Rank-1 Lattices in Computer Graphics

Sabrina Dammertz, geb. in Ravensburg
Institut für Medieninformatik, 2009
Amtierender Dekan: Prof. Dr. Michael Weber
Gutachter: Dr. Alexander Keller
Gutachter: Prof. Dr. Michael Weber

Abstract

Sampling is fundamental to computer graphics, as samples are used both for representing data and computing averages in the process of forming images. In this context rank-1 lattices are explored with respect to image and texture representation, simulation, and image synthesis. We also contribute the necessary algorithmic and theoretical investigations to find rank-1 lattices, which are suitable for specific requirements. Opposite to common belief, it turns out that rank-1 lattices improve graphics applications quite notably and result in more efficient algorithms, too.
Acknowledgements

Many thanks to

- Holger for his longlasting collaboration, support, and patience
- Alexander Keller for his guidance, encouragement, and support for this PhD
- mental images for the financial support
- the old and new computer graphics group
- Johannes Hanika and Leonhard Grünschloss for proof-reading, and giving valuable feedback
- Carsten Wächter, Matthias Raab, Martin Bader, Leonhard Grünschloss, Johannes Hanika, Manuel Finckh, and Stefan Menz for many helpful discussions
- my family
# Contents

1 Introduction ................................. 1
   1.1 Lattice Structure of Pseudo-Random Number Generators .............. 2
   1.2 Images, Displays, and Sensors with Lattice Structure ............... 5
   1.3 Motivation and Structure of the Thesis ............................... 7

2 Rank-1 Lattices .............................. 9
   2.1 Geometry .................................. 11
      2.1.1 Basis ................................ 11
      2.1.2 Dual Lattice .......................... 16
      2.1.3 Minimum Distance ....................... 18
      2.1.4 Sampling Efficiency of Rank-1 Lattices ..................... 21
      2.1.5 Locating Lattice Points ................... 21
      2.1.6 Shifted Rank-1 Lattices ................. 25
   2.2 Rank-1 Lattice Sequences ................... 25
   2.3 Quasi-Monte Carlo Error Bounds .................. 27
      2.3.1 Classic Quasi-Monte Carlo Error Bounds .................. 27
      2.3.2 Error Bound for Lipschitz Continuous Functions .......... 30
      2.3.3 Randomized Quasi-Monte Carlo Error Bound .............. 32

3 Parameters for Rank-1 Lattices .......... 35
   3.1 Constructions .............................. 35
      3.1.1 Fibonacci Lattice ........................ 35
      3.1.2 Construction by the Continued Fraction Equal to $\sqrt{3}$ ... 36
   3.2 Maximised Minimum Distance Rank-1 Lattices ....................... 37
      3.2.1 Searching on the Unit Square ..................... 39
      3.2.2 MMD Rank-1 Lattice Sequences .................. 48
   3.3 Search for Anisotropic Rank-1 Lattices ....................... 54
   3.4 Weighted Norms .............................. 58
   3.5 $(t,m,2)$-Nets from Shifted Rank-1 Lattices .................... 61

4 Fourier Transform on Rank-1 Lattices .... 69
   4.1 Choosing the Wave Vectors .......................... 70
   4.2 Spectral Synthesis of Ocean Waves ....................... 73
   4.3 Stable Simulation of Fluids ........................ 75
Contents

5 Rasterization on Rank-1 Lattices 79
  5.1 Lines ........................................ 80
  5.2 Circles ...................................... 82

6 Textures on Rank-1 Lattices 87
  6.1 Data Structure ................................ 87
    6.1.1 Nearest Neighbor Look-Up ................ 87
    6.1.2 Linear Interpolation ....................... 88
    6.1.3 Rank-1 Lattice B-Splines of Degree $k$ ... 89
    6.1.4 Tiling and Multi-Resolution ............... 91
    6.1.5 Acquisition ................................ 93
  6.2 Results ..................................... 94
    6.2.1 Maximized Minimum Distance Rank-1 Lattices 94
    6.2.2 Anisotropic Rank-1 Lattices ............... 95

7 Anti-Aliasing by Rank-1 Lattices 99
  7.1 Spectrally Adaptive Sampling .................. 100
  7.2 Jittering Rank-1 Lattices ..................... 101
    7.2.1 Mapping in an Isotropic Way ............... 106
    7.2.2 Results ................................ 108

8 Summary 115
  8.1 Conclusion ................................ 116
  8.2 Future Work ................................ 116

A Source Code 119
  A.1 Search Algorithms ........................... 119
  A.2 Auxiliary Functions ......................... 123

B Lattice Parameter Tables 129
  B.1 MMD Rank-1 Lattices not available in Korobov Form ... 129
  B.2 MMD Rank-1 Lattices with $n = 2^i$ Points for $i = 2, \ldots, 31$ ... 130
  B.3 $(i, m, 2)$-Nets from Shifted Rank-1 Lattices in Base $b$ ... 132
1 Introduction

Lattices are well known structures in everyday life which often are only unconsciously perceived: Ranging from cobbled pavement in pedestrian areas, mesh-wire fences, mosaics and decorations to the screen raster and camera sensors, they are for example also used to partition the playing fields in strategy games. Moreover, they are applied in crystallography, in communication theory for modulation and quantization and in coding and decoding \cite{AEVZ02} and play an important role in sphere packing problems \cite{CSB87}. The most common two-dimensional lattices are the rectangular, the square, and the hexagonal lattice, the last two of which are shown in Figure 1.1. The hexagonal lattice allows the densest packing of circles in a plane, which means that the largest mutual minimum distance between two lattice points in two dimensions is featured in this kind of lattice \cite{CSB87}.

According to their applications in mathematics and computer science many numerical algorithms are based upon them, such as quadrature rules in integration problems and function approximation \cite{Sch97}. Recently, lattices have quite actively been subject of research in the field of quasi-Monte Carlo with respect to high-dimensional numerical integration \cite{KSW06, NC06, CKN06, CN08, SJ08}. In computer graphics lattices are used as sampling patterns for pixel anti-aliasing. The simplest example is placing a sampling point in the center of each pixel. Additionally, in \cite{Gla95} the rectangular, hexagonal, triangular and diamond lattice are examined as sampling techniques. Current display technology incorporates the square lattice in the square pixel layout. Thus, all algorithms involved in pixel processing, as for example rasterization or the accumulation buffer, work on this lattice. Furthermore, sampling lattices are basic tools for the digitalization of continuous data. Whereas the square lattice is mostly used in this context, \cite{MS05} give a thorough survey

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{lattices.png}
\caption{Example for rank-2 lattices.}
\end{figure}
1.1 Lattice Structure of Pseudo-Random Number Generators

Random numbers appear in a wide range of applications, as for example in numerical analysis and cryptography. A sequence of random numbers has to fulfill the following properties: the random numbers must be unpredictable, independent from each other, and uniformly distributed within a given domain. However, real random numbers are expensive and difficult to generate on standard computer hardware. Therefore usually pseudo-random generators are used, which try to mimic random behavior by deterministic algorithms. As a consequence, the resulting pseudo-random numbers are predictable and only approximately independent.

A very common and popular principle for the generation of pseudo-random numbers is the linear congruential generator (LCG) \[X_{k+1} = (aX_k + c) \mod m, \quad m > 0, k \geq 0,\] each of which is described by the tuple \((a, c, m, X_0)\), where \(a \in \{1, \ldots, m-1\}\) is the multiplier, \(c \in \{0, \ldots, m-1\}\) is the increment, \(m \in \mathbb{N}^+\) is the modulus, and \(X_0 \in \{0, \ldots, m-1\}\) is the initial value. Dividing each \(X_k\) by the modulus \(m\) yields numbers in the unit interval \([0, 1)\). Due to the modulus the set of generated numbers is finite and therefore must expose a period.

This periodicity can be analyzed and visualized by an important property of LCGs, which is that the set of all \(m\) points
\[
\{(X_k, X_{k+1}, \ldots, X_{k+s-1}) | 0 \leq k < m\},
\]
which are called overlapping \(s\)-tuples, conforms to the intersection of an \(s\)-dimensional integer lattice with the hypercube \([0, m]^s\) \[LC97\]. This lattice structure is illustrated for dimension \(s = 2\) in Figure 1.2 using a LCG with parameters \(a = 137, c = 187, \ m = 256\) and \(X_0 = 0\) \[Knu81\].

Figure 1.2: Two-dimensional shifted lattice by all overlapping 2-tuples \((X_n, X_{n+1})\) for the linear congruential generator (LCG) \(X_{n+1} = (137 \cdot X_n + 187) \mod 256\).

of the hexagonal lattice in the field of image processing (see Section 1.2).
The lattice structure even shows up in the more restricted, but simpler setting of multiplicative congruential generators: For \(c = 0\), a prime modulus \(m\) and a multiplier \(1 < a < m\) the LCG has a maximal period length of \(m - 1\). Note that in this case the seed \(X_0 = 0\) has to be excluded and the origin has to be added to the point set for a total number of \(m\) points. If the LCG fulfills these assumptions the set (1.2) can be expressed as

\[
\left\{ \frac{1}{m} (x, r(x), r(r(x)), \ldots, r^{k-1}(x)) \mid 0 \leq x < m \right\},
\]

for the recurrence

\[
r(x) = (ax) \mod m.
\]

Inserting the recurrence (1.4) into (1.3) reveals that this set actually represents a rank-1 lattice in Korobov form [Nie92]

\[
\left\{ \frac{1}{m} (x, ax, a^2 x, \ldots, a^{s-1} x) \mod 1 \mid 0 \leq x < m \right\},
\]

which we will analyze in great depth in the next chapter. This means that the set (1.3) corresponds to the set of all vectors of \(s\) successive values which result from the recurrence (1.4) [LC97], starting from all possible seeds \(0 \leq x < m\).

Figure 1.3 shows a sequence of images from [Ent01], where for a fixed number \(n\) of points the modulus \(m\) and multiplier \(a\) are adapted such that the period of the LCG is decreased until the modulus corresponds to the number of points \(n\), yielding a Korobov rank-1 lattice. The image series thus illustrates the transition from pseudo-random to regular structures by only changing the set of parameters of one algorithm.

**From Monte Carlo to Quasi-Monte Carlo**

Pseudo random numbers are used in numerical integration in order to cope with the curse of dimensions that arises from tensor product quadrature [Sch97]. The Monte Carlo method allows one to perform numerical integration of high-dimensional integrals independent of the dimension \(s\) for square integrable functions \(f\) on the \(s\)-dimensional unit cube \(I^s = [0, 1)^s\). The key idea is to construct a random variable whose expected value is the desired integral. Then the integral

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

is approximated by averaging \(n \in \mathbb{N}\) independent identically distributed (i.i.d) realizations of the random variable with probability density \(p(x) = 1\) over the domain \([0, 1)^s\). Using the central limit theorem and the strong law of large numbers a probabilistic error bound can be derived [Nie92], which states that the error is less than \(\frac{3\sigma(f)}{\sqrt{n}}\) with a probability of

\[
P \left( \left\{ \int_{I^s} f(x) dx - \frac{1}{n} \sum_{i=0}^{n-1} f(x_i) \right\} < \frac{3\sigma(f)}{\sqrt{n}} \right) \approx 0.997.
\]
Figure 1.3: Visualization of the transition from pseudo-random to quasi-random samples from a multiplicative linear congruence generator. While the number of samples is fixed to \( n = 2^{10} - 3 = 1021 \), the multiplier \( a \) and modulus \( m \) and thus the generator period are varied making the underlying lattice structure more and more obvious.
The proof takes advantage of the square integrability of \( f \) and the independence of the random variables. Moreover, point sampling the integrand at positions \( x_i \) has to be possible except for sets of zero measure. Note that \( \sigma(f) \) corresponds to the standard deviation of the function \( f \).

However, the independence of the sampling points, which is crucial for the proof of the error bound (1.7) of the Monte Carlo method, is not necessary for convergence. Neglecting the unpredictability and independence allows to design sampling patterns which are much more uniform than random numbers can be. Using such deterministic point sets in equation (1.6) yields a technique called quasi-Monte Carlo. The supplement quasi refers to the fact that the algorithm for estimating the integral (1.6) is the same as for the Monte Carlo method. Contrary to Monte Carlo, quasi-Monte Carlo provides deterministic error bounds, the most common of which are dealt with in Section 2.3.

If a whole function

\[
g(y) := \int f(x, y) \, dx \approx \frac{1}{n} \sum_{i=0}^{n-1} f(x_i, y)
\]

is estimated instead of a functional, this is called integro-approximation. In the case of Monte Carlo integration the generalization for random functions is said to be the method of dependent tests. For more thorough introduction to Monte Carlo and quasi-Monte Carlo see for example [Nie92, SJ94, Owe98].

The important observation now is that a pseudo-random number generator can be used for both Monte Carlo and quasi-Monte Carlo integration: Looking at the results from the previous section [ES01], using only a small portion of an LCG with a huge period generates overlapping \( s \)-tuples suitable for Monte Carlo integration, while using an LCG with rank-1 lattice structure corresponds to quasi-Monte Carlo integration. Numerical integration thus represents a perfect example to illustrate the transition from pseudo- to quasi-random numbers and gives reason to take a closer look at the exposed lattice structure. The same is true for function approximation, which is discussed in the next section.

### 1.2 Images, Displays, and Sensors with Lattice Structure

Displays and images are typically characterized by square pixels. Alternatively to the conventional square grid, hexagonal sampling grids have been investigated for more than 40 years, giving rise to the field of hexagonal image processing. With applications in medical imaging, cameras or hexagonal displays, hexagonal image processing (HIP) is also quite an active field in computer vision. [MS05] gives a comprehensive survey of research on this topic and presents a framework for hexagonal image processing based on hierarchical aggregates. A lot of research has been performed with respect to generating hexagonally sampled images. For example [MS05, VWPL02, CVFH08] address the problem of resampling between orthogonal and hexagonal images. However, the hexagonal lattice has not been commonly used in computer graphics so far and there are no major applications with respect to textures for rendering. In two dimensions hexagonal lattices have been examined independently in the context of digital halftoning by [JO03] and [LU04], [RLAL06]. In three dimensions non-Cartesian lattices have been studied with respect to the visualization...
Figure 1.4: Comparison of rank-2 (left) and maximized minimum distance rank-1 (right) lattice displays.

of volumetric data. [EVM08] use box splines in order to efficiently reconstruct volumetric data sampled on the Body Centered Cubic (BCC) lattice, taking advantage of its optimal spectral sphere packing property. In [NM02] 4D BCC grids are examined in the context of visualizing 4D data sets. [Csb05] proposes a novel high-quality reconstruction scheme to reconstruct volumetric data sampled on an optimal BCC grid.

Both square and hexagonal pixels (called hexels) have the property of tiling the plane. This means that hexagonally and square sampled images can be considered as a periodic monohedral tiling [MS05]. The tiles represent the picture elements, being the Voronoi cells of the sampling points. Similar to the hexagonal lattice a rank-1 lattice tiles the plane by its Delaunay triangulation and Voronoi diagram, which is illustrated in Figure 2.2 of Section 2.1. This leads to the idea of structuring the pixel layout by the Voronoi cells of a rank-1 lattice. The concept of images on maximized rank-1 lattices (see Section 3.2) has already been outlined coarsely in [Ski05]. However, there has been no realization or implementation up to now. Chapter 6 describes how such images can be generated and accessed.

This kind of display and image layout has several advantages.

- Contrary to the hexagonal and square lattice, rank-1 lattices perfectly tile the unit square \( I = [0, 1)^2 \) periodically for any number \( n \) of points. Therefore more flexibility is offered with respect to the shape of the image domain and the number of pixels can be chosen freely. This is illustrated in Figure 3.3, where the square and hexagonal lattices for \( n = 56 \) points do not tile the unit square.

- As shown in Figure 3.3, maximizing the minimum distance approximates the hexagonal grid. As a consequence they yield almost hexagonal picture elements which can not visually be distinguished from hexagonal pixels anymore. Thus, rank-1 lattices benefit from the strengths of the hexagonal lattice, such as a higher sampling efficiency (see Section 2.1.4) as compared to the square lattice [MS05, PM62], a better angular resolution as there are more nearest neighbors, i.e. adjacent lattice cells, and permits optically better results than the traditional square lattice. This results for ex-
ample in a smoother representation of curved objects. This can be seen in Figure 1.4, where the original image (middle) has been computed in reduced resolution, once on a traditional rank-2 lattice and once on a maximized minimum distance rank-1 lattice.

- Whereas the hexagonal lattice is defined on the irrational numbers (due to $\sqrt{3}$), rank-1 lattices live on the rational numbers and therefore naturally allow for integer arithmetic.

- Image processing algorithms which are based upon the fast Fourier transform become simpler and can be implemented in a more efficient way (see Chapter 4).

However, both vertical and horizontal lines can not be displayed smoothly anymore and thus have a more jagged appearance than on the square lattice for example.

As proposed in [Ski05] the concept of rank-1 lattice displays can technically be realized in a number of ways, including point (e.g. RGB LEDs) and area light sources (e.g. OLEDs) and thus offers applications for TFT and LCD displays, CCD cameras, projector technology, 3d-Displays, etc. Due to the periodicity of rank-1 lattices a display can be assembled of several modules each of which having the same number of pixels. A special case of rank-1 lattice display modules consists in choosing the number of picture elements as a power of 2. This has the advantage that a demultiplexer can be used in order to address the single cells. If the number of modules equals a power of 2 as well, the same control mechanism can be applied for the single modules. This concept perfectly fits all aspects of computer technology, taking advantage of memory layout, cache lines, addressing, etc.

1.3 Motivation and Structure of the Thesis

Rank-1 lattices possess a very simple structure which allows them to be generated in a very fast way. As they are very flexible without the need of changing the simplicity of their representation and as they can be processed using integer arithmetic, they represent an interesting starting point for handling applications in computer graphics.

Therefore this thesis explores the use of rank-1 lattices in computer graphics by the following topics:

**Chapter 2** defines and introduces rank-1 lattices by illustrating their geometric properties involved in the following chapters. Moreover, a new error bound for the integration of Lipschitz continuous functions by rank-1 lattices is derived, which motivates one of the two search principles in order to determine the generator vector of a rank-1 lattice proposed in the next chapter.

**Chapter 3** Additionally, to so-called maximized minimum distance rank-1 lattices, the search of anisotropic lattices is examined. Finally, the chapter combines and evaluates the features of rank-1 lattices and $(t,m,s)$-nets in $(t,m,2)$-lattices.
Chapter 4 shows that the $s$-dimensional Fourier transform on rank-1 lattices reduces to only one dimension, which thus allows for an efficient implementation. This is demonstrated for two applications in computer graphics and simulation.

Chapter 5 and 6 deal with the use of rank-1 lattices with respect to image and display technology by first illustrating rasterization algorithms for lines and circles on this kind of lattices.

Chapter 6 continues by introducing the necessary data structures and access methods for representing textures on rank-1 lattices and concludes with the handling of large textures.

Chapter 7 proposes two new anti-aliasing algorithms using rank-1 lattices. The first one takes advantage of the spectrum of the image function in order to improve anti-aliasing, whereas the second one represents a new method in order to improve jittered sampling by rank-1 lattices.

Note that the images at the bottom of each page are the maximized minimum distance rank-1 lattices for the corresponding page numbers.
2 Rank-1 Lattices

Figure 2.1: Generating a rank-1 lattice for $n = 8$ points using the generator vector $\mathbf{g} = (1, 3)$. From left to right the number of lattice points increases from one to eight points. For each image the multiple $\{ \frac{i}{8} \cdot \mathbf{g} \}_1, i = 0, \ldots, 7$ of the generator vector $\mathbf{g}$ is plotted. The rightmost image in the lower row represents the whole lattice $L_{8,(1,3)}$.

Rank-1 lattice rules were first proposed by Korobov [Kor59] and have been widely examined since then [Nie92, SJ94, HW81]. The points $\mathbf{x}_i$ of an $s$-dimensional rank-1 lattice [Nie92, SJ94] in the unit cube $I^s = [0, 1)^s$

$$L_{n,\mathbf{g}} := \left\{ \mathbf{x}_i := \left\{ \frac{i}{n} \mathbf{g} \right\}_1 \mid i = 0, \ldots, n-1 \right\}$$

are generated by using one suitable integer generator vector $\mathbf{g} \in \mathbb{N}^s$ for a fixed number $n \in \mathbb{N}$ of points, where $\{ \mathbf{x} \}_1$ denotes the fractional part of a vector $\mathbf{x}$, i.e. $\{ \mathbf{x} \}_1 := \mathbf{x} \mod 1$. The $mod \ 1$ operation restricts the lattice to the unit square resulting in a one-periodic pattern. This means that rank-1 lattices are point symmetric to the origin. The generation of a rank-1 lattice is illustrated in Figure 2.1, where for each point $\mathbf{x}_i$ the multiple of the generator vector yielding this point is plotted.
Generally, a lattice $L$ in $\mathbb{R}^s$ is a discrete subset of $\mathbb{R}^s$ which is closed under addition and subtraction and always contains the origin. An integration lattice in $\mathbb{R}^s$ is defined to additionally contain $\mathbb{Z}^s$ as a subset. A so-called lattice rule is a quadrature rule of the form

$$Qf = \frac{1}{n} \sum_{i=0}^{n-1} f(x_i)$$

for approximating the integral over the unit cube for a function $f$, where $x_0, \ldots, x_{n-1} \in L \cap [0,1)^s$ are the points of an integration lattice [SJ94].

Lattices can be classified by their rank $r$, which is described by the following theorem [SJ94]:

**Theorem 1.** Let $Q$ be any $s$-dimensional lattice rule with a number of $n \geq 2$ points. Then there exists a uniquely determined integer $r$ (the “rank”) with $1 \leq r \leq s$ and uniquely determined integers $n_1, \ldots, n_r > 1$ satisfying

$$n_{k+1} \text{ divides } n_k, \quad 1 \leq k \leq r - 1,$$

such that $Q$ can be expressed in the form

$$Qf = \frac{1}{n} \sum_{j_r=0}^{n_r-1} \cdots \sum_{j_1=0}^{n_1-1} f \left( \left\{ \frac{j_1}{n_1} z_1 + \cdots + \frac{j_r}{n_r} z_r \right\} \right),$$

where $z_1, \ldots, z_r$ are linearly independent integer vectors, and the order of the rule is

$$n = n_1 n_2 \cdots n_r.$$
2.1 Geometry

Figure 2.2 illustrates the geometrical properties of rank-1 lattices which are described in this section for the example of $n = 32$ points and the generator vector $g = (\frac{1}{7})$. The solid lines depict the Voronoi diagram. Each cell is generated by a lattice point and contains the points of the unit torus, which are closer to this lattice point than to any other. In addition the lattice points are the centroids of the Voronoi cells. The dashed lines represent the dual graph, i.e. the Delaunay tessellation, which can be generated by dividing the fundamental parallelepiped along its shorter diagonal. The fundamental parallelepiped is spanned by the basis vectors $b_1$, $b_2$, and the origin. The length of the first basis vector corresponds to the minimum distance of the lattice, i.e. to the minimum mutual distance between two lattice points.

2.1.1 Basis

An $s \times s$ matrix $B = (b_1 \ldots b_s)$ is called a basis of the lattice $L$, if

$$L = \{x = B \cdot l : l \in \mathbb{Z}^s\},$$

i.e. if every point in the lattice can be generated by an integer linear combination of the basis vectors $b_1, \ldots, b_s$. The number $s$ of basis vectors is said to be the dimension of $L$.

The basis vectors and the origin span the so-called fundamental parallelepiped

$$\Lambda = \Lambda(b_1, \ldots, b_s) = \{\gamma_1 b_1 + \ldots + \gamma_s b_s : 0 \leq \gamma_i \leq 1, 1 \leq i \leq s\}, \quad (2.1)$$

the volume of which corresponds to $|\det(B)|$. The fundamental parallelepiped induces a partition of $\mathbb{R}^s$ into lattice cells of the same volume and orientation:
\[ \{ \mathbf{x} + \Lambda \} = \{ \mathbf{y} \in \mathbb{R}^d | \mathbf{y} = \mathbf{x} + \mathbf{z}, \text{ where } \mathbf{z} \in \Lambda \} | \mathbf{x} \in L \}. \quad (2.2) \]

A lattice has infinitely many different bases which do not vary in their determinants, however. This means that \( \det(L) := |\det(B)| \) is invariant for each lattice \( L \), therefore being called the lattice determinant. Hence, a matrix \( B \) of linear independent vectors only represents a valid basis for \( L \) if its determinant equals \( \det(L) \). As the application of unimodular transformations \( \mathbf{x} = \mathbf{A}\mathbf{x} \) with \( |\det(A)| = 1 \) for a square integer matrix \( A \) does not change \( \det(L) \), a lattice basis is unique with respect to such transformations, as for example swapping two basis vectors [BW02].

**Basis Reduction**

As a lattice does not have a unique basis an interesting task in lattice theory consists in finding “good” bases according to a given criterion. This process is called basis reduction. There are different notions of reduced bases in lattice theory, which often refer to the length of the basis vectors in order to find a basis with short and nearly orthogonal vectors.

