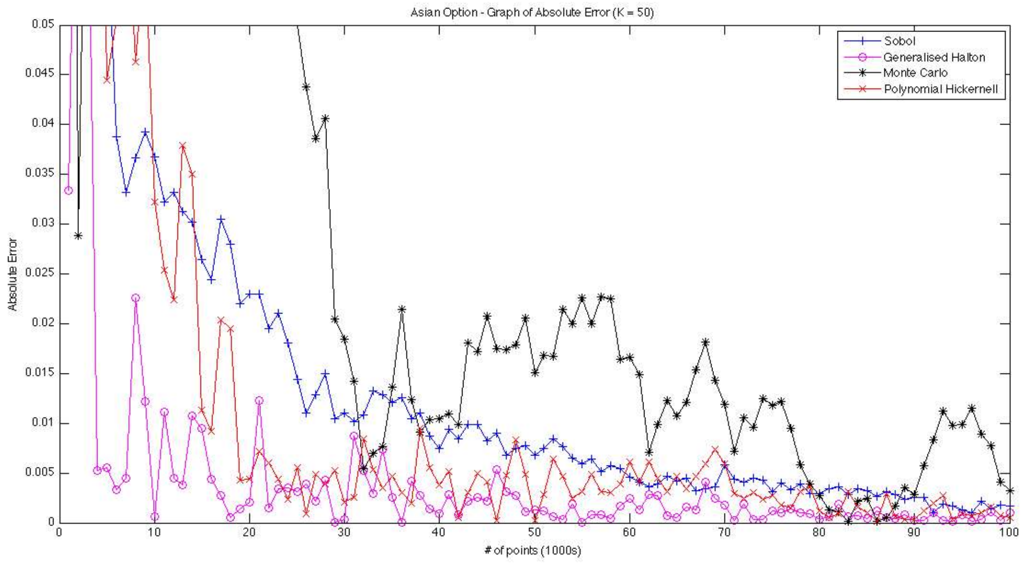


Figure 5.3: Absolute error for Asian Option ($K = 50$)

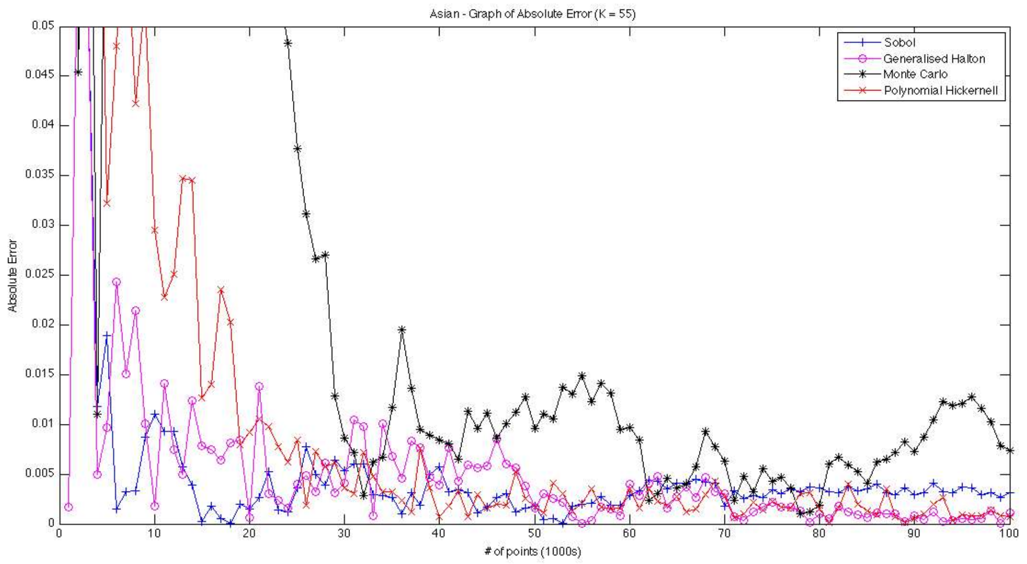


Figure 5.4: Absolute error for Asian Option ($K = 55$)

sequence. Sobol' progresses steadily to the result but it isn't as good as the first two mentioned. Monte Carlo ends on a good note but it is clearly not in the same class as the others.