A **Hermite reduced** basis \( B = (\mathbf{b}_1 \ldots \mathbf{b}_s) \) fulfills the condition

\[
L(\mathbf{b}_1, \ldots, \mathbf{b}_s) = L(\mathbf{b}_i', \ldots, \mathbf{b}_i') \quad \text{and} \quad ||\mathbf{b}_j'|| = ||\mathbf{b}_j|| \quad \text{for } j \in \{1, \ldots, i-1\} \Rightarrow ||\mathbf{b}_i|| \leq ||\mathbf{b}_i'||
\]

for all \( i \in \{1, \ldots, s\} \) and for all \( B' = (\mathbf{b}_1' \ldots \mathbf{b}_s') \) [AEVZ02]. Therefore a basis matrix \( B \) is reduced if the sequence \( ||\mathbf{b}_1||, \ldots, ||\mathbf{b}_i|| \) is the first element in a list where the corresponding sequences of all basis matrices of the same lattice are lexicographically ordered. In such a reduced basis the shortest non-zero lattice vector is given by the first basis vector. Although every lattice has a **Hermite reduced basis**, there exists no explicit construction.

In 1905 Minkowski introduced a less strict reduction criterion. A basis \( B \) is **Minkowski reduced** if \( \mathbf{b}_1 \) is the shortest non-zero vector in \( L \) and for \( 2 \leq i \leq s \) \( \mathbf{b}_i \) is the shortest vector in \( L \) such that \( (\mathbf{b}_1, \ldots, \mathbf{b}_i) \) may be extended to a basis of \( L \), i.e. \( \{\mathbf{b}_1, \ldots, \mathbf{b}_i\} \) is a so-called primitive set. Contrary to the Minkowski reduction, Korkhine and Zolotarav require the component \( \mathbf{b}_i' \) of \( \mathbf{b}_i \) which is orthogonal to the vector space spanned by \( \{\mathbf{b}_1, \ldots, \mathbf{b}_{i-1}\} \) to be as short as possible under the condition of \( \{\mathbf{b}_1, \ldots, \mathbf{b}_i\} \) keeping the primitive set property [Kan87] (KZ reduction). Lenstra, Lenstra and Lovász described a polynomial-time reduction algorithm similar to but weaker than the KZ reduction, resulting in a non-zero vector with length at most \( 2^{n/2} \) the length of the shortest vector [LJL84, Kan87]. This classical algorithm represents the foundation for many other algorithms in Algorithmic Geometry of Numbers (see for example [Kan87, Car02]).

For two dimensions there exists a very simple and classical algorithm, namely the Gaussian reduction [Kan87, Car02], which runs in a number of \( O(\log n) \) algorithmic steps [Rot97]. The Gaussian reduction represents one of the earliest reduction algorithms for \( s = 2 \). Note that the concepts of Minkowski and KZ reduced basis are equivalent in two dimensions and that Gaussian reduction finds the corresponding bases [Kan87].

Let \( \{\mathbf{b}_1, \mathbf{b}_2\} \) be a basis of a lattice \( L \), with \( ||\mathbf{b}_1|| \leq ||\mathbf{b}_2|| \), which is assured by renaming the basis vectors if necessary. Then the transformation \( U \)

\[
\hat{B} = U \cdot B = \begin{pmatrix} 1 & 0 \\ -\lambda & 1 \end{pmatrix} \cdot B
\]
with $\lambda \in \mathbb{Z}$ yields a new basis $\hat{B} = \begin{pmatrix} \hat{b}_1 \\ \hat{b}_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 - \lambda b_1 \end{pmatrix}$ for $L$, since for any $x = s \cdot b_1 + t \cdot b_2 \in L$ we have

$$\begin{align*}
x &= \alpha \hat{b}_1 + \beta \hat{b}_2 \\
&= \alpha b_1 + \beta (b_2 - \lambda b_1) \\
&= (\alpha - \beta \lambda) b_1 + \beta b_2 \\
\implies \alpha &= s + t \cdot \lambda \quad \text{and} \quad \beta = t
\end{align*}$$

and therefore

$$x = s \cdot b_1 + t \cdot b_2 = (s + t \cdot \lambda) \cdot \hat{b}_1 + t \cdot \hat{b}_2.$$ 

The shortest vector $\hat{b}_2 = b_2 - \lambda b_1$ is achieved by the component of $b_2$ perpendicular to $b_1$. Setting $\lambda$ to the nearest integer to $\frac{b_1 \cdot b_2}{||b_1||^2}$ the second basis vector $b_2$ is replaced by $\hat{b}_2$

$$\hat{b}_2 = b_2 - \left[ \frac{b_1 \cdot b_2}{||b_1||^2} \right] b_1.$$ 

If for this new $b_2$ we have $||b_2|| < ||b_1||$, the vectors are swapped and the procedure is repeated. Otherwise the algorithm is stopped, returning a reduced basis with the shortest vector in $L$ as the first basis vector.

Obviously the Gaussian reduction terminates in a finite amount of steps, since at each step the length of $b_1$ is reduced, and there are only a finite number of lattice points $x_i$ with $||x_i|| \leq ||b_1||$.

In the following we use the Gaussian reduction in order to compute the basis of a two-dimensional rank-1 lattice $L_{n,g}$. For that purpose an initial basis has to be determined for $L_{n,g}$, before being able to apply the reduction.

This task is easy in the case of Korobov lattices. Let

$$X_k = aX_{k-1} \mod n, \quad n \geq 0$$

be the linear congruential generator corresponding to $L_{n,g}$, with $g = (1, a, \ldots, a^{n-1})$. Then a basis can be constructed as follows [LC97]: Let $e_i(j)$ describe the $i$-th unit vector in dimension $j$. The first component of $b_1$ is made up of $e_{1}(1) = 1$, while for the other components we have $b_{1,k} = X_k = aX_{k-1}$, where $X_0 = e_{1}(1)$. This means that the first basis vector is given by the generator vector $g$. For $j > 1$

$$b_j = n \cdot e_{i(j)}.$$ 

So for $s = 2$ a valid basis simply is

$$b_1 = g = (1, a) \quad \text{and} \quad b_2 = (0, n).$$

Rote described a solution to finding a basis for a general two-dimensional rank-1 lattice $L_{n,g}$, $g = (g_x, g_y) \in \mathbb{N}^2$ in [Rot97]: Evidently a rank-1 lattice $L_{n,g}$ can be generated by

$$x = s \cdot b_1 + t \cdot b_2 = (s + t \cdot g_x) \cdot \hat{b}_1 + t \cdot \hat{b}_2,$$
three vectors \( \mathbf{u} := (n, 0) \), \( \mathbf{v} := (0, n) \), and \( \mathbf{w} := (0, 0) \). As they are linear dependent and rational, there exist integers \( t_1, t_2 \) and \( t_3 \) with their greatest common divisor being equal to one, i.e. \( \gcd(t_1, t_2, t_3) = 1 \),

\[
\begin{align*}
t_1 \mathbf{u} + t_2 \mathbf{v} + t_3 \mathbf{w} &= 0,
\end{align*}
\]

which immediately can be set to \( t_1 := n, t_2 := -g_x \) and \( t_3 := -g_y \). Note that the generator vector has to fulfill the condition \( \gcd(n, g_x, g_y) = 1 \). The vector \( (t_1, t_2, t_3) \) is used to build a unimodular matrix

\[
A = \begin{pmatrix}
t_1 & \alpha_1 & \beta_1 \\
t_2 & \alpha_2 & \beta_2 \\
t_3 & \alpha_3 & \beta_3
\end{pmatrix},
\]

i.e. an integer matrix with \( \det(A) = 1 \). Then a lattice basis can be expressed as \( \mathbf{b}_1 = \alpha_1 \mathbf{u} + \alpha_2 \mathbf{v} + \alpha_3 \mathbf{w} \) and \( \mathbf{b}_2 = \beta_1 \mathbf{u} + \beta_2 \mathbf{v} + \beta_3 \mathbf{w} \), which is equivalent to

\[
(\mathbf{u}, \mathbf{v}, \mathbf{w})A = (0, \mathbf{b}_1, \mathbf{b}_2).
\]

As \( \mathbf{b}_1 \) and \( \mathbf{b}_2 \) are computed by integer linear combinations of \( \mathbf{u}, \mathbf{v} \) and \( \mathbf{w} \), obviously \( L(\mathbf{b}_1, \mathbf{b}_2) \subseteq L(\mathbf{u}, \mathbf{v}, \mathbf{w}) \). The other way round, the vectors \( \mathbf{u}, \mathbf{v}, \mathbf{w} \) can be constituted as a linear combination of \( \mathbf{b}_1, \mathbf{b}_2 \) and an integer matrix \( A^{-1} \),

\[
(\mathbf{u}, \mathbf{v}, \mathbf{w}) = (0, \mathbf{b}_1, \mathbf{b}_2)A^{-1},
\]

which means that \( L(\mathbf{u}, \mathbf{v}, \mathbf{w}) \subseteq L(\mathbf{b}_1, \mathbf{b}_2) \). Hence the vectors \( \mathbf{b}_1 \) and \( \mathbf{b}_2 \) really build a basis of \( L \).

By means of the extended Euclidean algorithm (see for example [Sch01]) the integers \( \alpha_i \) and \( \beta_i \) are selected such that \( \det(A) = 1 \) [Hua82, Rot97]. This is illustrated in Listing 2.1, which summarizes the whole process of computing a reduced basis of a rank-1 lattice using the Gaussian lattice basis reduction. As the extended Euclidean algorithm runs in a number of \( \mathcal{O}(\log n) \) steps, the total runtime of Listing 2.1 corresponds to \( \mathcal{O}(\log n + \log n) = \mathcal{O}(\log n) \).

```python
def gaussianBasisReduction(n, gx, gy):
    # initialize basis
    if (gx == 1):
        # lattice in Korobov form with g = (1, a)
        b1x = gx
        b1y = gy
        b2x = 0
        b2y = n
    elif (gy == 1):
        # lattice in Korobov form with g = (a, 1)
        b1x = gx
        b1y = gy
        b2x = n
        b2y = 0
    else:
        a2, a1, d = extended_euclid(n, gx)
        b3, gamma, d1 = extended_euclid(d, gy)
        b1 = n // d * gamma
        b2 = -gx // d * gamma
        b1x = a1 * gx + a2 * n
        b1y = a1 * gy
```

14
\[ b_{2x} = b_1 \ast gx + b_2 \ast n \]
\[ b_{2y} = b_1 \ast gy + b_3 \ast n \]

# perform reduction
\[ l_1 = b_{1x}b_{1x} + b_{1y}b_{1y} \]
\[ l_2 = b_{2x}b_{2x} + b_{2y}b_{2y} \]

while True:
    if (l1 > l2):
        tmpx, tmpy = b1x, b1y
        b1x, b1y = b2x, b2y
        b2x, b2y = tmpx, tmpy
        t = float(b1x*b2x + b1y*b2y) / float(b1x*b1x + b1y*b1y)
        b2x = (mu*b1x)
        b2y = (mu*b1y)
        l1 = b_{1x}b_{1x} + b_{1y}b_{1y}
        l2 = b_{2x}b_{2x} + b_{2y}b_{2y}
    if (l1 <= l2):
        break

return b1x, b1y, b2x, b2y

Listing 2.1: The algorithm computes the reduced basis of the lattice \( L_{n,g} \) with \( g = (gx, gy) \).

**Convention**

Like in [Ski05], of all possible lattice bases the Minkowski reduced bases are the most useful for our purpose. Such a basis contains the \( s \) shortest linearly independent vectors. General algorithms for its computation in high dimensions can be found in [AG85, Hel85] for example. Depending on the application we either take the approach of the Gaussian basis reduction, as it exactly yields a Minkowski reduced basis in two dimensions as stated above, or we use an exhaustive search. In the latter case we apply a computer search over all \( n \) points \( x_i \) using their shortest distance to the origin on the unit torus, which is quite similar to algorithm 2.2. This method has the advantage to be able to keep track of the indices \( i \) of the basis vectors which are not obvious for lattices not being in Korobov form (see Section 2.1.5). Whereas the Gaussian reduction is used in Section 3.5 in order to search for \((t,m,2)\)-lattices, in Chapters 5 and 6 the exhaustive approach is applied for addressing on rank-1 lattices.

In contrast to the basis vectors of the square and hexagonal lattice which are uniquely defined, the basis layout may differ for each rank-1 lattice. In order to avoid distinction of cases for neighborhood computations we demand a lattice basis to obey the following convention. Thereby we only consider two-dimensional lattices and bases. Based on a Minkowski reduced basis \( B' = (b_1',b_2') \top \) the angle between \( b_1' \) and \( b_2' \) is to be less than \( 90^\circ \):

\[
(b_1,b_2) = \begin{cases} 
(b_1',b_1' + b_2') & \text{if } b_1' \cdot b_2' < 0 \\
(b_1',b_2') & \text{otherwise} 
\end{cases}
\]  

(2.3)
Let \( \mathbf{b}_1 \) and \( \mathbf{b}_2 \) be a basis of \( L_n \). Setting \( \mathbf{b}_2' = \mathbf{b}_1 + \mathbf{b}_2 \) still yields a valid basis of the lattice, since

\[
|\det(\mathbf{b}_1, \mathbf{b}_1 + \mathbf{b}_2)| = |b_{11}(b_{12} + b_{22}) - b_{12}(b_{11} + b_{21})| = |(b_{11}b_{22} - b_{12}b_{21}) + (b_{11}b_{12} - b_{12}b_{11})| = |\det(\mathbf{b}_1, \mathbf{b}_2)| = \det(L) = \frac{1}{n}.
\]

The convention is illustrated in Figure 2.3, where \( \{\mathbf{b}_1', \mathbf{b}_2'\} \) corresponds to the original Minkowsky reduced basis and \( \{\mathbf{b}_1, \mathbf{b}_2\} \) is the final result.

Given any two basis vectors \( \mathbf{b}_1 \) and \( \mathbf{b}_2 \) sometimes a third (redundant) basis vector is required (for example for address calculations on rank-1 lattices in Section 2.1.5) which is computed by

\[ \mathbf{b}_3 = \mathbf{b}_2 - \mathbf{b}_1 \]

This vector then corresponds to the former vector \( \mathbf{b}_2' \) if \( \mathbf{b}_1' \cdot \mathbf{b}_2' < 0 \) and we have \( |\det(\mathbf{b}_1, \mathbf{b}_3)| = \frac{1}{n} \) as well as \( |\det(\mathbf{b}_2, \mathbf{b}_3)| = \frac{1}{n} \). Dependent on the application \( \mathbf{b}_1 \) and \( \mathbf{b}_2 \) are directly chosen as the two shortest linearly independent lattice vectors or are additionally determined to obey the convention by equation (2.3).

### 2.1.2 Dual Lattice

Given a lattice \( L \), which is also called primal lattice, the dual or reciprocal lattice \( L^\perp \) is defined as the set of real vectors \( \mathbf{h} \in \mathbb{R}^d \) with

\[
L^\perp := \{ \mathbf{h} \in \mathbb{R}^d | \mathbf{h} \cdot \mathbf{x} \in \mathbb{Z} \text{ for all } \mathbf{x} \in L \}.
\]
If the basis $B$ of $L$ is known, the basis $B^\perp$ of the dual lattice $L^\perp$ arises as the transpose of the inverse of $B$, i.e. $B^\perp := (B^{-1})^T$, and the reciprocal of the determinant of $L$ yields the lattice determinant of $L^\perp$.

The geometric interpretation of the dual lattice refers to the hyperplanes, i.e. planes of dimension $s-1$, which cover all points of $L$ [SJ94]. For a lattice $L$ there exist many different possibilities to place families of equally spaced parallel hyperplanes such that every point in $L$ lies on one of the hyperplanes and each hyperplane holds at least one point of $L$. Thereby each set of hyperplanes is connected to a point $h$ in the dual lattice by the relation

$$\{ x | h \cdot x = \delta \}, \text{ for } \delta = 0, \pm 1, \pm 2, \ldots, \text{ and } x \in L$$

(2.5)

and $h$ corresponds to the vector being perpendicular to those planes. The distance between each plane is computed as the reciprocal of the Euclidean norm of $h$. This geometric interpretation is visualized in Figure 2.4 for the rank-1 lattice $L_{5, (1,3)}$ along with the family of hyperplanes defined by the dual lattice point $h = (1, 2)$.

Figure 2.4: The rank-1 lattice $L_{5, (1,3)}$ (left) and its dual lattice (right) [Lem00, Chapter 2, p. 25].

Let $x \in L_n$. Based on equation (2.4), for a rank-1 lattice we have

$$h \cdot x = h \cdot i \cdot g$$

$$= \frac{i}{n} (h \cdot g) \in \text{def } \mathbb{Z}$$

$$\iff h \cdot g = j \cdot n \text{ for } j \in \mathbb{N}$$

$$\iff h \cdot g \equiv 0 \text{ (mod } n)$$

$$\implies h \in \mathbb{Z}^\prime.$$

This means that for rank-1 lattices the dual lattice simplifies to [SJ94]

$$L^\perp = \{ h \in \mathbb{Z}^\prime | h \cdot g \equiv 0 \text{ (mod } n) \}.$$
Due to $B^\perp = (B^{-1})^\top$ for a two-dimensional rank-1 lattice $L_{n,g}$ its dual lattice $L^\perp$ results as the infinitely expanded lattice $L_{n,g} + z, z \in \mathbb{Z}^s$ rotated by 90°. For Korobov lattices $L_{n,a}$ we can even show that the dual lattice $L^\perp \cap [0,n)^2$ corresponds to a Korobov rank-1 lattice, $L_{n,a'}$, which is scaled by $n$, again. The proof is based on the fact that the two generator vectors $g = (1,a)$ and $g^\perp = (1,a')$ are perpendicular to each other, i.e. that their dot product equals 0 modulo $n$.

$$(1,a') \cdot (1,a)^\top \equiv 0 \pmod{n}$$
$$1 + a' \cdot a \equiv 0 \pmod{n}$$
$$a' \cdot a = k \cdot n - 1 \quad \text{for } k \in \mathbb{N}$$
$$a' = \frac{k \cdot n - 1}{a} \quad \text{for } a' \in \mathbb{N}$$
$$\iff k \cdot n - 1 \equiv 0 \pmod{a}$$
$$\iff k \cdot n \equiv 1 \pmod{a}$$

The parameter $k \in \mathbb{N}$ can be computed as the multiplicative inverse by means of the extended Euclidean algorithm.

### 2.1.3 Minimum Distance

The quality of a point set $P_n$ can be measured by the mutual minimum distance

$$d_{\text{min}}(P_n) := \min \min_{0 \leq i < n, i < j < n} \|x_j - x_i\|_2$$

of the sample points using the Euclidean distance on the unit torus. Contrary to the discrepancy the minimum distance is isotropic and shift-invariant. In [Kel06] Keller computed the minimum distance of some common quasi-Monte Carlo point sets and observed that point sets having a small discrepancy also feature a large minimum distance.

Calculating the minimum distance in a lattice is known as the shortest vector problem in lattice theory [AEVZ02]. Since a lattice is closed under addition and subtraction the difference between two lattice points yields another point in the lattice and the minimum distance corresponds to the length of the shortest vector in the lattice. Therefore the minimum distance

$$d_{\text{min}}(L_{n,g}) := \min_{0 \leq i < n} \|x_i\|$$

of a rank-1 lattice $L_{n,g}$ is simplified to the minimum norm of the lattice points themselves.

Algorithms for computing the shortest vector in a general lattice have been developed in [Die75, FP85, Kn81] and efficient implementations exist even for higher dimensions [LC97]. Agrell et al. review several closest-point search methods, being closely related to finding the length of the shortest vector, for lattices without a regular structure [AEVZ02]. Specializing the setting to rank-1 lattices in two dimensions allows one to take simpler approaches.
Exhaustive Search

Given a lattice \( L_{n,g} \), the simplest strategy to compute the length of the shortest vector consists in stepping through all lattice points and comparing their distance to the origin to the length of the shortest vector found so far [Ski05]. In doing so, the periodicity of the lattice has to be taken into account, i.e. all computations have to be performed on the unit torus. Thus for the length of the vector \( x \cdot n \) we have

\[
|x \cdot n| = \begin{cases} 
|\{(g_x \cdot i)_n, (g_y \cdot i)_n\}|^2 & \text{if } \{g_x \cdot i\}_n \leq n - \{g_x \cdot i\}_n \text{ and } \{g_y \cdot i\}_n \leq n - \{g_y \cdot i\}_n \\
|\{(g_x \cdot i)_n, (g_y \cdot i)_n - n\}|^2 & \text{if } \{g_x \cdot i\}_n \leq n - \{g_x \cdot i\}_n \text{ and } \{g_y \cdot i\}_n > n - \{g_y \cdot i\}_n \\
|\{(g_x \cdot i)_n - n, (g_y \cdot i)_n\}|^2 & \text{if } \{g_x \cdot i\}_n > n - \{g_x \cdot i\}_n \text{ and } \{g_y \cdot i\}_n \leq n - \{g_y \cdot i\}_n \\
|\{(g_x \cdot i)_n - n, (g_y \cdot i)_n - n\}|^2 & \text{if } \{g_x \cdot i\}_n > n - \{g_x \cdot i\}_n \text{ and } \{g_y \cdot i\}_n > n - \{g_y \cdot i\}_n
\end{cases}
\tag{2.7}
\]

This algorithm (see Listing 2.2) has a complexity of \( \Theta(n) \) which may considerably slow down the search for rank-1 lattices for large \( n \), as discussed in Chapter 3. But it has the advantage that additional information, such as the index \( i \) of the shortest vector, can be gained “for free” by simply using an additional variable.

```python
def ComputeMinimumDistance(n, gx, gy):
    mindist = n*n
    tmpX = gx
    tmpY = gy
    for j in range(1, n):
        x = min(tmpX, n-tmpX)
        y = min(tmpY, n-tmpY)
        mindist = min(x*x + y*y, mindist)
    return mindist
```

Listing 2.2: Exhaustive search for computing the minimum distance for \( s = 2 \).

Taking into account the largest possible minimum distance \( l \) (see equation (3.3) in Section 3.2) for a rank-1 lattice \( L_{n,g} \) the complexity of Listing 2.2 can be reduced to \( \Theta(\sqrt{n}) \) [Ski05] in the case of Korobov lattices, where the \( x \)-coordinate of \( x \cdot n \) corresponds to the index of the lattice point (see Section 2.1.5). We do not have to loop over all lattice points in order to get the length of the shortest vector, but restrain to the points lying within the circle of radius \( l \cdot n \) around the origin. This means that it suffices to run the for-loop for \( 1 \leq i \leq l \cdot n \).

Spectral Test

As already stated in [Ski05], a very efficient way to search for MMD lattices (see Section 3.2) in Korobov form for \( s = 2 \) is given by the spectral test [Knu81], which measures the quality of linear congruential pseudo random number generators (see Section 1.1), i.e. the quality of \( s \)-dimensional lattices, by determining the \( s \)-dimensional accuracy \( v_s \). According to [Knu81] all good generators pass this test, and additionally it is failed by all LCGs known to be bad. The \( s \)-dimensional accuracy \( v_s \) describes the reciprocal of the maximum distance between hyperplanes, taken over all families of parallel \( (s-1) \)-dimensional hyperplanes covering all points (see Section 2.1 about the geometric interpretation of the dual lattice).
It can be shown [Knu81] that $\nu_s$ corresponds to the length of the shortest vector in the dual lattice

$$L_n^\perp := \{ h \in \mathbb{Z}^s | h \cdot g \equiv 0 \pmod{n} \},$$

for $g = (1, a, a^2, \ldots, a^{s-1})$. Since for $s = 2$ the length of the shortest vector in $L^\perp$ equals the length of the shortest vector in $L$ multiplied by $n$, the spectral test delivers the minimum mutual distance between two lattice points for one $a \in \{1, \ldots, n-1\}$ in $[0, n)^2$ by taking advantage of the self duality of the 2-dimensional rank-1 lattice. This means that the dual lattice is obtained by rotating the primal lattice by 90$^\circ$ degrees and vice versa. As the searching algorithm is performed on $[0, n)^2$, the two-dimensional accuracy $\nu_2$ delivers the sought quantity, i.e. the length of the shortest vector in the lattice $L_{n,a}$.

The following algorithm [Knu81] represents an implementation of the two-dimensional spectral test, expanding it by the additional if-statement (*) in order to deal with lattices not satisfying the condition $\gcd(n,a) = 1$ (i.e no full period). Listing 2.3 represents the steps S1 to S3 in the algorithm for the $s$-dimensional spectral test described in [Knu81]. These three steps compute the two-dimensional accuracy $\nu_2$ by a special approach which resembles the Euclidean algorithm. Writing the two-dimensional spectral test in a recursive form, for the variable $h$ by $h_{i+2} = h_i - qh_{i+1} = h_i \mod h_{i+1}$, enables to prove the number of steps in a similar way as in the Euclidean algorithm, which thus results as $O(\log n)$. The correctness of Listing 2.3 is proved in [Knu81, p. 578, exercise 5].

Since the principle of self-duality does not apply for $s > 2$, in this case the spectral test cannot be used to compute the minimum distance in a Korobov rank-1 lattice anymore and other algorithms have to be employed instead.

```python
def SpectralTest(n, a):
    # S1: Initialize
    h = a
    h = n
    p = 1
    p = 0;
    s = 1 + a+a

    # S2: Euclidean Step
    q = h/h
    u = h - q*h
    v = p - q*p
    while ((u*u + v*v) < s):
        s = u*u + v*v
        h = h
        h = u
        p = p
        p = v
        if (h == 0):  # (*)
            return s
    q = h/h;
    u = h - q*h
    v = p - q*p

    # S3: Compute \nu_2
    u = u - h
    v = v - p
```
if \((u^*u + v^*v) < s\):
\[ s = u^*u + v^*v \]

return \(s\)

Listing 2.3: Two-dimensional spectral test. \(n\) represents the number of lattice points and \(a\) the parameter in the generator vector of a Korobov lattice.

Gaussian Basis Reduction

Given a lattice basis \(B\), with \(\{b_1, \ldots, b_s\}\), \(||b_1|| \leq ||b_2|| \leq \ldots \leq ||b_s||\) being the \(s\) shortest vectors in \(L\), the minimum distance would simply correspond to the length of the first basis vector. Clearly, the \(s\) shortest vectors in a lattice span the vector space spanned by \(L\). However, the \(s\) shortest vectors do not necessarily build a basis of \(L\). On the other hand, a lattice basis is not imperative to contain the \(s\) shortest vectors in a lattice.

So another solution to finding the shortest vector in a a two-dimensional rank-1 lattice, and thus the minimum distance, consists in computing a reduced basis. For this purpose we use the Gaussian basis reduction, as described at the beginning of this section, by modifying Listing 2.1 to additionally return the length of first basis vector. Such the minimum distance is computed in a number of \(O(\log(n))\) algorithmic steps.

2.1.4 Sampling Efficiency of Rank-1 Lattices

The sampling efficiency [PM62] measures how well the sampling points of a given lattice capture the isotropic spectrum of a band-limited function. A sampling efficiency of 100% would allow to represent the band-limited function perfectly. For rank-1 lattices the sampling efficiency corresponds to the fraction

\[ \eta := \frac{R}{P} \]

of the volume \(R\) of the in-circle of the fundamental Voronoi cell and the volume \(P\) of the fundamental Voronoi cell itself in the dual lattice.

2.1.5 Locating Lattice Points

In order to store data using the geometry of rank-lattices, as for example in the case of rank-1 lattice textures (see Chapter 6), we must be able to access the underlying data structure, i.e. to get the address of individual lattice points. Contrary to the square lattice the points of a rank-1 lattice cannot easily be addressed by 2-dimensional integer Cartesian coordinates, since they are not aligned in two orthogonal directions. An alternative choice for the coordinate axes would be the axes induced by the basis vectors of the lattice, yielding integer coordinates as well. However, this does not result in a straightforward addressing scheme. In the following we describe an approach, in which the points of a rank-1 lattice are accessed by only one integer coordinate.
Exploiting the property that every lattice point \( x_i \) can be generated by a single generator vector \( g \) allows for a very simple addressing scheme. As \( x_i = \{ i \cdot g \}_1 \) is uniquely defined by \( i \), this index can be used for addressing purposes. This way data on a rank-1 lattice can be stored as a linear array \( data \) of size \( n \). Thus the value at the \( i \)-th lattice point is stored at position \( data[i] \) and conversely the value for the \( i \)-th point can be accessed by the index \( i \).

In the case of Korobov lattices the index \( i \) can be retrieved given the integer coordinates
\[
{x_i \cdot n = \left\{ \frac{i}{n} \cdot g \right\}_1 \cdot n = \{ i \cdot (1, a) \}_n = \{ (i, a \cdot i) \}_n = (i, \{ a \cdot i \}_n)
\]
of a lattice point [Ski05]. The first component of \( x_i \cdot n \in [0, n)^2 \) inherently corresponds to the point index. For general rank-1 lattices the \( x \)-coordinate of a lattice point may not be uniquely determined anymore. Therefore the index of an arbitrary lattice point \( x_i \in [0, 1)^2 \) is computed via the indices of the basis vectors of \( L_{n,g} \), which is explained in the next section.

**Addressing Scheme**

Let \( B = (b_1, b_2) \) be a Minkowski-reduced basis (see Section 2.1.3) of the lattice \( L_{n,g} \) and let \( k = index_{b_1} \) and \( l = index_{b_2} \) be the indices of the lattice points corresponding to \( b_1 \) and \( b_2 \), which can be determined using the exhaustive search approach for \( B \) by additionally storing the indices of both the shortest and second-shortest linearly independent vectors during the search. Due to the periodicity of the lattice
\[
b_{1,j} \cdot n = \begin{cases} \{ k \cdot g_j \}_n & \text{if } \{ k \cdot g_j \}_n \leq n - \{ k \cdot g_j \}_n \\ \{ k \cdot g_j \}_n - n & \text{otherwise} \end{cases} \tag{2.8}
\]

and
\[
b_{2,j} \cdot n = \begin{cases} \{ l \cdot g_j \}_n & \text{if } \{ l \cdot g_j \}_n \leq n - \{ l \cdot g_j \}_n \\ \{ l \cdot g_j \}_n - n & \text{otherwise} \end{cases} \tag{2.9}
\]
holds for the basis vectors for \( j \in \{1, 2\} \).

Given any lattice point \( x_i \in [0, 1)^2 \), the first step in computing its point index \( i \) consists in transforming \( x_i \) into the lattice basis, i.e. its coordinates \( x_{i,B} = (s, t) \) in the lattice basis have to be computed, such that \( x_i \) can be expressed as
\[
x_i = s \cdot b_1 + t \cdot b_2, \quad s, t \in \mathbb{Z}
\]
Then the index \( i \) of a lattice point \( x_i \) can be computed according to the following theorem.

**Theorem 2.** Let \( x_{i,B} = (s, t) \) be the representation of the lattice point \( x_i \) in the Minkowski reduced basis \( B \) of \( L_{n,g} \) and let \( k = index_{b_1} \) and \( l = index_{b_2} \) be the indices of the basis vectors \( b_1 \) and \( b_2 \). Then the index \( i \) of \( x_i \) results as
\[
i = \{ s \cdot k + t \cdot l \}_n
\]
Proof. We first consider the case for \( \{k \cdot g\}_n \leq n - \{l \cdot g\}_n \) and \( \{l \cdot g\}_n \leq n - \{k \cdot g\}_n \) in equations (2.8) and (2.9). Then for the basis vectors \( \mathbf{b}_1 \) and \( \mathbf{b}_2 \) we have

\[
\begin{align*}
\mathbf{b}_1 \cdot n &= x_k \cdot n = \{k \cdot g\}_n, \\
\mathbf{b}_2 \cdot n &= x_l \cdot n = \{l \cdot g\}_n.
\end{align*}
\]

(2.10)

(2.11)

Writing \( x_i \) as a linear combination of the basis vectors and inserting equations (2.10) and (2.11) for \( \mathbf{b}_1 \) and \( \mathbf{b}_2 \) yields

\[
\begin{align*}
x_i \cdot n &= (s \cdot \mathbf{b}_1 + t \cdot \mathbf{b}_2) \cdot n \\
&= \{s \cdot x_k \cdot n + t \cdot x_l \cdot n\}_n \\
&= \{s \cdot \{k \cdot g\}_n + t \cdot \{l \cdot g\}_n\}_n \\
&= \{(s \cdot k + t \cdot l) \cdot \mathbf{g}\}_n = \{(s \cdot k + t \cdot l\}_n \cdot \{\mathbf{g}\}_n\}_n,
\end{align*}
\]

where we use the rules from modular arithmetic that \( \{a + b\}_n = \{a\}_n + \{b\}_n \) and \( \{a \cdot b\}_n = \{a\}_n \cdot \{b\}_n \) [Ste01]. Finally by comparison of coefficients we have \( i = \{s \cdot k + t \cdot l\}_n \), as \( x_i \cdot n = \{i \cdot \mathbf{g}\}_n \).

Now let

\[
\begin{align*}
\mathbf{b}_1 \cdot n &= \{k \cdot g\}_n - n \\
\mathbf{b}_2 \cdot n &= \{l \cdot g\}_n - n.
\end{align*}
\]

In this case

\[
\begin{align*}
x_i \cdot n &= (s \cdot \mathbf{b}_1 + t \cdot \mathbf{b}_2) \cdot n \\
&= \{s \cdot \{k \cdot g\}_n - n + t \cdot \{l \cdot g\}_n - n\}_n \\
&= \{s \cdot \{k \cdot g\}_n - s \cdot n + t \cdot \{l \cdot g\}_n - t \cdot n\}_n \\
&= \{s \cdot \{k \cdot g\}_n + t \cdot \{l \cdot g\}_n\}_n - \{s \cdot n + t \cdot n\}_n \\
&= \{s \cdot \{k \cdot g\}_n + t \cdot \{l \cdot g\}_n\}_n.
\end{align*}
\]

(2.13)

(2.14)

As \(-s \cdot n \mod n = 0\) and \(-t \cdot n \mod n = 0\) equation (2.13) results in equation (2.14) which already has been covered in equation (2.12).

The remaining two cases for

\[
\begin{align*}
\mathbf{b}_1 \cdot n &= \{k \cdot g\}_n \\
\mathbf{b}_2 \cdot n &= \{l \cdot g\}_n - n
\end{align*}
\]

and

\[
\begin{align*}
\mathbf{b}_1 \cdot n &= \{k \cdot g\}_n - n \\
\mathbf{b}_2 \cdot n &= \{l \cdot g\}_n
\end{align*}
\]

can be shown similarly. \qed
Figure 2.5: Example for computing the index of an arbitrary lattice point for the lattice 
$L_{56,(4,7)}$ with $index_{b_1} = 1$ and $index_{b_2} = 16$.

Hence similarly to Korobov lattices, it is possible to store general rank-1 lattices as a linear array, using an additional basis transformation along with 2 multiplications and one addition. However, a disadvantage of the presented addressing scheme is that neighboring rank-1 lattice pixels are not necessary neighbors in the image array on disk.

Figure 2.5 illustrates this addressing scheme for the lattice $L_{56,(4,7)}$ and $k = 1, l = 16$. For example consider the lattice point $x \cdot 56 = (4, 21) = 3 \cdot b_1 \cdot 56 + (-1) \cdot b_2 \cdot 56$. Therefore $i = \{3 \cdot 1 + (-1) \cdot 16\} = \{-13\}_{56} = 43$. Similar for the point $x \cdot 56 = (52, 35)$ we have $x \cdot 56 = 5 \cdot b_1 \cdot 56 + 4 \cdot b_2 \cdot 56$, yielding $i = \{5 \cdot 1 + 4 \cdot 16\} = \{69\}_{56} = 13$. $x_{43}$ and $x_{13}$ are highlighted in Figure 2.5.

**Arithmetic**

The two main arithmetic operations necessary for address computations in texture and image processing are addition and subtraction modulo $n$. Index calculations can be performed in the same manner as for determining the index of a lattice point as described above. For example the index $i$ of the lattice point $x_i$ resulting from summing the position vectors of two points $x_{j_1}$ and $x_{j_2}$ is simply $i = \{j_1 + j_2\}_n$. Adding $x_{1} \cdot 56 = (4, 7)$ and $x_{16} \cdot 56 = (8, 0)$ in Figure 2.5 results in the vector $x_{1+16} \cdot 56 = x_{17} \cdot 56 = (4+8)_{56}, (7+0)_{56} = (12, 7)$. Subtraction works in the same way. Generally, the following corollary applies.

**Corollary 1.** Let $i_0$ be the index of $x_{i_0} \in [0,1]^2$ and let $x_{i_0,B} = (s,t)$ denote its representation in lattice coordinates. Then the index of the vector $x_i = (s + \lambda) \cdot b_1 + (t + \gamma) \cdot b_2$ with lattice basis representation $x_{i,B} = (s + \lambda, t + \gamma)$ is

$$i = \{i_0 + \lambda \cdot k + \gamma \cdot l\}_n.$$
This means that the indices of the neighbors of a lattice point \( \mathbf{x}_i \) simply can be found by adding and subtracting the indices of the basis vectors with respect to the index \( i \).

### 2.1.6 Shifted Rank-1 Lattices

A shifted rank-1 lattice \( L_{n, \mathbf{g}}^\Delta \) simply arises from a rank-1 lattice \( L_{n, \mathbf{g}} \) by adding an \( s \)-dimensional shift vector \( \Delta \in \mathbb{R}^s \) modulo 1:

\[
L_{n, \mathbf{g}}^\Delta := \left\{ \mathbf{x}_i := \frac{i}{n} \mathbf{g} + \Delta \mid i = 0, \ldots, n - 1 \right\}.
\]  

(2.15)

In Sections 2.2, 3.2.2 and 3.5 shifted lattices will be considered with respect to rank-1 lattice sequences and \((t, m, s)\)-nets. An error bound for shifted lattices is given in Section 2.3.3 with respect to random shifts.

### 2.2 Rank-1 Lattice Sequences

In order to generate rank-1 lattices the number of lattice points has to be determined in advance. This means that it is not possible to add points to \( L_{n, \mathbf{g}} \), since the lattice structure would be destroyed otherwise. In [HH97, HHLL01] the authors present a solution to this problem by describing the construction of an infinite sequence of points, with \( l \cdot b^m \) to \( (l + 1)b^m - 1 \) points forming a lattice for integers \( b \geq 2 \), \( m \geq 0 \) and \( l > 0 \). Thus a lattice \( L_{n, \mathbf{g}} \) can be extended without discarding the previous points.

By definition computing the \( i \)-th point of a rank-1 lattice \( \mathbf{x}_i = \left\{ \frac{i}{n} \mathbf{g} \right\}_1 \) involves the number of points \( n \). Hence, the formula for rank-1 lattices has to be rewritten such that this dependence on \( n \) is resolved. Let \( n = b^m \). Consider the Van der Corput sequence \( \Phi_b \) which mirrors the \( b \)-ary representation of an integer \( i \) about the decimal point:

\[
\Phi_b(i) : \mathbb{N}_0 \longrightarrow \mathbb{Q} \cap [0, 1)
\]

\[
i = \sum_{j=0}^\infty a_j(i) b^j \quad \rightarrow \quad \sum_{j=0}^\infty a_j(i) b^{-j-1},
\]  

(2.16)

where \( a_j(i) \) denotes the \( j \)-th digit of the integer \( i \) represented in base \( b \).

The first \( n \) values of (2.16), being sorted in ascending order, correspond to the sequence of fractions

\[
0, \frac{1}{b^m}, \frac{1}{b^{m-1}}, \frac{b^m - 1}{b^m}
\]

Therefore the term \( \frac{i}{n} = \frac{i}{b^m}, i = \{0, \ldots, n - 1\} \) in (2.15) can be detached from \( n \) by writing \( \Phi_b(i) \) instead.

Usually, the \( s \)-dimensional generator vector \( \mathbf{g} \) is also related to \( n \). Its \( b \)-ary representation is given by

\[
\mathbf{g} = (g_1(m-1) \ldots g_{10},
\quad g_2(m-1) \ldots g_{20},
\quad \ldots,
\quad g_s(m-1) \ldots g_{s0}) \quad (\text{base } b),
\]
with \( g_{jk} \in \{0, \ldots, b-1\} \) being digits. The index \( j \in \{1, \ldots, s\} \) corresponds to the dimension and the index \( k \in \{0, \ldots, m-1\} \) denotes the digit index. Since for \( k > m \) the digits have no influence on the point set of a rank-1 lattice with \( b^m \) points due to the modulo \( b^m \) operation, the generator vector can be expressed as an infinite string of digits

\[
g = (\ldots g_{11}g_{10}, \ldots g_{21}g_{20}, \ldots, \ldots g_{s1}g_{s0}) \text{ (base } b). \tag{2.17}
\]

This results in a generator vector applicable for all \( m > 0 \). Removing the upper limit on \( i \) finally yields the definition of an infinite rank-1 lattice sequence [HH97].

**Definition 1.** An infinite rank-1 lattice sequence in base \( b \) with generating vector \( g \) of the form (2.17) and shift \( \Delta \) is defined as:

\[
\{ \{ \Phi_b(i)g + \Delta \}_1 | i = 0, 1, 2, \ldots \} \tag{2.18}
\]

Taking the first \( b^m \) terms of (2.18) generates a rank-1 lattice of the form (2.15), which we call the initial rank-1 lattice for this \( m > 0 \) in Section 3.2.2. Moreover, each set of \( l \cdot b^m \) to \( (l+1)b^m - 1 \), \( l > 0 \), points builds a shifted copy of the initial rank-1 lattice. This is expressed in the following Theorem [HH97].

**Theorem 3.** Suppose that \( P \) is the set consisting of the \( l + 1 \)th run of \( b^m \) terms of the infinite rank-1 lattice sequence defined in equation (2.18):

\[
P = \{ \{ \Phi_b(lb^m + i)g + \Delta \}_1 | i = 0, \ldots, b^m - 1 \}, l = 0, 1, \ldots
\]

Then, \( P \) is a rank-1 lattice with shift \( \Phi_b(l)b^{-m}g + \Delta \), that is,

\[
P = \{ \{ \Phi_b(i)g + \Phi_b(l)b^{-m}g + \Delta \}_1 | i = 0, \ldots, b^m - 1 \}
\]

**Construction**

The search for generator vectors for infinite lattice sequences usually involves the minimization of a loss function [HH97] relating to some measure of discrepancy, as for example the spectral test. According to equation (2.17) an infinite lattice sequence requires an \( s \times \infty \) array of digits \( g_{jk} \). Practically, the maximum number of available digits \( m_{\text{max}} \) is limited, though. Thus it is sufficient to use a generator vector consisting of \( s_{\text{max}} \times m_{\text{max}} \) digits \( h_{jk} \), with \( n_{\text{max}} = b^{m_{\text{max}}} \) and \( s_{\text{max}} \) corresponding to the maximum dimension. In [HH97, HHLL01] generator vectors in Korobov form \( g = (1, a, a^2, \ldots, a^{s-1}) \) are determined by minimizing the following loss function

\[
G(a) = \max_{m,s} \tilde{G}(a, m, s),
\]

where \( \tilde{G}(a, m, s) \) relies on some measure of discrepancy. A good generator vector for a range of numbers and dimensions is achieved by maximizing \( \tilde{G}(a, m, s) \) over a range of \( s \) and \( m \). See [HH97, HHLL01] for details.
2.3 Quasi-Monte Carlo Error Bounds

The functions in computer graphics are square integrable due to finite energy and bounded, i.e. \( f(x) \in L_2^b \). However, they are only piecewise continuous, and the discontinuities are difficult to identify. Note that the function space \( L_2^b \) does not contain Dirac’s \( \delta \) distribution which appears in computer graphics in the context of modelling specular reflections for example. Often the structure of the high-dimensional integrals in image synthesis comprises several 2 – 3 dimensional integral operators, as it is the case for sampling the pixel area, depth of field (and additional motion blur), scattering events, etc. Consequently the sampling points can be padded using low-dimensional stratified patterns in a very efficient way, as Kollig and Keller have shown in [KK02].

Quasi-Monte Carlo integro-approximation has already found successful applications in computer graphics, as seen for example in [Kel01, Kel03, WK08]. In the following we give a survey of classical error bounds in the quasi-Monte Carlo setting being related to computer graphics and lattices. Then we present a new error bound for rank-1 lattices and Lipschitz continuous, periodic functions.

2.3.1 Classic Quasi-Monte Carlo Error Bounds

Let \( P_n = \{x_0, \ldots, x_{n-1}\} \subset I^s \) be a deterministic point set in order to integrate a given integrand \( f \) using quasi-Monte Carlo, as depicted in equation (1.6).

Contrary to the Monte Carlo method, which features a probabilistic error bound, deterministic error bounds for integrating (1.6) can be proved in the quasi-Monte Carlo setting. See [Nie92, Nie03, Lem09] for a comprehensive overview and more details on this topic.

The Koksma-Hlawka Inequality

The Koksma-Hlawka inequality represents the classical error bound for integrating (1.6). This error bound involves the notion of star discrepancy

\[
D^*(P_n) := \sup_{A: \prod_{j=1}^s [0, a_j] \subseteq I^s} \left| \int_{P_n} \chi_A(x) dx - \frac{1}{n} \sum_{i=0}^{n-1} \chi_A(x_i) \right| =: \lambda_s(A),
\]

which measures the uniform distribution of a given point set \( P_n \) with respect to the family of all subintervals \( A \) of \( I^s \), with \( A := \prod_{j=1}^s [0, a_j] \). \( \chi_A(x_i) \) represents the characteristic function of the set \( A \) and \( \lambda_s \) denotes \( s \)-dimensional Lebesgue measure. Thus \( \sum_{i=0}^{n-1} \chi_A(x_i) \) counts the number of sampling points \( x_i \in A, 0 \leq i < n \) and \( D^*(P_n) \) can be interpreted as the worst case integration error when integrating the axis aligned bounded boxes \( A \) anchored at the origin. If for \( n \to \infty \), \( D^* \) becomes zero, the point set is said to be uniform.

The Koksma-Hlawka inequality bounds the integration error by the product of the star discrepancy \( D^*(P_n) \) of the sampling points and the variation of the integrand \( V(f) \) in the...
sense of Hardy and Krause [Nie92]:

\[
\left| \int_P f(x) dx - \frac{1}{n} \sum_{i=0}^{n-1} f(x_i) \right| \leq V(f)D'(\mathcal{P}_n).
\]

This means that the error is split into a property of the sampling pattern and the integrand.

Based on axis aligned boxes, the error bound represents an anisotropic measure and therefore, rotating the point set influences the discrepancy of the sample points. Moreover, the variation in the sense of Hardy and Krause already becomes infinite in the case of non-axis aligned discontinuities [Kel06], thus being inapplicable to functions in computer graphics.

**Lattice Rule Error**

The error of a lattice rule \( Q f = \frac{1}{n} \sum_{i=0}^{n-1} f(x_i) \) can be formulated in terms of the Fourier coefficients of the integrand \( f \) requiring \( f \) to be periodic and to belong to the function class whose Fourier coefficients decay sufficiently fast with increasing frequency [SJ94].

Let

\[
f(x) = \sum_{h \in \mathbb{Z}^n} \hat{f}(h) e^{2\pi \sqrt{-1} h \cdot x}, \quad \text{for } x \in \mathbb{R}^n,
\]

be the absolutely convergent Fourier series expansion of \( f \), where

\[
\hat{f}(h) = \int_P f(x) e^{-2\pi \sqrt{-1} h \cdot x} dx
\]

represents the Fourier coefficient of \( f \) evaluated at \( h \). Inserting the Fourier series expansion into the lattice rule \( Q f \) yields

\[
Q f = \frac{1}{n} \sum_{i=0}^{n-1} \hat{f}(h_i) e^{2\pi \sqrt{-1} h_i \cdot x_i} = \frac{1}{n} \sum_{h \in \mathbb{Z}^n} \hat{f}(h) \sum_{i=0}^{n-1} e^{2\pi \sqrt{-1} h \cdot x_i},
\]

as \( |\sum_{h \in \mathbb{Z}^n} |\hat{f}(h)| | < \infty \). Using the fact that \( z := h \cdot x_i \in \mathbb{Z} \), if \( h \in L^\perp \) for \( x_i \in L \), \( 0 \leq i < n \) and that \( \frac{h_i}{n} := h \cdot x_i \in \mathbb{Q} \), if \( h \notin L^\perp \) for some integer \( k \), we have

\[
\sum_{i=0}^{n-1} e^{2\pi \sqrt{-1} z} = \begin{cases} \sum_{i=0}^{n-1} e^{2\pi \sqrt{-1} z} = 0 & \text{if } h \notin L^\perp \\ \sum_{i=0}^{n-1} e^{2\pi \sqrt{-1} z} = \sum_{i=0}^{n-1} \cos(2\pi z) + \sqrt{-1} \sin(2\pi z) = n & \text{if } h \in L^\perp \end{cases}
\]

For the proof of the case \( h \notin L^\perp \) see [SJ94]. Therefore the integration error results as

\[
\left| Q f - \int_P f(x) dx \right| = \left| \frac{1}{n} \hat{f}(0) \cdot n + \frac{1}{n} \sum_{0 \neq h \in L^\perp} \hat{f}(h) \cdot n - \hat{f}(0) \right| = \sum_{0 \neq h \in L^\perp} \hat{f}(h).
\]

Obviously the error depends only on the Fourier coefficients of \( f(x) \), evaluated at the points of the dual lattice \( L^\perp \). If the Fourier coefficients decrease with increasing \( h \), i.e.