When the strike price is equal to the initial asset price, shown in Figure 5.3, we see that the generalised Halton sequence edges out the polynomial Hickernell sequence for the top spot with Sobol' following closely for the third position. Once again, Monte Carlo is poor.

In Figure 5.4, the strike price exceeds the initial value and the rivalry between the generalised Halton sequence and the polynomial Hickernell sequence continues. It is not clear as to which one is superior but they are followed by Sobol' and then by Monte Carlo.

The constructions for Sobol' and the generalised Halton sequence are well tested and for the polynomial Hickernell sequence to compete with these so effectively is quite remarkable. In contrast to the MBS problem where 2-dimensional projections were used, we decided to use 3-dimensional projections here. Through experimentation we came to the realisation that considering 3-dimensional projections is more suitable for the Asian option case.

Chapter 6

Conclusion

The Polynomial Hickernell sequence is a quasi-Monte Carlo construction that has been given very little attention. After having the opportunity to experience the computational burden it imposes, this is hardly surprising. Nevertheless, the performance has been very good and hopefully other researchers will want to get involved in this area. The implementation process was riddled with many challenges and as rewarding as it is to have successful results, there are further aspects that need to be examined before polynomial Hickernell sequences can be applied in practice.

Using the C programming language to implement this method came with a few additional responsibilities. The programmer is required to manage memory, so extreme caution had to be exercised when creating and discarding data structures. This took some attention away from the problem at hand but was a necessary sacrifice to get results in a timely fashion. In addition, we elected to write libraries for polynomial and matrix operations. As a result, tailor-made functions could be written for operations between matrices and polynomials, and we could eliminate unnecessary system calls for memory allocation.

In the beginning, we could not sensibly manage expectations since very little work had been done on this topic. By incorporating ideas from other researchers in related areas we were able to put together a plan for initialising the sequence. The original idea for selecting the g polynomials was to simply use the first s values of the linear recurring sequence. Some of these g's did not have a constant term when expressed in terms of $p(z)$ and this led to singular generator matrices. The associated results were poor and in revising this approach we came up with what is given in Section 4.1. There was a massive improvement in the results and it could rub shoulders with some of the giants in relation to quasi-Monte Carlo constructions. This is very exciting news and augurs well for the future of polynomial Hickernell sequences.

There are a few immediate concerns with the method we have presented. Our implementation was only tested for two classes of problems and more cases will be needed to build confidence in this approach. We are not certain about when to choose different low-dimensional projections. A critical challenge is that as the magnitude of the projections increases, so does the computational burden. Also, we choose g based on a hybrid random search. This means that there exists the possibility that our g is not the best available. Hopefully, these concerns are of sufficient interest to get others involved in developing this concept further.

References

[1] R. E. Caflisch, W. Morokoff, and A. B. Owen. Valuation of mortgage-backed securities using Brownian bridges to reduce effective dimension. *The Journal of Computational Finance*, pages 27–46, 1997.

[2] Henri Faure and Christiane Lemieux. Generalized Halton Sequences in 2007: A Comparative Study. Technical report, 2007.

[3] Christiane Lemieux. *Monte Carlo and Quasi-Monte Carlo Sampling.* Springer, 233 Spring Street, New York, NY, 10013-1578, USA, 2009.

[4] Gottlieb Pirsic and Wolfgang Ch. Schmid. Calculation of the Quality Parameter of Digital Nets and Application to Their Construction. *Journal of Complexity*, 17:827–839, 2001.

[5] Wolfgang Ch. Schmid. Projection of digital nets and sequences. *Mathematics and Computers in Simulation*, 55:239–247, 2001.

[6] I. M. Sobol. The distribution of points in a cube and the appropriate evaluation of integrals. *USSR Comput. Math. Math. Phys.*, 7(4):86–112, 1967.

[7] Shu Tezuka. Polynomial Arithmetic Analogue of Hickernell Sequences. In *Monte Carlo and quasi-Monte Carlo methods 2002*, pages 451–459. Springer, 2004.

[8] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra.* Cambridge University Press, 2nd edition, 2003.