28
\[ |\hat{f}(\mathbf{h})| \leq \frac{c}{(h_1 \cdot h_2 \cdots h_s)^\alpha} \] with \( c > 0, \alpha > 1 \) and \( \bar{h}_j := \max\{1, |h_j|\} \) for \( j = 1, \ldots, s \) [Kel02], the error is bounded by
\[
\left| \frac{1}{n} \sum_{i=0}^{n-1} f(x_i) - \int f(x) \, dx \right| \leq \sum_{\mathbf{0} \neq \mathbf{h} \in L^\perp} \frac{c}{(\bar{h}_1 \cdot \bar{h}_2 \cdots \bar{h}_s)^\alpha}.
\]

This bound also extends to parametric integration [Ski05]
\[
\left| \frac{1}{n} \sum_{i=0}^{n-1} f(x_i, y) - \int f(x, y) \, dx \right| \leq \sum_{\mathbf{0} \neq \mathbf{h} \in L^\perp} \sum_{y} \frac{c}{(\bar{h}_1 \cdot \bar{h}_2)^\alpha}.
\]

Since the smoothness conditions usually do not hold for the setting of computer graphics, we cannot use this error bound either.

\((\mathcal{M}, \mu)\)-Uniformity

The notion of \((\mathcal{M}, \mu)\)-Uniformity introduced by Niederreiter [Nie03] supports partitions which are not axis aligned and relies on the stratification properties of the sampling points.

**Definition 2.** Let \((X, \mathcal{B}, \mu)\) be an arbitrary probability space and let \(\mathcal{M}\) be a nonempty subset of \(\mathcal{B}\). A point set \(P_n\) of \(n\) elements of \(X\) is called \((\mathcal{M}, \mu)\)-uniform if
\[
\sum_{i=0}^{n-1} \chi_M(x_i) = \mu(M) \cdot n \quad \text{for all } M \in \mathcal{M},
\]
where \(\chi_M(x_i) = 1\) if \(x_i \in M\), zero otherwise.

For example points from the cartesian product midpoint rule and radical inversion based points, as e.g. \((t, m, s)\)-nets (see Section 3.5), are \((\mathcal{M}, \mu)\)-uniform point sets [Nie03]. As the Voronoi diagram of a rank-1 lattice induces a partition of the unit cube \(I_s\) into \(n\) sets of identical shape and volume \(\frac{1}{n}\) [Kel06], rank-1 lattices also share the property of being \((\mathcal{M}, \mu)\)-uniform. This shows that for \((\mathcal{M}, \mu)\)-uniformity all \(\mu(M)\) must have the same denominator \(n\).

Using this measure of uniformity Niederreiter proves the following deterministic error bound [Nie03].

**Theorem 4.** Let \((X, \mathcal{B}, \mu)\) be an arbitrary probability space and let \(\mathcal{M} = \{M_1, \ldots, M_k\}\) be a partition of \(X\) with \(M_j \in \mathcal{B}\) for \(1 \leq j \leq k\). Then for any \((\mathcal{M}, \mu)\)-uniform point set \(P = \{x_0, \ldots, x_{n-1}\}\) and any function \(f\) on \(X\) we have
\[
\left| \frac{1}{n} \sum_{i=0}^{n-1} f(x_i) - \int_X f(x) \, d\mu(x) \right| \leq \sum_{j=1}^k \mu(M_j) \left( \sup_{x \in M_j} f(x) - \inf_{x \in M_j} f(x) \right),
\]
which can easily be generalized to integro-approximation [Kel06]:

\[29\]
Theorem 5. Let \((X, \mathcal{B}, \mu)\) be an arbitrary probability space and let \(\mathcal{M} = \{M_1, \ldots, M_k\}\) be a partition of \(X\) with \(M_j \in \mathcal{B}\) for \(1 \leq j \leq k\). Then for any \((\mathcal{M}, \mu)\)-uniform point set \(P = \{x_0, \ldots, x_{n-1}\}\) and any bounded function \(f\), which restricted to \(X\) is \(\mu\)-integrable, we have

\[
\left\| \frac{1}{n} \sum_{i=0}^{n-1} f(x_i, y) - \int_X f(x, y) d\mu(x) \right\| \leq \sum_{j=1}^{k} \mu(M_j) \sup_{x \in M_j} \left| f(x, y) - \inf_{x \in M_j} f(x, y) \right|.
\]

With \((X, \mathcal{B}, \mu) = ([0, 1]^s, \mathcal{B}, \lambda_s)\), where \(\mathcal{B}\) corresponds to the Borel-sets and \(\lambda_s\) to the \(s\)-dimensional Lebesgue-measure, this bound also applies in the context of computer graphics. This error bound works for any function \(f \in L^2(P)\), however, the error cannot be separated into a property of the integrand and the sampling pattern any longer.

### 2.3.2 Error Bound for Lipschitz Continuous Functions

Although the classical error bounds do not fit in the setting of computer graphics, quasi-Monte Carlo methods achieve good results in a vast number of numerical experiments. The main reason is that the integrands are often piecewise continuous, while the discontinuities cannot be captured by the classical error bounds. Thus they cannot explain the observed convergence. We now examine an error bound for Lipschitz continuous, periodic functions with respect to parametric integration thereby completing Niederreiter’s work [Nie03] for the special case of rank-1 lattices.

Given a Minkowski reduced basis of a rank-1 lattice \(L_{n,g}\) the basis vectors induce the Delaunay tessellation of the lattice and its dual, the Voronoi diagram. In order to derive the error bound we need the following

**Definition 3.** The radius \(r(n, g)\) of a rank-1 lattice is the smallest circumcircle of the fundamental Voronoi cell with respect to some suitable norm.

This quantity corresponds to the dispersion

\[
d_n(L_{n,g}; P) = \sup_{x \in P} \min_{0 \leq i < n} d(x, x_i)
\]

of a rank-1 lattice as well as the notion of the covering radius in coding theory. Figure 2.6 shows the Voronoi diagram along with the circumcircle of radius \(r(32, (1, 7))\) of the Korobov lattice \(L_{32,7}\).

Based on the results of [DFG99] and by taking advantage of the geometrical properties of rank-1 lattices the proof is very simple and resembles the proofs of the paper [Nie03].

**Theorem 6.** Let \(f\) be a Lipschitz continuous function periodic on \([0, 1]^{s+s'}\), with

\[
\|f(x_1, y) - f(x_2, y)\| \leq L\|x_1 - x_2\|
\]

and Lipschitz constant \(L\) independent of \(x_1, x_2\) and \(y\), where \(\dim x_1 = \dim x_2 = s\) and \(\dim y = s'\). Further let \(P_n = \{x_0, \ldots, x_{n-1}\}\) be a rank-1 lattice. Then

\[
\left\| \int_{[0,1]^{s+s'}} f(x, y) dx - \frac{1}{n} \sum_{i=0}^{n-1} f(x_i, y) \right\| \leq L \cdot r(n, g)
\]
for some suitable norm, where $r(n, g)$ is the radius of $P_n$.

**Proof.** Let $\mathcal{M} = \{M_0, \ldots, M_{n-1}\}$ be the partition of $I^s$ by the Voronoi diagram of a rank-1 lattice. Then in a first step the quadrature error can be estimated similar to $[\text{DFG99}]$:

\[
\left| \int_{[0,1]^s} f(x,y)dx - \frac{1}{n} \sum_{i=0}^{n-1} f(x_i, y) \right| = \left| \sum_{i=0}^{n-1} \int_{M_i} (f(x,y) - f(x_i, y))dx \right| \\
\leq \sum_{i=0}^{n-1} \int_{M_i} \|f(x,y) - f(x_i, y)\|dx \\
\leq L \sum_{i=0}^{n-1} \int_{M_i} \|x - x_i\|dx \\
= L \cdot n \int_{M_0} \|x\|dx \\
\leq L \cdot n \cdot \lambda_s(M_0) \sup_{\|x\| \leq r(n, g)} \|x\| \\
= L \cdot n \cdot \frac{1}{n} \cdot r(n, g) = L \cdot r(n, g). \tag{2.20}
\]

Since the $M_i$ are of identical shape and volume and due to the point symmetry of the lattice, we can choose $x_i$ as $x_0 = 0$ in equation (2.19) which then can further be simplified resulting in equation (2.20).

Obviously the error bound results as a product of a property of the integrand and a property of the sampling pattern again. Omitting the parameter $y$ in Theorem 6 yields the integration error bound.

Let $\Omega \subset \mathbb{R}^s$, $\mathcal{M} = \{M_0, \ldots, M_{n-1}\}$ be an arbitrary tessellation of $\Omega$, and $\{A_i\}_{i=0}^{n-1}$ the volumes of $\{M_i\}_{i=0}^{n-1}$. Then equation (2.19) represents the special case of the error estimation.
of [DFG99]

\[
\int_{\Omega} f(x)dx - A_i \sum_{i=0}^{n-1} f(x_i) \leq L \sum_{i=0}^{n-1} \int_{M_i} \|x - x_i\|dx \tag{2.21}
\]

for rank-1 lattices. It can be proved that this error bound is minimized by choosing \(\{x_i\}_{i=0}^{n-1}\) and \(\{M_i\}_{i=0}^{n-1}\) such that the \(\{M_i\}_{i=0}^{n-1}\) are the Voronoi sets for the \(x_i\) and the \(x_i\) are the mass centroids of the Voronoi sets at the same time [DFG99]. This means that rank-1 lattices are a suitable choice to minimize the integration error for Lipschitz continuous functions, since these conditions apply to these point sets due to their geometrical properties.

The theoretical rate of the new bound \(O(n^{-1/s})\) is already known from the field of information based complexity and approximation theory. It obviously is cursed by the dimension, which is hidden in the radius \(r(n, g)\). However, the important issue about this theorem is not the rate as we consider \(s = 2\), but that it yields a shift invariant and isotropic measure of uniformity of a point set \(P_n\) with respect to the mutual minimum distance on the unit torus and thus a criterion for lattice search using the primal instead of the dual lattice by means of the following corollary:

**Corollary 2.** Maximizing the minimum distance

\[
d_{\text{min}}(L_{n, g}) := \min_{0 < i < n} \|x_i\|
\]

in a rank-1 lattice decreases the radius \(r(n, g)\) and thus the integration error.

Note that \(\|\ldots\|\), denotes the Euclidean distance on the unit torus.

This corollary can be derived by the following observation. The minimum distance \(d_{\text{min}}(L_{n, g})\) of a rank-1 lattice corresponds to two times the radius of the in-circle of the fundamental Voronoi cell. Maximizing the minimum distance in a rank-1 lattice thus increases this radius. The Voronoi cells, being of equal size and constant volume, approximate a sphere the more, the bigger the minimum distance becomes. Consequently the gap between the radius of the circumcircle and the in-circle of the Voronoi cells decreases. This means that \(r(n, g)\) decreases as \(d_{\text{min}}(L_{n, g})\) increases, which is stated in the corollary. Although there are similarities to sphere packings, it is important to note that this argument is not built upon them.

In [Kel06] Keller observed that, for Hammersley points, Sobol sequence and Larcher-Pillichshammer points, a smaller discrepancy often brings about a larger minimum distance. In the same manner, the Faure permuted Halton sequence features a larger minimum distance than the original Halton sequence. Also see [GHSK08], where Grunschloss et al. examined constructions for \((t, m, s)\)-nets with maximized minimum distance.

### 2.3.3 Randomized Quasi-Monte Carlo Error Bound

Contrary to experimental results the quasi-Monte Carlo error bounds are often very pessimistic and the underlying function classes do not apply to the setting of computer graphics in many cases. Combining Monte Carlo and quasi-Monte Carlo integration offers a
technique in order to be able to use the probabilistic error bound available in Monte Carlo in the context of quasi-Monte Carlo integration [Lem00, Lem09]. Thus the function class of \( L^2_b \) is covered again.

Randomized quasi-Monte Carlo works by generating \( m \in \mathbb{N} \) random replications of a deterministic point set \( P_n := \{x_0, \ldots, x_{n-1}\} \) yielding

\[
\tilde{P}_{l,n} := \{\tilde{x}_{l,0}, \ldots, \tilde{x}_{l,n-1}\}
\]

for \( 1 \leq l \leq m, l \in \mathbb{N} \) such that each point \( \tilde{x}_{l,i} \) for \( i \) fixed is uniformly distributed, i.e. \( \tilde{x}_{l,i} \sim U([0,1]^s) \). This makes sure that the corresponding estimator is unbiased. Moreover the randomization must not destroy the uniform distribution of the original point set \( P_n \), i.e. \( \tilde{P}_{l,n} \) shall maintain the low-discrepancy property of \( P_n \).

A very simple randomization method [CP76] which was originally devised for lattice point sets consists in randomly shifting the deterministic point set

\[
x_{l,i} = x_i + \xi_l \mod 1
\]

for \( i = 1, \ldots, n \), where \( \xi_l \) is a uniform \( s \)-dimensional random vector \( \xi_l \sim U([0,1]^s) \) and modulo operation is performed for each coordinate.

As this so-called Cranley-Patterson rotation randomly shifts a point set \( P_n \) on the unit torus the minimum distance of \( P_n \) is maintained, whereas the discrepancy may change. Thus the geometric properties of rank-1 lattices are preserved, which can be considered as shifted lattices \( L^\Delta_n \) with \( \Delta = \xi_l \). An overview of other randomization techniques like scrambling and permutations aimed for \((t,m,s)\)-nets and \((t,s)\)-sequences for example can be found in [Lem09].

The unbiased Monte Carlo estimator for computing the integral (1.6) results as

\[
I_{m,n} := \frac{1}{m} \sum_{l=1}^{m} \frac{1}{n} \sum_{i=0}^{n-1} f(\tilde{x}_{l,i}),
\]

where the sets \( \{\tilde{x}_{1,i}, i = 0, \ldots, n-1\}, \ldots, \{\tilde{x}_{m,i}, i = 0, \ldots, n-1\} \) are \( m \) i.i.d. copies of \( P_n \).

The variance then can be estimated by

\[
\sigma^2(I_{m,n}) = \frac{1}{m(m-1)} \sum_{l=1}^{m} \left( \frac{1}{n} \sum_{i=0}^{n-1} f(\tilde{x}_{l,i}) - I_{m,n} \right)^2.
\]

(2.23)

Note that (2.23) yields a variance estimation for the integration by randomly shifted rank-1 lattices, i.e. \( \Delta = \xi_l \), as defined in Section 2.1.6.
3 Parameters for Rank-1 Lattices

The generator vector \( g \) of a rank-1 lattice can be chosen such that the resulting point set is of low discrepancy \([SJ94]\). Then the elements of this point set are called good lattice points. Although the existence of good lattice points can be proved \([Nie92]\), there are only very few algorithms to explicitly construct such lattices. Therefore usually extensive computer searches are necessary in order to determine the lattice parameters which are stored in lattice tables. In \([Nie92]\) Niederreiter gives a survey of references for such tables, which for example can be found in \([HW81, Hab83, Ent98]\). We also summarize the results of our exhaustive computer searches in tables found in the Appendix B.

3.1 Constructions

So far, there exists only a small number of explicit constructions for good lattice points by taking advantage of a relation of good lattices and continued fractions \([Nie92, WH01]\). In this context Niederreiter and Borosh \([BN83, Nie86]\) showed that good two-dimensional lattice points can be explicitly constructed for \( n \) being a power of two. Also the well known Fibonacci lattices \([Nie92, Kel04]\) are derived from this principle. Moreover, a construction by Cools \([CR97]\) builds upon the continued fraction equal to \( \sqrt{3} \). The latter two constructions are described in more detail in the following.

3.1.1 Fibonacci Lattice

Originally, the Fibonacci lattice was introduced as an instance of a two-dimensional Korobov lattice. Based on the Fibonacci sequence \( F_k := F_{k-1} + F_{k-2} \) with \( F_2 := F_1 := 1 \), the number of points is set to \( n = F_k, k \geq 2 \) and the generator vector is defined as \( g = (1, F_{k-1}) \). Figure 3.1 shows a Fibonacci lattice in the unit square with \( n = F_9 = 34 \) points and the generator vector \( g = (1, F_8) \).

Hua and Wang \([HW81, WH01]\) proposed a generalization of the Fibonacci lattice for dimensions \( s > 2 \) by using the generalized Fibonacci sequence:

\[
F_0^{(s)} = F_1^{(s)} = \ldots = F_{s-2}^{(s)} = 0, \quad F_{s-1}^{(s)} = 1,
\]

\[
F_k^{(s)} = F_k^{(s)} + \ldots + F_{k+s-1}^{(s)}, \quad k = 0, 1, \ldots.
\]

Based on this sequence they chose \( n_k = F_k^{(s)} \) and \( g = (1, F_k^{(s)} , \ldots , F_{k+s-1}^{(s)}) \).
The CR97 Cools and Reztsov construct a family of lattice rules by considering the continued fraction equal to $\sqrt{3}$.

In [CR97] Cools and Reztsov construct a family of lattice rules by considering the continued fraction equal to $\sqrt{3}$

$$1 + \frac{1}{1 + \frac{1}{2 + \frac{1}{1 + \frac{1}{2 + \cdots}}}}.$$

They use the sequence of convergents

$$\left\{ \frac{F_m}{M_m} \right\}_{m=1}^{\infty} = \left\{ \frac{1}{1}, \frac{2}{3}, \frac{5}{11}, \frac{7}{16}, \frac{19}{41}, \frac{26}{57}, \frac{256}{553}, \cdots \right\}$$

of this continued fraction to define the lattice rule

$$Q_{m,f} = \frac{1}{2F_mM_m} \sum_{j=0}^{2F_mM_m-1} f\left( \frac{j}{g} \right), \quad g = \left( \frac{1}{2F_m}, \frac{1}{2M_m} \right). \quad (3.1)$$

A convergent equals a rational approximation to the irrational number which is represented by the continued fraction, with the odd-numbered ones being larger and the even-numbered ones being smaller than this number.

This corresponds to the rank-1 lattice with integer generator vector $g$

$$L_{n,g} = \left\{ \frac{i}{2F_mM_m} (M_m, F_m) \right\}_{i=0}^{2F_mM_m-1}.$$

(3.2)

Since these lattices are constructed to exactly integrate trigonometric polynomials of a hexagonal spectrum, they obtain the maximum minimum distance between the lattice points, i.e. they actually represent maximized minimum distance lattices, which are defined in the following section. Examples for lattices belonging to this construction scheme are shown in Figure 3.2.
3.2 Maximized Minimum Distance Rank-1 Lattices

In computer graphics sampling patterns with blue noise spectral properties are used in analogy to the principle of maximized minimum distance (MMD) apparent in nature. For example the photo receptors in the retina are distributed according to this scheme [Yel83]. When using blue noise sampling patterns, aliasing artifacts are traded for noise, which is less disturbing for the human eye. Such sampling points can be generated for example by means of the Poisson disc random process [Coo86, Mit91, MF92], or by performing a Lloyd relaxation [Llo82] on an initial random point set [HDK01].

Similarly we can select rank-1 lattice generator vectors that maximize the minimum distance, which is motivated mathematically in Section 2.3.2. The notion of maximized minimum distance lattices has already been introduced in [Ski05] for the special case of Korobov lattices.

For \( s = 2 \) the sequence of rank-1 lattices with increasing minimum distance approximates the hexagonal lattice. This is illustrated in Figure 3.3 for \( n = 56 \) points, where the Voronoi diagrams of the corresponding lattices are plotted.

Figure 3.2: Examples for maximized minimum distance rank-1 lattices by Cools and Reztov.

Figure 3.3: In the sequence of rank-1 lattices \( L_{56,g} \) the minimum distance increases from left to right. For comparison the square lattice is added to the left, whereas the hexagonal lattice is plotted to the right of the image sequence.
Figure 3.4: Squared maximized minimum distance as a function of \( n \). The upper bounds are given by \((l \cdot n)^2\), as seen in equation (3.3), and the Minkowski upper bound. The Euclidean distance is computed for \( x_i \cdot n \in [0,n)^2 \).

As mentioned in the previous section the family of lattice rules of [CR97] actually represents MMD rank-1 lattices. However, the construction only covers lattices for \( n = 2F_mM_m \) points and for other \( n \) the generator vector usually has to be determined by computer search. This means that for a fixed number of points \( n \), we have to identify the generator vector \( g \in \mathbb{N}^s \) such that the minimum distance \( d_{\min}(L_n, g) \) between two lattice points achieves the maximum on the unit torus. In order to exclude lattice candidates which obviously do not fulfill the demand for maximized minimum distance, i.e. lattices \( L_n, g \) with \( x_i = x_j, i \neq j \) \((d_{\min}(L_n, g) = 0)\), we define a valid generator vector as follows:

**Definition 4.** A vector \( g \in \mathbb{N}^s \) is a valid generator vector for the rank-1 lattice \( L_n, g \) with \( d_{\min}(L_n, g) > 0 \) if

\[
\gcd(n, g_0, \ldots, g_{s-1}) = 1.
\]

Figure 3.4 shows the squared maximized minimum distance \( \text{MMD}(n) \) of \( L_n, g : n \in [0,n)^2 \) as a function of \( n \). Obviously the curve is very fluctuating, i.e. there exist \( n + k > n, k \in \mathbb{N} \), such that \( \text{MMD}(n+k) \leq \text{MMD}(n) \). The reason for this is that for each \( n \) an independent lattice search is performed, such that the resulting lattices have nothing in common but the search criterion. A similar observation is made in [SJ94] with respect to the integration error.

Let \( \lambda(L) \) denote the length of the shortest vector in a lattice \( L \), which is also said to be the first minimum of \( L \) [Kan87]. Then Minkowski’s first theorem states that \( \lambda(L) \) can be bounded as a function of the dimension and the determinant of the lattice \( \det(L) \):
Theorem 7.

\[ \lambda(L) \leq \sqrt{s} \cdot \det(L)^{\frac{1}{2}} \]

As we have \( \det(L_{n,g}) = \frac{1}{n} \), for two-dimensional rank-1 lattices this results in the following bound:

\[ \lambda(L) \leq \sqrt{2} \cdot \frac{1}{\sqrt{n}}. \]

The Minkowski bound \( (\lambda(L) \cdot n)^2 \leq 2 \cdot n \) is plotted in Figure 3.4, being only a loose bound to the maximized minimum distance for the rank-1 lattices.

As shown in [Ski05] a tight upper bound for the maximized minimum distance in two dimensions (see Figure 3.4) can be derived by the following considerations. The largest possible minimum distance \( l \) would result from a point set, whose triangulation consists of only equilateral triangles (analogous to hexagonal lattices) [CSB87]. Equating the area \( A = \frac{1}{n} \) of the fundamental parallelepiped of a rank-1 lattice and twice the area of such an equilateral triangle of side length \( l \) yields

\[ A = \frac{1}{n} = 2 \left( \frac{1}{2} \cdot l \cdot \sqrt{l^2 - \left( \frac{l}{2} \right)^2} \right) \iff l = \sqrt{\frac{2}{n \cdot \sqrt{3}}}. \]  

(3.3)

3.2.1 Searching on the Unit Square

The problem of constructing lattices with longest possible shortest nonzero vectors for a given lattice density \( \det(L) \) (see Section 2.1.1) is connected to the problem of finding the densest packing of spheres which has been studied for a long time [CSB87, Mar03, Sie89]. Computer searches for good lattices based on the lengths of shortest nonzero vectors have been reported in [L'E99, LL00] for example. They focus on the dual lattice, though, and use either exhaustive or random searches, the latter of which poses the problem of deciding how much time to spend on the search process. In the following we examine the search for MMD rank-1 lattices, where the search space is restricted by the structure of the generator vectors. Nevertheless, the search still represents a computationally expensive problem, since there are \((n-1)^s\) possibilities for the generator vector \( \mathbf{g} = (g_1, \ldots, g_s) \), \( g_i \in \{1, \ldots, n-1\}, i = 1, \ldots, s \). As the applications considered in the following chapters provide mainly a low-dimensional structure, we concentrate on the search for MMD rank-1 lattices in two dimensions.

Exhaustive Search

The naïve exhaustive search algorithm enumerates all possible generator vectors in order to find MMD lattices. For each generator vector candidate the minimum distance is evaluated either by the exhaustive search approach (see Listing 2.2 in Section 2.1.3) or by the faster Gaussian reduction which is used in Listing A.1. Additionally to reducing the complexity from \( \mathcal{O}(n^3) \) in the first case to \( \mathcal{O}(n^2 \log(n)) \), the Gaussian reduction allows to determine a Minkowski reduced basis for the lattice during the search process. Already for only \( s = 2 \) dimensions, scanning \( \mathcal{O}((n-1)^2) \) candidates becomes prohibitive for large \( n \) as used in
our applications. In experiments the exhaustive search has been feasible only for relatively small $n < 8500$. In this context feasible refers to “on-the-fly” search, i.e. the search should not last longer than some seconds. For example on an iMac G5 a C++ implementation of Listing A.1 took 98 seconds to compute the result for $n = 8500$. In order to be able to accelerate the exhaustive search for MMD rank-1 lattices, we take a closer look at the structure of the search space in the following.

In Figure 3.5 the minimum distance is illustrated over the whole search space, i.e. for all $g = (g_1, g_2) \in [1, n]^2$, with $n = 276$. Each possible generator vector $g$ is represented by a pixel in the image, whose color is determined by dividing the corresponding minimum distance $d_{\min}(L_{n,g})$ by the maximized minimum distance $MMD(n)$ for this $n$. This means that a black pixel denotes a minimum distance of 0, whereas a white pixel represents a generator vector with maximized minimum distance. In order to emphasize the generator vectors having a minimum distance in the range of $0.8 \cdot MMD(n) \leq d_{\min}(L_{n,g}) \leq MMD(n)$ a special color coding is used in the following cases: red denotes the maximized minimum distance, green the lattices having a minimum distance $d_{\min}(L_{n,g}) \geq 0.9 \cdot MMD(n)$, and blue determines lattices with $0.8 \cdot MMD(n) < d_{\min}(L_{n,g}) < 0.9 \cdot MMD(n)$.

As the minimum distance is computed by means of the Euclidean norm, circular structures are apparent all over the search space in Figure 3.5. Moreover Figure 3.5 shows that, for a given $n \in \mathbb{N}$, there may exist several generator vectors yielding the same minimum distance. Therefore a list is used in Listing A.1 in order to store the potential generator vectors. These symmetries are captured in the notion of geometrically equivalent lattices.

The number of distinct $s$-dimensional lattice rules for a given $n \in \mathbb{N}$ can be determined by counting the number $T_s(n)$ of dual lattices, which are expressed in Hermite normal form [SJ94]. As $T_s(n)$ is a multiplicative function, i.e. $T_s(m \cdot n) = T_s(n) \cdot T_s(m)$ if $m$ and $n$ are relatively prime, $T_s(n)$ can be computed for any $n$, if its prime powers $p^\beta$ are known. An explicit formula for $T_s(n)$ can be found in [SJ94]:

**Theorem 8.** For $p$ prime and $\beta$ a positive integer, we have

$$T_s(p^\beta) = \prod_{i=1}^{s-1} \left[ \frac{p^{\beta+i-1} - 1}{p^i - 1} \right] = \prod_{i=1}^{\beta} \left[ \frac{p^{\beta+i-1} - 1}{p^i - 1} \right]$$

Thus for $n = 8 = 2^3$ there are $T(2^3) = \frac{2^{3+1-1} - 1}{2^2 - 1} = 15$ distinct lattices. This expression contains the lattices which are geometrically equivalent but not identical. On the unit cube geometrically equivalent lattices are defined in the following way [SJ94]:

**Definition 5.** Two lattice rules are geometrically equivalent if one can be changed to the other by a relabeling of coordinates, or by replacing the coordinate $x_i$ by $1 - x_i$ (unless $x_i = 0$, in which case it is left unchanged), or by a combination of such transforms.

For example in two dimensions there are 8 possibilities for such symmetries, yielding geometrically equivalent lattices, which are listed in Table 3.1 and visualized in Figure 3.6.
Figure 3.5: Color coding the minimum distance over the search space for $n = 276$. 

- **MMD**
- $0.9 \cdot \text{MMD} \leq d_{\text{min}}(L_{n,g})$
- $0.8 \cdot \text{MMD} \leq d_{\text{min}}(L_{n,g}) < 0.9 \cdot \text{MMD}$
- $d_{\text{min}}(L_{n,g})/\text{MMD}$
Figure 3.6: Structure of the search space with respect to geometrically equivalent rank-1 lattices (see Table 3.1).

\[
\begin{array}{c|c}
(g_1, g_2) & (g_2, g_1) \\
(g_1, n - g_2) & (n - g_2, g_1) \\
(n - g_1, g_2) & (g_2, n - g_1) \\
(n - g_1, n - g_2) & (n - g_2, n - g_1)
\end{array}
\]

Table 3.1: Geometrically equivalent lattices \( L_{n,g} \) with \( g = (g_1, g_2) \) in two-dimensions (in \([0,n]^2\))

We are interested in the number of those lattices, since by definition they have the same minimum distance. However, no explicit formula is known yet [SJ94].

The number of identical lattices arises from group theory [SJ94]:

**Definition 6.** A group \( G \) of order \( N \) is said to be cyclic if there exists an element \( g \in G \) (a generator of the group) such that every element \( u \in G \) may be written as \( u = g^q \) for some nonnegative integer \( q \).

As a rank-1 lattice can be considered as a cyclic group by definition, group theory states that there are \( \phi(n) \) possibilities to choose the generator vector of \( L_{n,g} \), where \( \phi(n) \) is Euler’s totient function. Providing that \( g \) is a generator vector of \( L_{n,g} \), \( g^q := g \cdot q \mod n \) is also a generator if \( q \) and \( n \) are relatively prime. Therefore each lattice contains \( \phi(n) \) lattice points \( u \) generating \( L \), with \( \gcd(n, u_x, u_y) = 1 \).

Experimentally the exhaustive lattice search allows to determine the number of geometrically equivalent lattices \( \text{num}^{\text{geom}}_n \) by counting the number of generator vectors with the same minimum distance and dividing this number by the number of identical lattices \( \text{num}^{\text{id}}_n \).

In our experiments we observed the number of geometrically equivalent lattices \( \text{num}^{\text{geom}}_n \) to correspond to \( k \in \{2, 4, 6, 8, 10, 12, 18, 20\} \) for \( 4 \leq n \leq 1500 \). However, this number also contains lattices which are equivalent with respect to affine transformations, like shearing. So for \( n = 12 \) and a minimum distance of \( d_{\text{min}}(L_{12,g}) = 9 \), there are \( 24/4 = 6 \) different...
lattices, two of which are connected by a shear along the x-axis to the other geometrically equivalent lattices. This is illustrated in Figure 3.7.

\[ L_{12,(1,4)} \]
\[ L_{12,(1,8)} \]
\[ L_{12,(4,1)} \]
\[ L_{12,(4,5)} \]
\[ L_{12,(3,4)} \]
\[ L_{12,(4,3)} \]

Figure 3.7: For \( n = 12 \) and \( d_{\text{min}}(L_{12,g}) = 9 \) the number of generators equals \( 6 \cdot \phi(n) = 6 \cdot 4 \). The red dots denote the points within \( L_{12,g} \) generating identical lattices. \( L_{12,(1,4)} = g L_{12,(1,8)} = g L_{12,(4,1)} = g L_{12,(4,5)} \) are geometrically equivalent lattices according to Definition 5, as well as \( L_{12,(3,4)} = g L_{12,(4,3)} \). The first group of lattices is connected to the second one by means of a shear along the x-axis, e.g. \( \begin{pmatrix} 1 & 0.25 \\ 0 & 1 \end{pmatrix} L_{12,(3,4)} = L_{12,(1,4)} \).

To summarize, the number of rank-1 lattices with the same minimum distance corresponds to \( k \cdot \phi(n) \), where \( k \) denotes the number of geometrically equivalent lattices including affine transformations and \( \phi(n) \) equals the number of generator vectors generating identical lattices.

Accounting for geometrically equivalent lattices we only need to consider half the lower left quarter in the first quadrant (see the green triangle in Figure 3.6), as the search space for the exhaustive search, but this does not change the order of the algorithm.

**Searching for Rank-1 Lattices in Korobov Form**

One possibility to reduce the search space in order to speed up the search process is to restrict to rank-1 lattices in Korobov form [Ski05], which are uniquely determined by the tuple \((n,a)\). The search algorithm loops over all potential lattices \( L(n,a), a \in \{1, \ldots, n-1\} \) and determines the length of the shortest vector for each candidate either by means of the spectral test or by the Gaussian reduction. After comparing this quantity to the length of the vector with the maximum minimum distance so far, the related parameter \( a \) is possibly stored. According to the previous section this parameter is not unique, i.e. there exist several possibilities for \( a \) resulting in the same maximized minimum distance.

As for the exhaustive search the complexity of the search algorithm depends on the number of choices for the generator vector and the number of steps in order to compute the minimum distance. The minimum distance contributes a factor of \( \mathcal{O}(\log n) \) when either the two-dimensional spectral test or the Gaussian reduction is used. Since there are \( n \) possibilities to choose the parameter \( a \), the search algorithm terminates after \( \mathcal{O}(n \log n) \) steps, respectively.
Requiring Low-Dimensional Projections

Additionally, the search space can be restricted by demanding that \( n \) and \( g_i \) are relatively prime, i.e. \( \gcd(n, g_i) = 1, i \in \{1, \ldots, s\} \) [Ski05]. As a consequence the resulting lattice points are stratified equidistantly on each coordinate axis. So the resulting rank-1 lattice is an instance of a Latin hypercube sample and a lower bound for the minimum distance is \( \frac{1}{n} \).

However, the condition \( \gcd(n, g_i) = 1, i \in \{1, \ldots, s\} \) prevents to find the best lattice with regard to maximized minimum distance in some cases. This also applies to searching MMD rank-1 lattices in Korobov form. For example the maximized minimum distance of the lattices according to equation (3.2) cannot be achieved in Korobov form.

Figure 3.8 compares the MMD rank-1 lattices for \( n = 56 \) selected in Korobov form (a), for \( \gcd(n, a) = 1 \) in Korobov form (b) and by using the lattice family of [CR97] (c).

![Figures](image)

Figure 3.8: Maximized minimum distance lattices for \( n = 56 \): (a) Rank-1 lattice searched under the restriction of \( \gcd(n, a) = 1 \) in Korobov form. (b) Rank-1 lattice in Korobov form. (c) Rank-1 lattice selected without restrictions.

Optimizing the Exhaustive Search

Table B.1 lists some further \( n \in \{3, \ldots, 2028\} \) for which the best lattice concerning maximized minimum distance is not available in Korobov form, including the lattices of equation (3.2). However, for \( n \) being prime and for dimension \( s = 2 \) and \( n = 2^m \) it can be shown that a generator vector in Korobov form exists that obtains maximized minimum distance. If \( n \) is prime, every point \( x_j \neq 0 \) in the lattice \( L_{n,g} \) can be chosen as a generator vector, since \( \Phi(n) = n - 1 \), obviously including \( g = (1,a) \). For \( n \) being a power of 2, one of the two components of the generator \( g = (g_1, g_2) \) has to be odd (w.l.o.g. \( \gcd(g_1, n) = 1 \)). Otherwise points would coincide resulting in a minimum distance of 0. Then for every \( j \) with \( \gcd(j,n) = 1 \) every vector \( x_j = \{j \cdot g\}_n \) is a generator of the same lattice (generator of the cyclic group), too, and there must exist an \( l \in \{1,3,5,\ldots,n-1\} \) with \( x_l = (1,a) \).

Apart from the construction of equation (3.2) an exhaustive search is necessary in order to assure maximized minimum distance for general \( n \). There are several trivial possibilities to accelerate the exhaustive search algorithm. They are based on accelerating the minimum
distance calculation, though. Computing the minimum distance by the Gaussian reduction, reduces the complexity from the exhaustive computation case $O(n^3)$ to $O(n^2 \log(n))$. If the minimum distance in the currently considered lattice $L_{n,g}$ is already smaller than the maximum minimum distance found so far, the calculation of the minimum distance can be stopped. Furthermore, an experimental approach relies on the observation that if the maximum minimum distance has not changed for a certain amount of steps (10-20 in experiments), the maximum minimum distance is likely to have been found and the search can be stopped.

However, the last two optimizations do not change the order of complexity of the search algorithm, which means that the exhaustive search is still very slow for large $n$. Thus so far it is not possible to determine general MMD rank-1 lattices for a large number of points on-the-fly.

**Approximate Search for MMD Rank-1 Lattices**

Two observations make it possible to outperform the naïve search algorithm: First, if the number $n$ of lattice points is prime, all lattice points scaled by $n$ are generator vectors. As a consequence all shortest vectors scaled by $n$ must be generator vectors, too. Second, equation (3.3) states how to compute the largest possible minimum distance $l = \sqrt{\frac{2}{n+\sqrt{3}}}$ for a number of $n$ lattice points on the unit square. The resulting idea is to restrict the search space for generator vectors $g$ to a ring around the origin with inner radius $r$ and outer radius $R$, where $r = n \cdot l - \frac{k}{2} < n \cdot l < n \cdot l + \frac{k}{2} = R$, and $k \in \mathbb{N}$ is a selected positive integer (see Figure 3.9).

By rasterizing this ring on the integer lattice $\mathbb{N}^2$ using efficient algorithms from computer graphics, all potential generator vectors are enumerated. However, due to symmetry only one eighth of the ring needs to be rasterized (see Figure 3.11). Fixing the ring width $k$ independent of $n$, the rasterization runs in $O(n \cdot l) = O(\sqrt{n})$. The approximate search then
runs in \(O(\sqrt{n\log n})\), where the minimum distances are computed using Gaussian reduction. This complexity allows one to determine useful generator vectors for MMD rank-1 lattices on-the-fly even for large \(n\), as necessary for texture generation for example (see Chapter 5). An implementation of the rasterization search is given in Listing A.2, along with a circle rasterization in Listing A.7.

**Restriction of the Search Space** We computed the difference \(n \cdot l - ||g||\) for \(n = 4, \ldots, 10000\), where \(||g||\) is the length of the shortest generator vectors found by exhaustive search. Note that the generator vectors are integer vectors and therefore \(l\) has to be scaled by \(n\). The left graph in Figure 3.10 justifies the approach to restrict the search space to a ring of a fixed width \(k = R - r\): \(k = 4\) already covers quite a lot of the shortest generator vectors. Due to the complexity of the exhaustive search, the range of \(n > 10000\) has been investigated for random samples only. Altogether, a value of \(k = 6\) has proven to be a quite reasonable ring width as described now.

![Figure 3.10: Left: Difference \(n \cdot l - ||g||\) of the maximally possible length \(l\) scaled by \(n\) and the shortest generator vector of the exhaustive search for \(n = 4, \ldots, 10000\). Right: Ratio MMD\(_k\)/MMD\(_r\) for \(n \in \{4, \ldots, 131072\}\).](image)

**Numerical Evidence** For \(n = 4, \ldots, 10000\) and \(k = 6\) we compared the approximate rasterization search to the exhaustive search, i.e. the optimal results. In 99.1\% (i.e. 9908 out of 9997 cases), the approximate algorithm finds the optimal generator vector with respect to maximized minimum distance. The number of lattices for which an optimal generator vector coincides with a shortest vector equals 71\% (7098 cases), whereas in 28.1\% (2810 cases) an optimal generator vector is determined inside the ring with width \(k\), even if the generator vector is not a shortest vector. Otherwise the new search algorithm yields a maximized minimum distance that is never worse than 90\% of the optimum.

While the restricted search yields the correct results for \(n\) being prime, it is likely to also find optimal generator vectors for rank-1 lattices with respect to maximized minimum distance for non-prime \(n\). If the best generator vector is not found, at least an acceptable
one is found. Examples for the different cases are visualized in Figure 3.11: The search space is depicted by the light gray squares, which represent the rasterized octant of a ring with radius \( n \cdot l \) and width \( k = 6 \). Although hard to distinguish, the light gray circle is of radius \( n \cdot l \), while the black circle’s radius is the maximized minimum distance \( \text{MMD}_e \) determined by the exhaustive search. The set of generator vectors which result from this search algorithm and lie in the displayed range are plotted by the small black circles. The filled black dots belong to the lattice generated by the black vector as one element of the generator vectors resulting from the rasterization search with maximized minimum distance \( \text{MMD}_e \).

In the right graph of Figure 3.10 the maximized minimum distances resulting from searching \( \text{MMD} \) rank-1 lattices in Korobov form (\( \text{MMD}_k \)) and the rasterization search (\( \text{MMD}_r \)) are compared by plotting the ratio \( \frac{\text{MMD}_k}{\text{MMD}_r} \) for \( n = 4, \ldots, 10000 \), respectively. As apparent from the graphs the new search yields nearly optimal results with respect to the search criterion and delivers better results than the Korobov form in most cases. More precisely \( \text{MMD}_r \geq \text{MMD}_k \) in 99.1% of the cases, of which for 6.2% we have \( \text{MMD}_r > \text{MMD}_k \), if the \( \text{MMD} \) rank-1 lattice cannot be represented in Korobov form, and \( \text{MMD}_r = \text{MMD}_k \) in 92.9% of the cases.

Note that in two dimensions it is possible to retrieve the Korobov form \( g = (1, a) \) or \( g = (a, 1) \) of a general rank-1 lattice \( L_{n,g} \) if the generator vector \( g = (g_1, g_2) \) fulfills \( \gcd(n, g_1) = 1 \) or \( \gcd(n, g_2) = 1 \). This is accomplished by computing the multiplicative inverse \( x \) for \( g_1 \) or \( g_2 \), respectively: For \( \gcd(n, g_1) = 1 \) we have \( \left( g_1 \right)^{-1} \cdot x = \left( 1 \right)^{-1} \pmod{n} \), whereas in the second case the relation \( \left( g_2 \right)^{-1} \cdot x = \left( 1 \right)^{-1} \pmod{n} \) holds. The extended Euclidean algorithm \( \text{extendedEuclidean}(a, n) \) computes the triple \( (x, y, d) \) for the parameters \( (a, n) \) such that

\[
\gcd(a, n) = d = x \cdot a + y \cdot n
\]

If \( d = 1 \), i.e. \( \gcd(a, n) = x \cdot a + y \cdot n \equiv 1 \pmod{n} \), it follows that \( 1 \equiv x \cdot a \). Therefore the
multiplicative inverse with respect to \( a \) is given by \( \{x\}_n \). So in the case of \( \gcd(n, g_1) = 1 \) we calculate the parameter \( x \) for the multiplicative inverse with respect to \( g_1 \), setting \( a = \{x \cdot g_1\}_n \), and likewise in the case of \( \gcd(n, g_2) = 1 \), we compute \( x \) with respect to \( g_2 \), such that \( a = \{x \cdot g_1\}_n \).

### 3.2.2 MMD Rank-1 Lattice Sequences

In this section we apply the idea of infinite rank-1 lattice sequences (see Section 2.2)

\[
\{\Phi_b(i) g + \Delta\}_1 \mid i = 0, 1, 2, \ldots
\]

for maximized minimum distance lattices. This means that we search for rank-1 lattice sequences with maximized minimum distance in the sense that the weighted sum

\[
\sum_{m=m_{\min}}^{m_{\max}} (d_{\min}(L_{\Phi_b}^{g}))^2 b^m
\]

is maximized. Scaling the squared minimum distance by \( b^m \) gives equal importance to all lattices of the sequence since the area of a basis cell is \( \frac{1}{b^m} \). In the following let \( L_{\Phi_b}^{g} \) denote the lattice sequence of equation (2.18). As for the maximized minimum distance lattices, we only consider the corresponding sequences in two dimensions.

We examine two approaches to restrict the search space of the generator vectors, which yield similar results.

#### Lattice Sequence based on an Initial MMD Rank-1 Lattice

For \( l \in \mathbb{N}_0 \) and a fixed \( m \), each set of points \( \{x_{i \cdot b^m}, \ldots, x_{(i+1) \cdot b^m-1}\} \subset L_{\Phi_b}^{g} \) is a copy of \( L_{\Phi_b}^{g} \) shifted by \( \Delta(l) := \Phi_b(l) b^{-m} g \) [HHLL01] (see Section 2.2). The minimum distance of all copies is identical, as \( d_{\min} \) is shift invariant. For the example of \( L_{\Phi_b}^{g} \), this structural property [HH97] is depicted in Figure 3.12.

We now consider a two-dimensional generator vectors \( g = (g_1, g_2) \) with \( \gcd(n, g_1, g_2) = 1 \). All points of the rank-1 lattice sequence \( L_{\Phi_b}^{g} \) have to lie on a family of parallel hyperplanes defined by \( g \). The number of hyperplanes \( n_b \) in the unit square equals the number of times the hyperplane defined by the generator vector is wrapped around the unit.
The lattices $L_{n,(1,3)}$ of the lattice sequence $L_{(1,3)}^{\Phi_2}$ started with the initial MMD rank-1 lattice $L_{8,(1,3)}$.

square according to the definition of rank-1 lattices and can be computed in the following way. Due to simplicity the computations are performed on $[0,b^m)^2$ instead of the unit square. Neglecting the modulo operation, the largest $y$-value of a lattice point for $n=b^m, m \geq 1$ corresponds to $y = (b^m - 1) \cdot g_2$. Then the line spanned by the generator vector $(0,0) \mapsto (b^m - 1) \cdot (g_1,g_2)$ is wrapped \[ \left\lfloor \frac{(b^m - 1)g_2}{b^m} \right\rfloor g_2 - \left\lfloor \frac{g_2}{b^m} \right\rfloor g_2 \] times around $I=[0,b^m)^2$ with respect to the $y$-axis and \[ \left\lfloor \frac{(b^m - 1)g_1}{b^m} \right\rfloor g_1 - \left\lfloor \frac{g_1}{b^m} \right\rfloor g_1 \] with respect to the $x$-axis. Thus the number of hyperplanes in $[0,b^m)^2$ is $n_h = g_1 + g_2 - 1$ independent on the number of points and holds true for the complete lattice sequence. The subtraction of one in the previous term arises, since the hyperplane through the origin would be counted twice otherwise. As a consequence, the lattice points of the previous example $L_{(1,3)}^{\Phi_2}$ reside on at most three hyperplanes (see Figure 3.13). This means that the generator vector has to be modified such that the undesirable uniform bound on the minimum distance induced by the number of hyperplanes is improved.

Considering generator vectors of the form \[ g_{i,j} := (g_1 + i \cdot b^m, g_2 + j \cdot b^m) \text{ for } i,j \in \mathbb{N}_0, \]
we have $g_{i,j} \equiv g \mod b^m$. As a consequence $L_{b^m} = L_{b^m}^{\Phi_2}$, i.e. the minimum distance remains unchanged for $b^m$ points. However, the number of hyperplanes is increased to $n_h = g_1 + i \cdot b^m + g_2 + j \cdot b^m - 1$, as desired. For example $L_{(41,11)}^{\Phi_2}$ with $(41,11) = (1 + 5 \cdot 8,3 + 1 \cdot 8)$ does not restrict points to only three hyperplanes, but for $n = 8$ points generates the same rank-1 lattice as $L_{(1,3)}^{\Phi_2}$, i.e. $L_{8,(1,3)} = L_{8,(41,11)}$ (compare Figures 3.13 and 3.14).

The search procedure is started by selecting both a minimum number of points $b^{m_{\text{min}}}$ and maximum $b^{m_{\text{max}}}$. First a search from the previous section is run to find an initial MMD rank-1 lattice generator vector $g$ for $b^{m_{\text{min}}}$ points. Then the sum of minimum distances (3.4) is evaluated for each potential generator vector $g_{i,j}$ in order to find the maximum, where the search range is determined by

\[ g_1 + i \cdot b^{m_{\text{min}}} \leq b^{m_{\text{max}}} \Rightarrow i \leq \frac{b^{m_{\text{max}}} - g_1}{b^{m_{\text{min}}}} < b^{m_{\text{max}}-m_{\text{min}}} \text{ and} \]
\[ g_2 + j \cdot b^{m_{\text{min}}} \leq b^{m_{\text{max}}} \Rightarrow j \leq \frac{b^{m_{\text{max}}} - g_2}{b^{m_{\text{min}}}} < b^{m_{\text{max}}-m_{\text{min}}}. \]
Approximate Search by Restricting the Search Space

In the second approach the search is not based on an initial MMD rank-1 lattice. Instead we choose \( m_{\text{min}} = 1 \) and fix a value for \( m_{\text{max}} \), looking for a generator vector that maximizes equation (3.4).

In order to accelerate the search process, the search space can be restricted using the same strategy as in the rasterization search algorithm for rank-1 lattices (see Section 3.2.1). Then the search space is the union of the restricted search spaces for \( L_{\beta_{m}, g}, 1 < m \leq m_{\text{max}} \).

In experiments, the restricted search achieved the same results as the exhaustive computer search for \( n_{\text{max}} := b^{m_{\text{max}}} \leq 256 \) and \( b = 2, 3, 4 \), simultaneously reducing the run-time from \( \mathcal{O}(n_{\text{max}} \log n_{\text{max}}) \) to \( \mathcal{O}(\sqrt{n_{\text{max}}} \log n_{\text{max}}) \). See Listing A.4 for an implementation.

Comparing the minimum distances of the first \( b^m \) points of the sequence of this algorithm to the first approach, the sum of the minimum distances of the approach based on an initial MMD rank-1 lattice is less or equal to the sum of minimum distances of the second approach. Thereby the minimum distances are summed over the same range of \( m \).

Although the second approach is more general than the first one, the lattices produced by the sequence might not necessarily have the maximal possible minimum distance, which is assured at least for the initial lattice in the first approach.

Figure 3.16 shows the resulting lattice sequence for \( b = 3 \) and \( m_{\text{max}} = 7 \), while Table 3.2

<table>
<thead>
<tr>
<th>( m )</th>
<th>( d(m) ) first approach</th>
<th>( d(m) ) second approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>9</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>25</td>
<td>26</td>
</tr>
<tr>
<td>4</td>
<td>34</td>
<td>65</td>
</tr>
<tr>
<td>5</td>
<td>229</td>
<td>113</td>
</tr>
<tr>
<td>6</td>
<td>745</td>
<td>701</td>
</tr>
<tr>
<td>7</td>
<td>1033</td>
<td>2117</td>
</tr>
<tr>
<td>( \Sigma )</td>
<td>2075</td>
<td>3027</td>
</tr>
</tbody>
</table>

Table 3.2: Comparing the lattice sequences \( L_{\Phi_{3}(82,129)} \) and \( L_{\Phi_{3}(47,19)} \) with respect to the minimum distance of the first \( b^m \) points of the lattice sequence for \( b = 3 \) and \( 2 \leq m \leq 7 \). The initial MMD rank-1 lattice for \( L_{\Phi_{3}(82,129)} \) is given by \( L_{\beta_{3},(1,3)} \).
Figure 3.14: Searching an MMD rank-1 lattice sequence for the initial lattice $L_{2^1, (1, 3)}$ and $m_{\text{max}} = 7$ (see the previous example), yields $L_{(41, 11)}^{b_2}$ with $g_{5, 1} = (1 + 5 \cdot 2^3, 3 + 1 \cdot 2^3) = (41, 11)$. The gray lines show all possible hyperplanes. For each lattice of the rank-1 lattice sequence we compare its minimum distance $d(m) := d_{\text{min}}(L_{b_2, g_{5, 1}})^2 b^m$ to the maximum minimum distance that can be obtained by a single MMD rank-1 lattice.
Figure 3.15: Visualization of the search for a maximized minimum distance rank-1 lattice sequence based on an initial MMD rank-1 lattice for $L^{\text{init}}_{8,(1,3)}$ in base $b = 2$. The $x$-axis enumerates the generator vectors $\mathbf{g}_{i,j} := (g_1 + i \cdot b^m, g_2 + j \cdot b^m)$ for $i, j \in \mathbb{N}_0$, whereas the minimum distance divided by $b^m$ for $n = b^m$ is plotted on the $y$-axis. The green line marks the generator vector determined by the search algorithm.
Figure 3.16: $L_{(47,19)}^\Phi$ in base $b = 3$. For each lattice of the rank-1 lattice sequence we compare its minimum distance $d(m) := d_{\min}(L_{b^m, g})^2 b^m$ to the maximum minimum distance that can be obtained by a single MMD rank-1 lattice.
compares the minimum distances for the first \( b^m \) points of the lattice sequences resulting from both search algorithms presented in this section. By definition for \( m = 2 \) the lattice of the sequence \( L_{\Phi_3}^{(82,129)} \) represents an MMD rank-1 lattice, whereas for \( m = 3 \) the rank-1 lattice of the sequence \( L_{\Phi_4}^{(47,19)} \) achieves the maximal possible minimum distance as well. This structure might propose to use only the restricted search space for \( L_{BP_{m\max},g} \). However, this is not always sufficient to assure a large minimum distance for all \( m < m_{\max} \).

### 3.3 Search for Anisotropic Rank-1 Lattices

Lattices found by the previous approaches expose an isotropic spectrum, which means that the Fourier transform of the rank-1 lattice point set is isotropic. However, this is not always desirable in applications. We therefore present a simple method to determine the generator vector of a rank-1 lattice, such that a specific spectrum is approximated in two dimensions.

The Fourier transform of a rank-1 lattice point set \( L_{n,g} \) yields its dual lattice \( L_{n,g}^\perp \) [EDM04]. This means that we can describe the spectrum \( S_{n,g} \) of \( L_{n,g} \) by the fundamental Voronoi cell of \( L_{n,g}^\perp \), which is further characterized by its orientation \( \omega_{L} \) and width \( w_{L} \). These parameters are computed by means of the basis \( B^\perp \) of \( L_{n,g}^\perp \). Given a lattice basis \( B = (b_1, b_2)^T \) the dual basis can be easily determined by \( B^\perp = (B^{-1})^T \). In order to assure that \( B^\perp \) spans the Delaunay triangulation and thus the Voronoi diagram, the dual basis has to be reduced, for example using the Gaussian basis reduction and we assume \( \|b_1^\perp\| \leq \|b_2^\perp\| \).

Let
\[
v := \begin{cases} \frac{1}{2} (b_1^\perp + b_2^\perp) & \text{if } b_1^\perp \cdot b_2^\perp < 0 \\ \frac{1}{2} (b_2^\perp - b_1^\perp) & \text{otherwise} \end{cases}
\]
be one diagonal of the basis cell spanned by \( b_1^\perp \) and \( b_2^\perp \), such that \( v \) and \( b_1^\perp \) or rather \( v \) and \( b_2^\perp \) form a valid basis of the dual lattice as well. Then we approximate the orientation of the fundamental Voronoi cell by
\[
\overline{\omega}_{L} := b_2^\perp + v = \begin{cases} 2 \cdot b_2^\perp + b_1^\perp & \text{if } b_1^\perp \cdot b_2^\perp < 0 \\ 2 \cdot b_2^\perp - b_1^\perp & \text{otherwise}. \end{cases}
\]

The width \( w_{L} \) of \( S_{n,g} \) is defined as the length of the shortest basis vector normalized by the hexagonal bound \( l \) of equation (3.3), i.e.

\[
w_{L} = \frac{\|b_1^\perp\|}{l \cdot n}.
\]

Note that \( l \cdot n \) also represents an upper bound on the maximized minimum distance of the dual lattice, as the length of shortest vector in \( L_{n,g}^\perp \) corresponds to the length of the shortest vector in \( L_{n,g} \) scaled by \( n \) [Ski05]. Figure 3.18 shows the distinct spectra \( S_{n,g} \) which are possible for \( n = 8 \) points along with their orientation vectors \( \omega_{L} \) (green). The corresponding primal lattices are displayed in Figure 3.19.

54
Figure 3.17: Visualization of the search for a maximized minimum distance rank-1 lattice sequence in base $b = 3$ for $m_{\text{max}} = 7$. Similar to Figure 3.15 the x-axis enumerates the generator vectors and the minimum distance divided by $b^m$ for $n = b^m$ is plotted on the y-axis. The green line marks the generator vector determined by the search algorithm.
Figure 3.18: All distinct spectra, i.e. dual lattices, for $n = 8$ points.
Figure 3.19: The corresponding primal lattices to Figure 3.18. For each image the first of the generator vectors of the set generating $L_{8,g}$ is plotted.
The spectrum $T_{d,w}$, according to which we want to search the rank-1 lattice, is specified by its main direction, i.e., orientation, $d \in \mathbb{R}^2$ and its width $w$. The two-dimensional vector $d$ and the scalar $w$ are passed as an input parameter to the lattice search by an application. The width $w$ takes values in the range of $[0, 1]$ and represents the measure of desired anisotropy. The most anisotropic spectrum is denoted by $w = 0$, whereas $w = 1$ represents the isotropic one. Note that we have to allow $g_i = 0$, $i = 1, 2$ for the generator vector in order to be able to approximate spectra aligned to axes of the Cartesian coordinate system. For a fixed number of lattice points $n \in \mathbb{N}$ the search algorithm steps through all distinct lattices. This can be realized for example by using an $n \times n$ array, where the generator vectors of identical lattices are marked. Given any $g \in [0, n)^2$, the set of vectors yielding identical lattices is $\{k \cdot g \mod n | \gcd(n, k) = 1, k = 1, \ldots, n - 1\}$. After computing a Minkowski-reduced dual basis, for each dual lattice the orientation and width of the fundamental Voronoi cell are determined according to equations (3.5) and (3.6). Then the lattices are sorted with respect to $|w_L - w|$. For the smallest difference we choose the lattice, whose orientation $\vec{\omega}_L$ best approximates the main direction $d$ of $T_{d,w}$. Thereby the similarity $\text{sim}$ between those two vectors is measured by calculating the cosine of the angle between $\vec{\omega}_L$ and $d$

$$\text{sim} = \frac{d \cdot \vec{\omega}_L}{\|d\| \cdot \|\vec{\omega}_L\|}. \quad (3.7)$$

This is implemented in Listing A.5. Figure 3.20 shows an example for anisotropic rank-1 lattices having $n = 56$ points, where the spectrum is specified by $d = (\cos \alpha, \sin \alpha)$ with $\alpha = 303^\circ$ and the width varies from 0.1 to 1.0 in steps of 0.1. Using the Gaussian reduction for the lattice basis search, the algorithm runs in $O(n^2 \log n)$.

### 3.4 Weighted Norms

So far we have searched for rank-1 lattices on the unit square. However, applications are not restricted to quadratic domains. Simply scaling rank-1 lattices which have been searched on the unit square to a non-quadratic domain does not necessarily maintain the property of the lattice points with respect to the search criterion, as for example maximized minimum distance. Therefore the scale of the actual region has to be considered when changing the search domain.

Given a regular matrix $B^r$, we can generalize the search for rank-1 lattices to the domain spanned by the column vectors of $B^r$. All that needs to be done is considering the weighted norm

$$\|x_i\|_{\text{weighted norm}}^2 = ||B^r x_i||^2$$

in the definition of the minimum distance $d_{\text{min}}(L_{\text{org}})$ in equation (2.6) instead of the Euclidean norm [Kel06, Ski05]. Note that as before the distance to the origin has to be computed with respect to the unit torus.

In the following we show how to adapt the approximate search for MMD rank-1 lattices and the search of anisotropic rank-1 lattices to weighted norms.
Figure 3.20: Resulting spectra for a fixed direction $\mathbf{d} = (\cos 30^\circ, \sin 30^\circ)$ and width varying from 0.1 to 1.0.

- $w_L = 0.1$, $\mathbf{g} = (27, 2)$
- $w_L = 0.3$, $\mathbf{g} = (13, 2)$
- $w_L = 0.5$, $\mathbf{g} = (13, 1)$
- $w_L = 0.7$, $\mathbf{g} = (8, 7)$
- $w_L = 0.9$, $\mathbf{g} = (5, 7)$
- $w_L = 1.0$, $\mathbf{g} = (7, 4)$
Figure 3.21: Searching on a rectangular domain. Left: MMD rank-1 lattice $L_{512,(4,45)}$ in a domain of width-to-height ratio $x:y = 4:1$ in world coordinates. Right: The same lattice in the scaled basis with $x:y = 1:1$. The search region becomes an ellipse.

**Approximate Search for MMD Rank-1 Lattices**

For the special case of scaled rectangular domains, i.e. $B_r = (b_{r1}, b_{r2}) = ((x_0, 0), (0, y_0))$, the rasterization search can be adapted easily. Such domains occur for example in the context of texture generation where the image domain often is rectangular shaped. The left image in Figure 3.21 shows the MMD rank-1 lattice $L_{512,(4,45)}$ plotted in a domain having width-to-height ratio $x:y = 4:1$.

In order to account for the rectangular domain the lattice basis $B$ has to be transformed into Cartesian coordinates, before computing its determinant, i.e. area $A$. For the “weighted” lattice basis $B^w = B^r \cdot B$ the area of the basis cell is

$$A = |\det(B)^w| = |\det(B)^r| \cdot |\det(B)| = \frac{x \cdot y}{n} \quad \Rightarrow \quad l = \sqrt{\frac{2 \cdot x \cdot y}{n \cdot \sqrt{3}}}$$

in analogy to equation (3.3). Since we perform the rasterization directly in the sheared basis, the shortest vectors lie within an ellipse (see Figure 3.21). Its axes $a_x = (nl/x, 0)^\top$ and $a_y = (0, nl/y)^\top$ result from transforming the circle axes $(nl, 0)^\top$ and $(0, nl)^\top$ into the sheared basis $B^r$ of the actual region. The search process is visualized in Figure 3.22 for $n = 512$ and width-to-height ratio $4:1$.

As the rasterization runs in less than $O(\|a_x\| + \|a_y\|)$, with $\|a_x\|, \|a_y\| \in O(\sqrt{n})$, we still have a complexity of $O(\sqrt{n})$. Finally the Gaussian reduction needs to be adapted to weighted norms in order to compute the minimum distance. For that purpose the only modification consists in weighting the initial basis before performing the reduction steps. Therefore the search algorithm maintains a complexity of $O(\sqrt{n} \log n)$.

**Anisotropic Rank-1 Lattices**

In order to adapt the search for anisotropic rank-1 lattices of Section 3.3 to weighted norms the generator vector of a rank-1 lattice has to be determined such that a specific spectrum is approximated in Cartesian coordinates. This simply can be done by transforming the main direction $d \in \mathbb{R}^2$ into the sheared basis $B^r$ of the actual region.

$$d' = (B^r)^{-1}d$$
Figure 3.22: Visualization of the rasterization search for weighted norms. The color coding is analogous to Figure 3.11.

The remaining parts of the search algorithm stay the same and are performed in accordance with Listing A.5. An example for searching a rank-1 lattice according to \( \mathbf{d} = (1, 0) \) by a weighted norm is shown in Figure 3.23. Such lattices can be used for adaptive refinement in the context of anti-aliasing for instance and are applied for anisotropic rank-1 lattice textures (see Chapter 6).

3.5 \((t, m, 2)\)-Nets from Shifted Rank-1 Lattices

In Section 3.2 the quality of rank-1 lattices has been measured by the minimum mutual distance of the lattice points. Another method to quantify the uniform distribution of a given point set is to consider the number of points in all the partitions of the unit cube into rectangular boxes, which is specified by the concept of \((t, m, s)\)-nets and \((t, s)\)-sequences. For two integers \(0 \leq t \leq m\), a finite point set of \(b^m\) points in \(s\) dimensions is called a \((t, m, s)\)-net in base \(b\), if every elementary interval

\[
E := \prod_{j=1}^{s} \left( \frac{a_j}{b^{n_j}}, \frac{a_j + 1}{b^{n_j}} \right), \quad \text{for integers } n_j \geq 0 \text{ and } 0 \leq a_j < b^{n_j}
\]

of volume \(\lambda_s(E) = \frac{1}{b^{\sum_{j=1}^{s} n_j}} = b^{t-m}\) contains exactly \(b^t\) points [Nie92]. For \(t > 0\), an infinite point sequence is called a \((t, s)\)-sequence in base \(b\), if for all \(k \geq 0\) and \(m \geq t\), the vectors \(\mathbf{x}_{kb^m}, \ldots, \mathbf{x}_{(k+1)b^m-1} \in I^t\) form a \((t, m, s)\)-net. As smaller values of \(t\) assure a better uniform
Maximized minimum distance lattice $L_{8,1(5)}$ plotted in the basis of the lattice $L_{1048576,33919}$.

Lattice $L_{8,(3,2)}$ for the spectrum $(1,0)$ with similarity $\approx 0.998$ (see equation 3.7) in the sheared basis of $L_{1048576,33919}$.

Dual of the lattice $L_{8,(3,2)}$ along with the main direction $\mathbf{d} = (1,0)$.

Figure 3.23: Example for searching a rank-1 lattice according to the main direction $\mathbf{d} = (1,0)$ spectrum by a weighted norm.
distribution, \( t \) is said to be the quality parameter. Note the similarity of the definition of rank-1 lattice sequences (see Section 2.2) to the one of \((t,s)\)-sequences.

Figure 3.24 shows the maximized minimum distance rank-1 lattice \( L_{8,(1,5)} \), being a \((0,3,2)\)-net in base \( b = 2 \) at the same time. Such lattices will be called \((t,m,s)\)-lattices in the following.

But usually, maximized minimum distance lattices do not provide the property of \((t,m,s)\)-nets. Note that in order to achieve \( t = 0 \), the lattice points need to provide perfect low-dimensional projections, as illustrated in the leftmost and rightmost images of Figure 3.24, where the points are stratified equidistantly on each coordinate axis. This condition trivially is fulfilled by choosing Korobov lattices with \( \gcd(n,a) = 1 \). These lattices always form \((0,1,2)\)-nets in base \( b = n \) (see Table B.3). However, as seen in Section 3.2.1 this constraint reduces the maximum possible minimum distance.

Whereas the lattice \( L_{8,(1,5)} \) represents a \((0,3,2)\)-net in base \( b = 2 \) (as seen in Figure 3.24), this does not apply to the lattice \( L_{8,(1,3)} \) having the same minimum distance. Yet shifting \( L_{8,(1,3)} \) by \( \Delta \in \{ \left( \frac{1}{2},0 \right), \left( \frac{1}{8},0 \right), \left( \frac{7}{8},1 \right), \ldots \} \) yields a \((0,3,2)\)-net again. These shifts arise from the search algorithm presented in Listing A.6. Therefore we consider shifted rank-1 lattices in Korobov form in the following.

In order to get an idea of the arrangement of the potential shift vectors for \((t,m,2)\)-lattices we performed an exhaustive search for shifts \( \Delta = \left( \frac{i}{n}, \frac{j}{n} \right) \), with \( (i,j) \in \{0, \ldots, \left\lfloor \frac{n+1}{2} \right\rfloor \}^2 \). Restricting \( \Delta \) to this domain is sufficient due to the symmetry caused by performing all computations modulo 1. We took into account all shift vectors producing a valid \((t,m,2)\)-lattice regardless of the generator vectors and the resulting minimum distance, i.e. in this experiment \( n = b^m \) and \( t \) were the only fixed parameters. The results are displayed in Figure 3.25, where the dark gray squares represent shift vectors \( \Delta \cdot n \) producing the corresponding \((t,m,2)\)-lattices. \( \Delta = (0,0) \) is situated in the lower left corner of each image.

Subsequently, we only plotted those shift vectors for \((t,m,2)\)-lattices whose minimum distance is as large as possible using the same search domain for the shift vectors as above. The minimum distance belonging to these point sets can be found in Table B.3. As illustrated in Figure 3.26 the shift vectors lie on parallel, not necessarily equidistant and fully occupied diagonals which are wrapped around the search domain. These diagonals are mirrored for different lattices exposing the same minimum distance and the \((t,m,2)\)-net property. Combining all those diagonals in one image yields the patterns of Figure 3.25. This leads to the following observation: If a lattice can be shifted such that the \((t,m,2)\)-net property is achieved, shift coordinates exist near the origin, which can be exploited by the
Figure 3.25: The dark gray squares represent shift vectors $\Delta \cdot n$ within the domain $\{0, \ldots, \lfloor \frac{n+1}{2} \rfloor \}^2$, for which the rank-1 lattices become the corresponding $(t, m, 2)$-nets.
search algorithm.

As already seen in [Ski05] rank-1 lattices and \((0,2,2)-nets\) in base \(b\) are trivially connected due to the existence of only three kinds of elementary intervals

\[
\begin{align*}
\left[\frac{i}{b}, \frac{i+1}{b}\right) \times \left[\frac{j}{b}, \frac{j+1}{b}\right) & \quad \text{for} \quad 0 \leq i, j < b, \\
\left[\frac{i}{b^2}, \frac{i+1}{b^2}\right) \times [0, 1) & \quad \text{for} \quad 0 \leq i < b, \quad \text{and} \\
[0, 1) \times \left[\frac{i}{b^2}, \frac{i+1}{b^2}\right) & \quad \text{for} \quad 0 \leq i < b,
\end{align*}
\]

whereof each must contain exactly one of the \(n = b^2\) points of the shifted rank-1 lattice due to \(t = 0\). The latter two kinds of intervals guarantee perfect one-dimensional projections. Possible shift coordinates are given by the one-dimensional projections of the lattice points.

Since we consider lattices in Korobov form with \(\gcd(n,a) = 1\) it is sufficient to search shifts in only one of the elementary intervals of the first kind, such that the \(t = 0\) condition is fulfilled. To simplify matters the elementary interval \(\hat{E} = \left[0, \frac{1}{b}\right) \times \left[0, \frac{1}{b}\right)\) is chosen as shift domain. Scaling \(\hat{E}\) by \(n\) we can restrict to integer shifts \(\Delta \in [0, b)^2\) in the search algorithm, since rational shifts within this interval would not change the assignment of the lattice points to the elementary intervals. An illustration for \(L^3_{25}\), with \(\Delta = \left(\frac{2}{25}, \frac{2}{25}\right)\) is depicted in Figure 3.27.

Like \((0,2,2)-nets\) the resulting lattices share both the properties of jittered grid with respect to the stratification and Latin hypercube samples but can be computed faster by a simpler algorithm.

Now we extend the work of [Ski05] by searching for \((0,m,2)\)-lattices for \(m > 2\). On this account we change the shift domain such that the possible shifts \(\Delta\) only depend on the lattice itself. Given a lattice basis \(B\), a suitable shift domain is given by the corresponding fundamental parallelepiped. Due to the periodicity of the lattice, shift vectors outside this elementary cell \((\mathbf{b}_1 + \delta_x, \mathbf{b}_2 + \delta_y)\) result in the same shifted lattice as the shifts \((\delta_x, \delta_y)\). So the shift coordinates do not rely on \(m\) and the potential elementary intervals, respectively. Using the fundamental parallelepiped directly often results in negative shift vectors \(\Delta^-\), as it rarely is located in the first quadrant only. For ease of computation the negative shift vectors can be transformed into positive shifts \(\Delta^+\) by a modulo operation \(\Delta^+ = \Delta^- \mod n\). However, this might produce quite large positive shift vectors.

In order to get small positive shift vectors, in our experiments it turned out to be sufficient to shift the fundamental parallelepiped such that the lower left corner of its bounding box is anchored at the origin, as illustrated in Figure 3.28. The shift domain is discretized by using the points \((\frac{1}{n}, \frac{1}{n})\) lying inside the fundamental parallelepiped of the shifted bounding box. This allows to perform the computations in integer precision. Shifts \(\Delta = (\delta_x, \delta_y)\) lying outside the shifted lattice cell simply can be identified by expressing them in the lattice basis \(B\). For that purpose we have to keep in mind that the shift \(\Delta\) corresponds to the shift \(\Delta = \Delta + \mathbf{bbmin}\) in the original fundamental parallelepiped, with \(\mathbf{bbmin}\) being the lower left corner of the bounding box. Let \(\Delta^B = (\delta^B_x, \delta^B_y)\) be the shift \(\Delta\) transformed into the lattice basis \(B\). Then \((|\delta^B_x|, |\delta^B_y|) \neq (0,0)\) indicates that the considered shift vector is not located within the sought search domain.

65
Figure 3.26: Shift vectors $\Delta \cdot n \in \{0, \ldots, \left\lfloor \frac{n+1}{2} \right\rfloor \}^2$ for the corresponding $(t, m, 2)$-lattices with maximal possible minimum distance.
Figure 2.9: Lattice $L(25, 7)$ with its Minkowsky reduced basis $b_1 = (3, -4)$ and $b_2$. An elementary interval is defined by
\[ E := \prod_{j=1}^{s} [a_j n_j, a_j + 1 n_j), \]

**Example of a shifted rank-1 lattice**

A shifted rank-1 lattice is chosen as the bounding box (dashed) of the fundamental parallelepiped which is shifted such, that its lower left corner coincides with the origin.

Like in [Ski05] the search algorithm is structured in the following way: In a first step the list of generator vectors $g = (1, a)$, $a \in \{1, \ldots, \frac{n+1}{2}\}$ is sorted with respect to minimum distance. Next, we step through the sorted list in descending order, and iterate over all possible shift vectors of the shift domain in order to check whether the resulting lattice $L^A_{n,g}$ is a $(t, m, 2)$-net. The search algorithm differs from [Ski05] in the shift domain and the more general routine to verify the $(t, m, 2)$-property of the lattice. Moreover the minimum distance and the lattice basis (which is necessary to determine the shift domain) are computed at the same time using the Gaussian reduction algorithm 2.1, which is modified such that the two shortest basis vectors $b_1, b_2$ along with a vector $v = b_1 + b_2$ are returned in addition to the minimum distance. Note that $b_1, b_2$ follow the convention of Section 2.1.1.

\{0, b_1, b_2, v\} form the corners of the fundamental parallelepiped spanned by $b_1$ and $b_2$. See Listings A.10 and A.6 for the source code.

We performed the search for $(t, m, 2)$-lattices for $2 \leq b \leq 11$ and examined both the effect of increasing $m$ on the minimal gainable parameter $t$ and the minimum distance of the lattice points. The results of these experiments are shown in Table B.3, which also contains some of the $(0, 2, 2)$-lattices found in [Ski05].

Except for $b = 2$, the best quality parameter $t = 0$ was only attained for $1 \leq m \leq 3$. While for $m \leq 2$ the minimum distance is still close or equal to the maximized minimum distance, for $m = 3$ (and $m = 4$ in the case of $b = 2$) the minimum distance decreases extremely, as shown in Figure 3.29. Altogether the quality parameter $t$, $m$ and the minimum distance are related in the following way: The smaller $t$, the more the minimum distance decreases or $t$ has to increase for a larger minimum distance. For larger $m$, the smallest possible $t$
Figure 3.29: The \((0,3,2)\)-lattice in basis \(b = 5\) exposes a much smaller minimum distance than possible for lattices that do not form a \((t,m,s)\)-net (see table B.3).

increases to \(t > 0\). For example the \((0,4,2)\)-lattice in base \(b = 2\) and the \((0,2,2)\)-lattice in base \(b = 4\) have the same number of lattice points, with the first yielding a smaller minimum distance than the latter due to a larger number of elementary intervals. Hence those two quality parameters cannot be combined directly for rank-1 lattices.

[L’E96, L’E98, LL99] examine lattice rules in polynomial space, so called combined LSFR (Linear Shift Register Sequences) generators, which are maximally equidistributed. This property is related to the structure of a \((t,m,s)\)-net, i.e. can be expressed in the notion of a \((t,m,s)\)-net. Generally, this is not possible for LCG based lattices, namely Korobov lattices. An interesting visual resemblance of rank-1 lattices and \((t,m,s)\)-nets has been found in [GHSK08]. Grünschloss et. al. studied permutation generated nets in base 2 looking quite similar to rank-1 lattices at the first glance, since the points are lying on parallel diagonals similar to rank-1 lattices. This observation led to a construction for \((0,m,2)\)-nets by calculating “shifts” for each of the \(2^{\left\lceil \frac{m}{2} \right\rceil}\) diagonals, each one consisting of \(2^{\left\lfloor \frac{m}{2} \right\rfloor}\) points [Grü08], and not the whole point set as it is the case for shifted rank-1 lattices. The diagonals occur from wrapping around the first coordinate of the points modulo \(2^m\), which is quite similar to the generation of rank-1 lattices, where the generator vector is wrapped around the unit square. These diagonals can be considered as a family of hyperplanes covered by the points of a \((0,m,2)\)-net. In contrast to rank-1 lattices there is only one such family for these nets.
In this chapter we derive the fast Fourier transform on rank-1 lattices and the choice of the wave vectors along with applications in the context of simulation. Usually the transform has to be performed for each coordinate once. Instead of the standard tensor product algorithm, rank-1 lattices in $s$ dimensions allow for transforming the data using only the one-dimensional Fourier transform \cite{LH03}, which is simpler to implement and a little bit more efficient for the same number of lattice points. Note, that the curse of dimension cannot be eliminated in the general setting, as it is still hidden in the error bounds (see Section 2.3.2).

The set $\mathcal{K}_n := \{k_0, \ldots, k_{n-1}\} \subset \mathbb{Z}^s$ of wave vectors has to be selected such that for each wave vector we have

\begin{equation}
    k_j \in Z_j := \{k \in \mathbb{Z}^s | k \cdot g \equiv j \pmod{n}\},
\end{equation}

Figure 4.1: Illustration of the sets $Z_j := \{k \in \mathbb{Z}^s | k \cdot g \equiv j \pmod{n}\}, j = 0, \ldots, 7$ for the rank-1 lattice $L_{8,(1,3)}$.
where $g$ is the generator vector of the rank-1 lattice $L_{n,g}$ under consideration. Hence,

$$k_j \cdot x_l = k_j \cdot g = (j + r_j n) \frac{l}{n} = \frac{j l}{n} + r_j l$$

for some integer $r_j \in \mathbb{Z}$ and $x_l \in L_{n,g}$. Given Fourier coefficients $\hat{f}(k_j)$, synthesizing a function $f$ on the lattice $L_{n,g}$ by

$$f(x_l) = \sum_{j=0}^{n-1} \hat{f}(k_j) e^{2\pi \sqrt{-1} (\frac{n}{2} + r_j l)} = \sum_{j=0}^{n-1} \hat{f}(k_j) e^{2\pi \sqrt{-1} \frac{j l}{n}}$$

(4.2)

in fact turns out to be a one-dimensional finite Fourier series independent of the dimension $s$, because $r_j l$ is integer and therefore $e^{2\pi \sqrt{-1} r_j l} = 1$.

Given a function $f(x_l)$ the fast Fourier transform can be used for the analysis, too:

$$\hat{f}(k_j) = \frac{1}{n} \sum_{l=0}^{n-1} f(x_l) e^{-2\pi \sqrt{-1} \frac{j l}{n}}.$$

For $n$ being a power of two, the fast inverse Fourier transform can synthesize the function in all lattice points efficiently.

We illustrate the idea of performing the Fourier transform on rank-1 lattices for the generation of ocean waves as stochastic field synthesis in Section 4.2 and for a simple fluid dynamics simulation in Section 4.3. The most prominent example is the simulation of the ocean surface [FR86, Tes00] by random fields as used in the movies Titanic, Waterworld, or The Devil’s Advocate [Ent]. The same principle has been applied to modeling of turbulent wind fields and various other phenomena [SF91, SF93, Sta95, Sta97].

4.1 Choosing the Wave Vectors

There are infinitely many choices of wave vectors as defined by the sets $Z_j$ in equation (4.1). This is illustrated in Figure 4.1 for the lattice $L_{8,(1,3)}$. By definition the set $Z_0$ is equal to the dual lattice

$$L_{n,g}^\perp := \{ k \in \mathbb{Z}^s : k \cdot g \equiv 0 \pmod{n} \} = Z_0.$$

For $j > 0$, the $Z_j = Z_0 + \Delta_j$ result as shifted sets of $Z_0$. In the case of Korobov lattices the shift vectors $\Delta_j = (j, 0, \ldots, 0)$ can be computed explicitly. Let $k' \in Z_0$, i.e. $k' \cdot g \equiv 0 \mod n$,
and let $k \in \mathbb{Z}_j$. Then

$$k \cdot g \equiv j \pmod{n}$$

$$\iff \exists \lambda \in \mathbb{Z} \quad k \cdot g = \lambda \cdot n + j$$

$$\iff \exists \lambda \in \mathbb{Z} \quad k \cdot g - j = \lambda \cdot n$$

$$\iff \exists \alpha \in \mathbb{Z} \quad (k_0 \cdot g_0 + k_1 \cdot g_1 + \ldots + k_{s-1} \cdot g_{s-1} - j) = \lambda \cdot n$$

$$\iff \exists \alpha \in \mathbb{Z} \quad \left( \begin{array}{c} k_0 \\ k_1 \\ \vdots \\ k_{s-1} \end{array} \right) + \left( \begin{array}{c} -j \\ 0 \\ \vdots \\ 0 \end{array} \right) \cdot g = \lambda \cdot n$$

$$\iff \exists \alpha \in \mathbb{Z} \quad k' \cdot g = \lambda \cdot n$$

$$\iff k' \cdot g \equiv 0 \pmod{n}$$

This means that

$$k' = k + \left( \begin{array}{c} -j \\ 0 \\ \vdots \\ 0 \end{array} \right)$$

and therefore

$$k = k' + \left( \begin{array}{c} j \\ 0 \\ \vdots \\ 0 \end{array} \right) \Delta_j$$

Understanding the connection between the wave vectors and the structure of a rank-1 lattice helps to choose the best wave vectors for a given problem and provides a way to construct these wave vectors. Let $B^\perp = (B^{-1})^T$ denote the basis of the dual lattice $L_{n,g}^\perp$ of a rank-1 lattice $L_{n,g}$. The set $K_n$ of wave vectors is $B^\perp$ periodic \cite{PM62}, i.e. it has the property that the sets

$$K'_n := K_n + B^\perp 1 \quad 1 \in \mathbb{Z}'$$

are valid sets of $n$ wave vectors with $K'_n \cap K_n = \emptyset$ (for $1 \neq 0$) as well. Consequently, any tile that results in a monohedral tiling of $\mathbb{Z}'$ (see the illustration in Figure 4.2) with periodicity $B^\perp$ can be used as a set of wave vectors \cite{LH03}.

Once such a tiling is chosen all integer vectors in the interior of one cell are the wave vectors. Note that the choice of the tiling is arbitrary and the elements of a single tile are not necessarily connected. However, one can use spectral properties of the function that should be synthesized or analyzed in order to adapt the selection of the wave vectors to the given problem.

If no spectral properties are known, a reasonable assumption for practical problems is an isotropic spectrum (i.e. no preferred direction) and that low frequencies are most important.
This results in choosing the wave vectors in the fundamental Voronoi cell of $L_{n,g}^\perp$, which is illustrated in Figure 4.3.

The above assumptions provide another approach leading to maximized minimum distance in the dual and primal lattice, respectively, as a criterion for choosing the generator vector $g$ of a rank-1 lattice $L_{n,g}$:

The spectrum, i.e. the fundamental Voronoi cell, becomes most isotropic if $g$ is chosen such that the in-circle of the fundamental Voronoi cell of the dual lattice is maximized. This is equivalent to maximizing the sampling efficiency, which is defined in Section 2.1.4. For rank-1 lattices this ratio can be maximized by choosing the generator vector such that the minimal distance between any two points of the dual lattice is maximized. Due to the self-duality discussed in Section 2.1.3 this is equivalent to maximizing the minimum distance in the primal lattice for $s = 2$. Figure 4.4 shows the Voronoi diagrams of different
Figure 4.4: Sampling efficiency η of different lattices with n = 144. Note that the Fibonacci lattice with g = \(\frac{1}{89}\) is not the best choice with respect to sampling efficiency.

rank-1 lattices, where a rectangular and hexagonal lattice are shown for comparison. The hexagonal lattice is optimal with respect to the sampling efficiency in two dimensions and maximizing the minimum distance in a rank-1 lattice yields a good approximation. With increasing number of points the sampling efficiency of rank-1 lattices approaches the sampling efficiency of the hexagonal lattice.

Of course, if other spectral properties are known, the wave vectors can be chosen in the fundamental Voronoi cell of an anisotropic rank-1 lattice (Section 3.3) for a better approximation of this kind of functions.

### 4.2 Spectral Synthesis of Ocean Waves

Along the method used by Tessendorf [Tes00], a periodic ocean tile (see Figure 4.5) is realized as a stochastic field using Fourier synthesis on a rank-1 lattice. With the Fourier coefficients

\[
\tilde{h}(\mathbf{k}, t) := \tilde{h}_0(\mathbf{k}) e^{i\omega(\mathbf{k}) t} + \tilde{h}_0^*(\mathbf{\bar{k}}) e^{-i\omega(\mathbf{\bar{k}}) t},
\]

the height field

\[
h(x, t) := \sum_{j=0}^{n-1} \tilde{h}(\mathbf{k}, t) e^{2\pi i \frac{jl}{n}}
\]

becomes periodic in time t and real. For deep water the speed of a wave is given by the dispersion relation \(\omega(\mathbf{k}) = \sqrt{g ||\mathbf{k}||}\), where \(g\) is the gravitational constant. Based on observations from oceanography waves can be modeled statistically independent and normally distributed. Therefore, the amplitudes

\[
\tilde{h}_0(\mathbf{k}) := \frac{1}{\sqrt{2}} (\xi_r + i\xi_i) \sqrt{P_h(\mathbf{k})}
\]

are realized using Gaussian random numbers \(\xi_r\) and \(\xi_i\) modulated by a spectrum. Out of many alternatives we chose the Phillips spectrum

\[
P_h(\mathbf{k}) := A e^{-\frac{1}{2gL^2|\mathbf{k} \cdot \mathbf{w}|^2}} |\mathbf{k} \cdot \mathbf{w}|^2
\]

Phillips constant

\[
A
\]

Largest wave for wind speed v

\[
L = \frac{v^2}{g}
\]

wind direction
which considers parameters like wind speed and direction. For the sake of completeness we mention that the gradient vector of the height field can be computed using the Fourier transform as well. This yields more precise normals for shading as compared to those computed by finite differences.

**Implementation Details**

The synthesis of stochastic fields on rank-1 lattices consists of the following choices and decisions:

**Number \( n \) of lattice points:** Although rank-1 lattices exist for any number of points in any dimension, the fast Fourier transform is most efficient for \( n \) being a power of 2.

**Generator vector \( \mathbf{g} \):** In order to maximize the sampling efficiency we select a maximized minimum distance rank-1 lattice \( L_{n, \mathbf{g}} \) for \( n = 2^m, \ m \in \mathbb{N} \). Due to the choice of the number of lattice points as a power of 2, we are able to use Korobov lattices, which achieve maximized minimum distance in this case (see Section 3.2.1). Apart from applying the spectral test for the lattice search, this also has the advantage of a trivial addressing scheme. A list of all parameters for 2-dimensional maximized minimum distance Korobov lattices can be found in Table B.2.
**Basis** $B$: $B$ is determined as a Minkowski-reduced basis, which defines the Delaunay triangulation that is used as the triangle mesh. Table B.2 lists these basis vectors multiplied by $n$. Given a generator vector in Korobov form, the first coordinate of each of these integer basis vectors is the increment or decrement to find the index of a neighboring lattice point. See Section 2.1.5 for a more detailed description.

**Wave vectors** $K$: We enumerate all wave vectors in a conservative bounding box of the fundamental Voronoi cell of the dual lattice and select the $n$ shortest ones. As a simple conservative convex hull we chose the axis-aligned bounding box determined by the direct lattice point neighbors of the origin. A much more involved approach is to compute the fundamental Voronoi cell in the dual lattice and rasterize it on the integer lattice.

### 4.3 Stable Simulation of Fluids

The stable fluids algorithm by Stam [Sta99] is a practical way of simulating incompressible fluids for animation. Note that the algorithm focuses on real-time simulation rather than on precision. The simulation is based on the Navier-Stokes equations

\[
\begin{align*}
\quad \text{div } \mathbf{v} &= 0 \quad (4.3) \\
\quad \frac{\partial \mathbf{v}}{\partial t} &= -(\mathbf{v} \cdot \nabla)\mathbf{v} + \nu \Delta \mathbf{v} + \mathbf{f} \quad (4.4)
\end{align*}
\]

for incompressible fluids, where $\mathbf{v}$ is the velocity field, $\nu$ the viscosity, and $\mathbf{f}$ are the external forces. The solution strategy is to simulate equation (4.4) and remove the divergence (4.3) at the end of each time step by using a projection based on the Helmholtz-Hodge decomposition $\mathbf{w} = \mathbf{v} + \nabla q$, which states that a vector field $\mathbf{w}$ can be decomposed into a divergence free part $\mathbf{v}$ and the gradient of a scalar field $q$. The velocity field for the next time step $t + \Delta t$ is computed in four steps [Sta99]:

1. The external forces are added: $\mathbf{v}_1(x_i) := \mathbf{v}(x_i) + \Delta t \cdot f(x_i)$

2. The advection is computed by tracing back a particle back in time starting from point $x_i$ according to the velocity field. The position $p(x_i, -\Delta t)$ is computed by dividing the time step $\Delta t$ into smaller time steps and performing the Euler rule for each of the small time steps (see Figure 4.6). The velocities $\mathbf{v}_2(x_i) := \mathbf{v}_1(p(x_i, -\Delta t))$ are linearly interpolated using the closest lattice points. Due to linear interpolation the method is named stable, because the computed velocities can never exceed the old ones in magnitude.

3. The diffusion by the Laplace operator is efficiently computed as low pass filter in the Fourier domain. The Fourier coefficients are computed by

$$
\hat{\mathbf{v}}_2(k_j) := \sum_{l=0}^{n-1} \mathbf{v}_2(x_i) e^{-2\pi i j l / n}
$$
and filtered
\[ \hat{v}_3(k_j) := \frac{\hat{v}_2(k_j)}{1 + \nu \Delta t \cdot (k_j \cdot k_j)}. \]

4. The divergence is removed using the projection
\[ \hat{v}_4(\vec{k}_j) := \hat{v}_3(\vec{k}_j) - \frac{(k_j \cdot \hat{v}_3(\vec{k}_j))k_j}{(k_j \cdot k_j)} \]
and the velocity field at time step \( t + \Delta t \) is synthesized by
\[ v_4(\vec{x}_l) := \sum_{j=0}^{n-1} \hat{v}_4(\vec{k}_j)e^{2\pi i \vec{x}_l \cdot \vec{k}_j / \Lambda}. \]

**Implementation Details**

The stable fluids scheme has been implemented for two- and three-dimensional velocity fields as illustrated in Figure 4.7. The Fourier transformation techniques are the same as in the previous application example, except that they have to be performed for each component of the vector fields.

For linear interpolation (as required in step 2 of the algorithm) on a rank-1 lattice, the Minkowski-reduced basis \( B \) is used to access neighboring lattice points. The consideration of the following issues simplifies the implementation:

**Scaling** all lattice points by \( n \) results in integer coordinates for all \( x_l \) and the entries of the basis \( B \).

**Frame:** Representing the velocity field \( v \) in the basis \( B \) avoids a transformation during interpolation. In this case external forces have to be transformed into the basis \( B \).

**Accessing lattice cells:** The backtracking step requires to compute the index of a lattice cell containing a given point, which is simple if the lattice is given in Korobov form:
Figure 4.7: Left: The four images are subsequent snapshots of the stable fluid simulation on a rank-1 lattice. The arrows indicate the external forces applied to the periodic fluid. The fluid transports the background image and the effects of advection and diffusion, i.e. blur, are clearly visible. Right: Snapshot of a wind field simulation in three dimensions for the lattice $L_{32768}^g$ with $g = (1, 10871, 10871^2)$, where smoke is transported in a velocity field.

Multiplying the basis matrix $B$ by the integer parts of the coordinates of the point modulo $n$ yields a lattice point in Cartesian coordinates. Obviously the first component is the lattice point or cell index. Neighboring lattice points for interpolation now are found as described in the previous example (see Section 2.1.5).
5 Rasterization on Rank-1 Lattices

Contrary to vector screens, raster displays are composed of a lattice of picture elements (pixels). A pixel can either be interpreted as a function sample or as a functional, e.g. the integral over a filter kernel which depends on the display device. Geometric primitives, such as lines, triangles, or circles are usually specified in terms of 2-dimensional vertices on a Cartesian grid. In order to display them, they are approximated by sets of pixels generated by a so-called rasterizer [FvDFH96].

The two basic approaches to rasterization are:

- The geometric primitive is represented in explicit, i.e. parametric, form, which for the example of a line results as

  \[ P(t) = \begin{pmatrix} x_0 + t(x_1 - x_0) \\ y_0 + t(y_1 - y_0) \end{pmatrix} \]

  for \( t \in [0, 1] \) and the start point \( P_0 = (x_0, y_0) \) and the end point \( P_1 = (x_1, y_1) \). An example for a parametric line rasterization algorithm is the DDA (Digital Differential Analyzer), see [FvDFH96].

- The geometric primitive is defined by its implicit form, as for example the implicit line equation

  \[ f(x, y) = (y_1 - y_0)x - (x_1 - x_0)y - x_0y_1 = 0, \]

  for the start point \( P_0 \) and end point \( P_1 \), as defined above. The most common algorithms to draw lines in this setting are the Bresenham [Bre65] or midpoint algorithm [Pit67].

Using rank-1 lattice geometry as the pixel layout yields new algorithms for rasterization. Instead of rasterizing on a rectangular grid, in [Ski05] the rasterization is performed on the lattice cells induced by the fundamental parallelepiped of a MMD rank-1 lattice. As this may result in jagged lines due to the shape of the lattice cells, the Voronoi cells represent a better choice. However, this has only been mentioned briefly in [Ski05].

As proposed in [Ski05], the basic idea for adapting the traditional rasterization algorithms for rank-1 lattices is based on a change of frame. This means that the rasterizer works in the coordinate system which is given by the basis of the corresponding rank-1 lattice instead of the Cartesian coordinate system. Then the rasterization can be performed in the same way as in the case of the Cartesian lattice, as has been shown for the DDA in [Ski05].
5.1 Lines

Using the Voronoi cells as pixels for rasterization instead of the lattice cells induced by the fundamental parallelepiped, the idea for performing the DDA on rank-1 lattices principally remains the same. However, we have to take into account that the neighborhood of the pixels of the rank-1 lattice is different: Whereas the square pixels and the lattice cells have the same neighborhood, i.e. four direct or nearest and four indirect neighbors, a hexagonally shaped Voronoi cell has six nearest neighbors. Consequently, if the coordinates of the current pixel change in direction of both basis vectors in the next step, the next selected pixel is an indirect neighbor of the current pixel in case of the lattice cells, whereas the next pixel is not connected to the current one in case of the Voronoi cells. Therefore a straight line on the lattice cells may not be connected on the Voronoi cells using the same coordinates as the lattice cells, as illustrated in Figure 5.1.

![Figure 5.1: DDA on the lattice $L_{124,(1.5)}$ for the start and end points $P_0 = (5, -3)$ and $P_1 = (8, 2)$, which are given in lattice coordinates: (a) Rasterization on the lattice cells induced by the fundamental parallelepiped. (b) Plotting the line coordinates of (a) in the Voronoi diagram. (c) Rasterizing the line by taking into account the changed neighborhood conditions.](image)

A possible solution to this problem consists in checking whether there is a coordinate change along both basis vectors at each rasterization step. This can be done by computing the length of the vector $(u, v) = (x_n, y_n) - (x_c, y_c)$ connecting the current pixel $(x_c, y_c)$ and the next pixel $(x_n, y_n)$. If this length equals the length of the longer diagonal of the Delaunay cell, i.e. corresponds to $||b_2|| + ||b_1||$ if $b_1 \cdot b_2 > 0$, or $||b_2|| - ||b_1||$ otherwise, there will be a gap in the rasterized line. In this case we go half a step backwards after computing the next point $(\tilde{x}_{i+1}, \tilde{y}_{i+1}) = (\tilde{x}_i + \Delta x, \tilde{y}_i + \Delta y)$ on the line, which corresponds to computing the midpoint $(\tilde{x}_m, \tilde{y}_m) = (\frac{\tilde{x}_{i+1} + \tilde{x}_i}{2}, \frac{\tilde{y}_{i+1} + \tilde{y}_i}{2}) = (\tilde{x}_{i+1} - \frac{\Delta y}{2N}, \tilde{y}_{i+1} - \frac{\Delta x}{2N})$ between the current point $(\tilde{x}_i, \tilde{y}_i)$ and the next point $(\tilde{x}_{i+1}, \tilde{y}_{i+1})$ on the line. Thereby $\Delta x = x_e - x_s$ and $\Delta y = y_e - y_s$ for the start point $P_0 = (x_s, y_s)$ and the end point $P_1 = (x_e, y_e)$, and $N = \max\{|\Delta x|, |\Delta y|\}$ is the number of rasterization steps. There are two pixels $(x_r, y_r) = (x_c, y_c + v)$ and $(x_l, y_l) = (x_c + u, y_c)$ which are suitable to fill the gap. Note that the vector $(u, v)$ connects $(x_c, y_c)$ and $(x_n, y_n)$.
Figure 5.2: Close up of Figure 5.1 (b): \((x_c, y_c)\) highlights the current pixel, \((x_n, y_n)\) the next one. \((x_l, y_l)\) denotes the left potential pixel to fill the gap, whereas \((x_r, y_r)\) is the right one. The hollow dot represents the midpoint \((\hat{x}_m, \hat{y}_m)\) between the current and the next point on the line. As \((\hat{x}_m, \hat{y}_m)\) is closer to \((x_l, y_l)\) than to \((x_r, y_r)\), we choose \((x_l, y_l)\) as the next pixel being colored.

as defined above. We choose the pixel whose distance to the midpoint \((x_m, y_m)\) is smallest. This process is illustrated in Figure 5.2. A pseudo-code algorithm is given in Listing 5.1. Note that the ifloor operation rounds down the coordinates to the nearest integer, e.g. ifloor\((-1.7, 2.9) = (-2, 2)\). This approach reminds of the midpoint algorithm [Pit67], an example of which is shown in the next section for the example of rasterizing a circle.

```c
// start point (xs, ys) and end point (xe, ye) in lattice coordinates
DDA_on_rank1Lattice(xs, ys, xe, ye) {
    (xc, yc) = ifloor(xs, ys);
    dx = xe - xs;
    dy = ye - ys;
    n = max(\(|dx|\), \(|dy|\));
    steps = n;
    (x, y) = (xs, ys);
    do {
        x += dx/n; y += dy/n; // next point on the line
        (xn, yn) = ifloor(x, y); // pixel to be colored
        drawVoronoiCell(xn, yn);
        (u, v) = (xn, yn) - (xc, yc);
        if (||(u, v)|| = (||b2|| + ||b1||)){
            xm = x - dx/2n; ym = y - dy/2n;
            (xr, yr) = (xc, yc + v);
            (xl, yl) = (xc + u, yc);
            if \(||((xr, yr)-(xt, yt)|| < (xl, yl)-(xt, yt)||\)
                drawVoronoiCell(xr, yr);
            else
                drawVoronoiCell(xl, yl);
        }
        (xc, yc) = (xn, yn);
    } while(steps--)
}
```

Listing 5.1: DDA on Voronoi diagram of a rank-1 lattice.
5.2 Circles

In this section we adapt Bresenham’s midpoint circle algorithm [Bre77, FvDFH96] to rank-1 lattices. This algorithm rasterizes a circle centered at the origin by incrementally computing all pixels which are closest to the circle. In order to scan-convert circles which are not anchored at the origin, they can be translated to the origin before applying the rasterization algorithm with the pixels being plotted using the corresponding offset.

On the square lattice the midpoint algorithm takes advantage of a circle’s eight-way symmetry, which means that it is sufficient to consider only 45° of a circle. Rasterizing a circle on the hexagonal lattice, even a twelve-fold symmetry can be exploited [MS05]. As the basis vectors generally are not of the same length in a rank-1 lattice, i.e. $\|b_1\| \neq \|b_2\|$, we can only take advantage of point symmetry for rasterizing a circle on this kind of lattice. For that purpose, we divide the circle into six sectors, where the sectors $-1$, $-2$ and $-3$ result from the sectors $1$, $2$ and $3$ by point symmetry (see Figure 5.3). W.l.o.g. we assume that $b_1 \cdot b_2 > 0$ for the rest of this section. This means that the angle between the two basis vectors is smaller 90°. Let

$$b_3 := b_2 - b_1,$$

such that $b_3$ and $b_1$ or rather $b_3$ and $b_2$ form a valid basis of the rank-1 lattice as well. Thus $b_3$ can be considered as a third, redundant basis vector and we can enumerate all six nearest neighbors to a lattice point by $b_1$, $b_2$ and $b_3$. These three vectors are used to determine the boundaries of the six circle sectors. Each sector itself is again divided into two parts $a$ and

Figure 5.3: Circle midpoint algorithm on a rank-1 lattice.
by its bisecting line. In sector 1 the bisecting line is given by \( u - v = 0 \), in sector 2 by \( 2u + v = 0 \) and in the third sector by \( u + 2v = 0 \). Note that although the two half sectors look symmetrically, this is not the case since \(|b_i| \neq |b_j|\), for \( i, j \in \{1, 2, 3\} \) and \( i \neq j \). See the dark gray boundary pixels of the second sector for example.

Let \((u, v)\) be a two-dimensional point in the lattice basis and \((x, y)\) a point in Cartesian coordinates. Then \((u, v)\) is transformed into the Cartesian coordinate system by

\[
\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} b_{11} & b_{21} \\ b_{12} & b_{22} \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} b_{11}u + b_{21}v \\ b_{12}u + b_{22}v \end{pmatrix},
\]

(5.1)

where \(b_1 = (b_{11}, b_{21})^T\) and \(b_2 = (b_{21}, b_{22})^T\). As the basis vectors are not of equal length we define the radius along the first basis vector, i.e. the radius is given in units of \(|b_1|\). This means that the vector \((r, 0)\) in lattice coordinates corresponds to \((b_{11}, b_{12}) \cdot r\) in Cartesian coordinates and hence, \(r^2 = ||(r, 0)||^2\) results in \(\delta := ||b_1||^2 r^2\). Applying equation (5.1) to the implicit circle function

\[
r^2 = x^2 + y^2,
\]
a circle on a rank-1 lattice can be represented in Cartesian coordinates in the following way:

\[
||b_1||^2 r^2 = (b_{11}u + b_{21}v)^2 + (b_{12}u + b_{22}v)^2 = (b_{11}^2 + b_{21}^2)u^2 + 2uv(b_{11}b_{21} + b_{12}b_{22}) + (b_{21}^2 + b_{22}^2)v^2
\]

\[
= ||b_1||^2 u^2 + 2b_1 \cdot b_2 uv + ||b_2||^2 v^2 = \alpha u^2 + 2\beta uv + \gamma v^2.
\]

(5.2)

Therefore the implicit function results as

\[
D(u, v) = \alpha u^2 + 2\beta uv + \gamma v^2 - \delta.
\]

(5.3)

This function is zero for a point \((u, v)\) lying on the circle, positive if the point lies outside the circle, and negative inside the circle.

The principle of the midpoint algorithm on rank-1 lattices remains the same as for the square lattice. The algorithm works by choosing which of two pixels lies closer to the circle by computing the midpoint between the two pixels, which is inserted into the implicit function describing the circle. Depending on the sign, the corresponding pixel is chosen. Each sector of the circle is rasterized by walking from the sector boundaries to its bisecting line using the same number of steps. In order to prevent coloring a pixel twice, in the first sector we walk towards the bisecting line as long as \( u - 1 > v \). In the same manner the condition \( v - 1 > -2u \) suffices in the second sector, whereas we have \( 2v - 1 > -u \) in the third one. Due to the same reason the starting point in each \( b \) half sector is not colored either.

In the sector 1a the next pixel has to be selected between the pixels \( SE \) and \( E \), based on a previously chosen pixel \( P = (u_p, v_p) \). See Figure 5.3 for illustration. The decision is
made by means of a decision variable \( d_{\text{old}} \), which is the value of the function \( D(u, v) \) at the midpoint \((u_p - 0.5, v_p)\) between \( SE \) and \( E \). Therefore

\[
d_{\text{old}} = D(u_p - 0.5, v_p) = (u_p - 0.5)^2 \alpha + 2(u_p - 0.5)(v_p + 1)\beta + (v_p + 1)^2 \gamma - \delta
\]

If the midpoint between \( SE \) and \( E \) is inside the circle, i.e. \( d_{\text{old}} < 0 \), \( E = (u_p, v_p + 1) \) is closer to the circle and the next midpoint is one increment up in the direction of \( \mathbf{b}_2 \):

\[
d_{\text{new}} = D(u_p - 0.5, v_p + 2) = (u_p - 0.5)^2 \alpha + 2(u_p - 0.5)(v_p + 2)\beta + (v_p + 2)^2 \gamma - \delta
\]

This yields the increment

\[
\Delta E = d_{\text{new}} - d_{\text{old}} = (2u_p - 1)\beta + (2v_p + 3)\gamma. \quad (5.4)
\]

If the midpoint is outside (or on) the circle, i.e. \( d_{\text{old}} \geq 0 \), then the pixel \( SE = (u_p - 1, v_p + 1) \) lies closer to circle, yielding the new midpoint \((u_p - 1.5, v_p + 2)\). Thus, the new decision variable is

\[
d_{\text{new}} = D(u_p - 1.5, v_p + 2) = (u_p - 1.5)^2 \alpha + 2(u_p - 1.5)(v_p + 2)\beta + (v_p + 2)^2 \gamma - \delta
\]

and \( d_{\text{new}} = d_{\text{old}} + \Delta SE \) for

\[
\Delta SE = (2 - 2u_p)\alpha + (2u_p - 2v_p + 5)\beta + (2v_p + 3)\gamma. \quad (5.5)
\]

Note that selecting the pixel \( SE \) in the case \( d_{\text{old}} = 0 \) is an arbitrary choice. The midpoints, the points \( SE \) and \( E \), and the corresponding equations (5.4) and (5.5) are computed similarly for the other sections, which is summarized in Table 5.1.

As the radius is defined in units of \( \mathbf{b}_1 \) the starting point in the sector 1a simply corresponds to \( \mathbf{S}_1 = (r, 0) \) such that the initial midpoint is given by \((r - 0.5, 1)\) yielding

\[
d_0 = D(r - 0.5, 1) = (r^2 - r + 0.25)\alpha + (2r - 1)\beta + \gamma - \delta.
\]

Due to \(|\|\mathbf{b}_i\|| \neq |\|\mathbf{b}_j||\), for \( i, j \in \{1, 2, 3\} \) and \( i \neq j \), the start points in the other sectors are not trivially given by \( \mathbf{S}_2 = (0, r) \) and \( \mathbf{S}_3 = (-r, r) \), respectively. Instead their position vectors have to be scaled such that \(|\|\mathbf{S}_2\|| = r \) and \(|\|\mathbf{S}_3\|| = r \). This is achieved by \( \mathbf{S}_2 = [(0, r_2)], \) with \( r_2 = r\frac{|\|\mathbf{b}_1\||}{|\|\mathbf{b}_2\||}, \) and \( \mathbf{S}_3 = [(-r_3, r_3)], \) for \( r_3 = r\frac{|\|\mathbf{b}_1\||}{|\|\mathbf{b}_3\||}. \)

In summary, two steps are performed at each rasterization step. At first a pixel is selected depending on the sign of the decision variable, which is then updated using the corresponding increment \( \Delta E \) and \( \Delta SE \), respectively. The resulting algorithm is more complicated than its counterparts on the square or hexagonal lattice which is due to the increased flexibility offered by the structure of rank-1 lattices. A pseudo-code example of this algorithm is given in Listing 5.2 for the first circle sector. The remaining sectors can be implemented in the same way, using Table 5.1. The routine \texttt{drawCirclePoints}(u, v, p) plots the Voronoi cell \((u, v)\) by the offset \( p - \mathbf{0} \), such that the circle is centered at the point \( p \).
Table 5.1: Equations for the midpoint \( m \) between the two pixels \( E \) and \( SE \), for \( \Delta_E \), \( \Delta_{SE} \) and \( d_0 \) for all sectors of the circle.
/center p of the circle in lattice coordinates
\* \* lattice basis (b1x, b1y), (b2x, b2y)
*/

void rasterizeCircle(radius, p)
{
    // Init
    alpha = (b1x*b1x) + (b1y*b1y);
    beta = (b1x*b2x) + (b1y*b2y);
    gamma = (b2x*b2x) + (b2y*b2y);
    delta = alpha*radius*radius;
    rad = ifloor(radius);
    rad2 = ifloor(rad*sqrt(alpha)/sqrt(beta));

    // Sector 1
    // start point half sector a
    u = rad; v = 0;
    // init decision function in half sector a
    d = alpha*(rad-0.5)*(rad-0.5) + beta*(2*rad-1) + gamma - delta;

    // start point half sector b
    u1 = 0; v1 = rad2;
    // init decision function in half sector b
    dl = alpha + (2*rad2-1)*beta + (rad2-0.5)*(rad2-0.5)*gamma - delta;

drawCirclePoints(u, v, p);

    while (u-1 > v){
        // half sector a
        if (d < 0){
            d += beta*(2*u-1) + (2*v+3)*gamma;
        }
        else{
            d += (2-2*u)*alpha + (2*u-2*v-5)*beta + (2*v+3)*gamma;
            u -= 1;
        }
        v += 1.0;
        drawCirclePoints(u, v, p);

        // half sector b
        if (dl < 0){
            dl += (2*u1+3)*alpha + (2*v1-1)*beta;
        }
        else{
            dl += (2*u1+3)*alpha + (2*v1-2*u1-5)*beta + (2-2*v1)*gamma;
            v1 -= 1;
        }
        u1 += 1;
        drawCirclePoints(u1, v1, p);
    }

    // ...
}

Listing 5.2: Circle midpoint algorithm on a rank-1 lattice.
6 Textures on Rank-1 Lattices

Textures in computer graphics are generally stored on orthogonal tensor product lattices even though it is known that these lattices are not optimal with respect to sampling efficiency (see Section 2.1.4). The optimal lattice for isotropic image content is a hexagonal lattice, however, the handling of these lattices is difficult, because they are not able to tile a square region and their irrational basis is not optimal for computation. As shown in Section 3.2 rank-1 lattices are able to approximate a hexagonal lattice arbitrarily close while all coordinates of lattice points are rational numbers. We therefore investigate the use of rank-1 lattices to store textures.

6.1 Data Structure

A texture with rank-1 lattice geometry is described by the number of pixels $n$, a generator vector $g$ and the corresponding Minkowski-reduced basis along with the indices of the basis vectors being necessary for address computations. The image data is organized as a linear array $img$ of size $3 \cdot n$ (RGB color model, i.e. red, green and blue), which is addressed by the lattice point index (see Section 2.1.5). In addition the aspect ratio, i.e. the width-to-height ratio $w:h$, has to be specified. If this ratio is 1 : 1 (i.e. square resolution) the lattice and its basis are searched on the unit square, otherwise, by means of a weighted norm (see Section 3.4).

In order to deal with rank-1 lattice textures in a renderer in the following we describe several possibilities for texture look-up and access, the handling of huge textures with respect to tiling and caching and how to acquire image data on the proposed data structure.

6.1.1 Nearest Neighbor Look-Up

The simplest texture look-up strategy uses the color of the rank-1 lattice texel (r1-texel) closest to the texture coordinate. This means to compute the nearest lattice point to $(u,v) \in [0,1)^2$.

For this purpose the texture coordinate $(u,v) \in [0,1)^2$ has to be transformed into the lattice basis, yielding the lattice coordinates $(s,t)$. Then $(d_1,d_2) = [s,t]$ is the anchor point of the lattice cell induced by the fundamental parallelepiped (in lattice coordinates) in which the point $(u,v)$ lies in $[0,1)^2$. The nearest lattice point to $(u,v)$ is contained in the set $M = \{(d_1,d_2),(d_1+1,d_2),(d_1,d_2+1),(d_1+1,d_2+1)\}$ representing the vertices of the lattice cell. Hence it results as the vertex $v \in M$ with the shortest Euclidean distance to $(u,v)$. Note that the distance computation has to be performed in the Cartesian coordinate system.
system, as the Voronoi cell is distorted in the rank-1 lattice coordinate system. Since the lattice coordinates of \( \mathbf{v} \) are known, its index can simply be computed according to the addressing scheme outlined in Section 2.1.5.

An example for a nearest neighbor texture-look up with periodic boundary conditions for rank-1 lattices in Korobov form is given in the next paragraph.

**GLSL Example.** The concept of rank-1 lattice textures can be used in real time rendering using OpenGL or DirectX for example. The textures are then stored as 1d or 2d textures but are accessed as an 1d array. Therefore, the pixel shaders used in the rendering engine have to be extended by the addressing scheme from Section 2.1.5. Listing 6.1 shows an example GLSL code for rank-1 texture access.

```glsl
varying vec4 uv; // 2d texture coordinate
uniform sampler2D r1data; // image on the rank-1 lattice (as 1d array)
uniform vec4 L; // lattice parameters (n,sN,g,d)
uniform vec4 b; // basis vectors for L

void main()
{
    // anchor point of Delaunay cell
    vec2 ft = floor(vec2(uv.x * b.q - uv.y * b.p,
                         uv.y * b.s - uv.x * b.t) * L.w);

    // 4 indices of the 4 corner points
    vec4 idx = mod(vec4(ft.x * b.s + ft.y * b.p,
                      (ft.x + 1.0) * b.s + ft.y * b.p,
                      (ft.x + 1.0) * b.s + (ft.y + 1.0) * b.p,
                      ft.x * b.s + (ft.y + 1.0) * b.p), L.x);

    // corner points in Cartesian coordinates
    vec4 px = idx/L.x;
    vec4 py = mod((idx * L.z) / L.x, 1.0);

    // squared distance to the 2d texture coordinate
    vec4 dp = (px - uv.x) * (px - uv.x) + (py - uv.y) * (py - uv.y);

    // get the index based on the horizontal minimum mm:
    float mm = min(min(dp.x, dp.y), min(dp.z, dp.w));
    float i = (mm==dp.y)?idx.y:
               (mm==dp.z)?idx.z:(mm==dp.w)?idx.w:idx.x);

    // access the 2d image texture as 1d array
    vec2 coord = floor(vec2(mod(i, L.y), i / L.y)) / L.y;
    gl_FragColor = texture2D(r1data, coord);
}
```

Listing 6.1: GLSL implementation of nearest neighbor texture look-up with periodic boundary conditions for Korobov lattices.

### 6.1.2 Linear Interpolation

If the camera comes close to textured objects in a rendering system, they are likely to get quite a jagged appearance, when using the nearest neighbor texture look-up. This is also an important issue in real-time applications where textures are often only available in lower
resolutions. As for the square lattice there are several possibilities regarding texture interpolation. Additionally to the commonly used bilinear interpolation, the geometric structure of rank-1 lattices also implies interpolation on the Delaunay triangles.

**Bilinear Interpolation on Lattice Cells Induced by the Fundamental Parallelepiped:** A bilinear interpolation is performed on the lattice cell \( x_i + \Lambda \) (see Section 2.1.1) in which the texture coordinate lies. The anchor point of the lattice cell (in lattice coordinates) is given by \((d_1, d_2) = \lfloor s, t \rfloor\), with \((s, t)\) representing the lattice coordinates of \((u, v)\). Hence the parallelepiped is spanned by the vertices \{\((d_1, d_2), (d_1 + 1, d_2), (d_1, d_2 + 1), (d_1 + 1, d_2 + 1)\}\) and the colors of the associated r1-texels are interpolated.

**Barycentric Interpolation on Delaunay Triangle:** After the lattice cell \( x_i + \Lambda \) in which \((u, v)\) lies has been determined, the texel colors of both Delaunay triangles are weighted using barycentric interpolation. That way the triangle in which the texture coordinate lies is detected automatically. This means that the texture coordinate lies in that triangle for which the sum of the barycentric coordinates is less or equal to one and the coordinates itself are greater or equal to zero. See Figure 6.2 for a comparison of the proposed interpolation schemes.

### 6.1.3 Rank-1 Lattice B-Splines of Degree \( k \)

Using interpolation filters of higher orders improves the visual quality of the resampled image. However, the traditional B-spline filters cannot be applied directly, as the hexagonal structure of the Voronoi cell will be violated both for radial symmetric and tensor-product B-splines, such that hard edges appear at the interpolated pixels. Therefore we want to construct a family of bi-variate rank-1 lattice splines such that they take into account the shape of the Voronoi cells, i.e. the first order filter (degree 0) exactly yields the Voronoi cell of the rank-1 lattice, and they have a limited size of support, which enlarges with increasing degree. Moreover they are defined to be centered at the origin. Note that the following derivation of the rank-1 lattice B-splines only represents an approximation to the correct solution, which would be similar to the approach in [VBU‘04].

Based on (uniform) B-splines of degree \( k \)

\[
N_{j,k}(t) = \frac{t-j}{k} N_{j,k-1}(t) + \frac{j+k+1-t}{k} N_{j+1,k-1}(t)
\]

\[
N_{j,0} = \begin{cases} 1 & j \leq t < j+1 \\ 0 & \text{otherwise} \end{cases}
\]

with support \([j, j+k+1]\), we design approximate rank-1 lattice splines by a tensor-product-like approach according to the previous properties.

Let w.l.o.g. \( \mathbf{b}_1 \cdot \mathbf{b}_2 > 0 \). As illustrated in Figure 6.1 on the left, the Voronoi cells in a rank-1 lattice can be described by the three vectors \( \frac{1}{2} \mathbf{b}_1, \frac{1}{2} \mathbf{b}_2, \) and \( \frac{1}{2} \mathbf{b}_3 = \frac{1}{2}(\mathbf{b}_2 - \mathbf{b}_1) \) which are orthogonal to the edges of the Voronoi cell. These vectors can also be understood as the symmetry axes of the Voronoi cells. Consequently let \( \mathbf{a}_i = \frac{k+1}{2} \mathbf{b}_i, i = 1, 2, 3, k \geq 0, \) be the
Figure 6.1: Left: Geometry for the r1-B-spline filter kernel. The filter domain for $k = 0$ is highlighted in red, for $k = 1$ in blue and for $k = 2$ in black. Right: Geometry for texture access by a r1-B-spline of degree $k = 1$.

axes of the filter domain of degree $k$, which are displayed for different degrees $k$ in Figure 6.1 on the left along with the associated filter domains. The three axes

$$\text{supp}(a_i) = \left[ -\frac{k+1}{2} b_1, -\frac{k+1}{2} b_i \right], i = 1, 2, 3$$

exactly define the support of a rank-1 lattice filter. The filter weights for a vector $p$ arise from multiplying the scalar projections $p$ onto the axes of the filter domain inserted into the uniform B-spline formula (6.1)

$$h_k(p) = \prod_{i=1}^{3} \frac{N - j + k}{2} \left( \frac{p \cdot a_i}{\|a_i\|} \right)$$

with $l_i = \frac{1}{2} \|b_i\|$. Setting $j = -\frac{k+1}{2}$ in equation (6.1) causes the B-spline to be centered at the origin. The bottom row of Figure 6.2 shows the resulting rank-1 lattice B-splines for $k = 0, 1, 2, 3$.

**Implementation.** In order to apply the family of r1-B-splines for texture access, a r1-B-spline of degree $k$ is anchored in $(u, v)$. Then the colors of the r1-texels in the filter domain are weighted according to the resulting B-spline values. Note that the interpolated color has to be divided by the sum of weights afterwards in order to assure normalization. The rank-1 lattice texels in the B-spline support can be identified in the following way: Compute the nearest lattice point $x$ to $(u, v)$. As the set of lattice points within the filter support is a subset of the points in the support of the B-spline filter anchored at $x$, it suffices to list exactly those lattice points, which lie in the domain $[-k \cdot b_1, -k \cdot b_2] \times [k \cdot b_1, k \cdot b_2]$ and can be enumerated by a simple for-loop. This is visualized in Figure 6.1 on the right. The red hexagon denotes the filter support for $k = 1$ anchored at $(u, v)$, the black hexagon represents the filter support with respect to the nearest lattice point (blue); the blue parallelepiped corresponds to the domain $[-k \cdot b_1, -k \cdot b_2] \times [k \cdot b_1, k \cdot b_2]$. 
Comparison. Figure 6.2 compares close-ups of the different interpolation schemes. The upper row shows the nearest neighbor look-up and the linear interpolation methods. While parallelepipeds become visible in the case of the bilinear interpolation, the Delaunay triangles appear in the barycentric interpolation scheme. These structures only become visible if the textured object is very close to the camera. The lower row illustrates the texture access by the r1-B-splines. In contrast to the linear interpolation they take into account the Voronoi cell structure of the rank-1 lattice texture.

6.1.4 Tiling and Multi-Resolution

In 1990, [Pea90] proposed the textures on demand (TOD) approach, a technique designed to structure and efficiently access large amounts of stored texture data such that the I/O and CPU cost of access is minimized. The key elements of the system are texture tiles (i.e. the texture is tiled in square units adapted to fixed I/O size) and prefiltered textures provided by means of resolution sets.

Due to their periodicity rank-1 lattices are suited very well to tiled texture layout. Tiled
rank-1 lattice resolution sets (rank-1 lattice r-sets) are built following the example of [Pea90] and by taking advantage of the rank-1 lattice structure. The tile size \( n \), i.e. number of lattice points per tile, can be chosen freely, and thus is independent of source image resolution. For example \( n \) can always be chosen as any integer power of 2, as a rank-1 lattice image can be generated for any target resolution with arbitrary width-to-height ratio. This is not always possible for images on square lattices, where the resolution is given as the product of the \( x \)- and \( y \)-resolution. A tiled image on a rank-1 lattice is composed of \( S \times T \) tiles, denoted by \([S,T]_n\). Therefore, the image image corresponds to \( S \cdot T \cdot n \).

The size of the rank-1 lattice r-set is determined by the number of tiles \([S_0,T_0]_n\) in the highest resolution. The resolution of the images in the rank-1 lattice r-set is set to be \([S,T]\) where \( S \) and \( T \) are integers with \( S_0 \geq S \geq 0 \) and \( T_0 \geq T \geq 0 \). One possibility to define a complete rank-1 lattice r-set is to demand the rank-1 lattice r-set to consist of all down-sampled versions of the source image, where the parameters \( S \) and \( T \) are determined by consecutive integer bisection so that there are \([\log(S_0) + 1] \times [\log(T_0) + 1]\) elements in the rank-1 lattice r-set. An example for a complete rank-1 lattice r-set according to this scheme is shown in Figure 6.3 with \([S_0,T_0] = [4,4]\) and \( L_{2048,31,59}^{tile}\). The rank-1 lattice r-set thus consists of the following tiled images \([4,4]_n, [4,2]_n, [4,1]_n, [2,4]_n, [1,4]_n, [2,2]_n, [2,1]_n, [1,2]_n, [1,1]_n\), \( n = 2048\).

If \( S_0 \) and \( T_0 \) are a power of 2 the storage cost for a complete rank-1 lattice r-set can be computed in the same way as in [Pea90] and therefore is at most four times the storage cost of the original image \([S_0,T_0]\).

**Implementation**

The tiled rank-1 lattice r-set is generated from a source image of a resolution of \( xRes \times yRes \) pixels. Additionally, the rank-1 lattice r-set is specified by the number \( n \) of lattice points per tile and the number of (initial) tiles \([S_0,T_0]\) in the highest resolution. In order to compute the lattice parameters \( L_{x,y,g}\), the dimension in pixels of the initial tiles has to be determined by \( xRes/S_0 \times yRes/T_0\), yielding the width-to-height ratio of the search domain. This lattice describes the image tiles and stays the same throughout the rank-1 lattice r-set generation process. For each resolution \([S,T]\) of the rank-1 lattice r-set a tiled rank-1 lattice image has to be computed with the tile dimension in pixels being equal to \( xRes/S \times yRes/T\). This can simply be realized by generating a rank-1 lattice image for each tile shifted over the source image using non-periodic boundary conditions so that the tiles can be put together seamlessly.

The TOD implementation based on rank-1 lattices principally follows the structure of [Pea90]. The only difference consists in the rank-1 lattice r-set member identification, which is performed in two steps. At first we determine the rank-1 lattice r-set members satisfying the width-to-height ratio of the texture region \((s_{width}, t_{width})\) with center \((s,t)\) of which the renderer requests a filtered value. Among those candidates we select the one with \( \frac{1}{nS} \geq s_{width} \cdot t_{width} \), i.e. we choose the resolution for which the pixel area corresponds to the area of the required texture region. Given a suitable member of the rank-1 lattice r-set, the tile number, which is necessary to build the tile key, is obtained by \([s \cdot S], [t \cdot T]\) in order to fulfill the texture pixel demand of the renderer the tile itself is indexed by a nearest.
neighbor rank-1 lattice texture look-up, as detailed in Section 6.1.

6.1.5 Acquisition

Similar to hexagonally sampled images there are two methods to acquire images on rank-1 lattices. The first one is to use specialized hardware for image acquisition, ([MS05] give a survey in the context of hexagonal sensors, also being apparent in medical imaging and remote sensing), but up to now there exists no rank-1 lattice hardware like displays or acquisition devices. The second approach works by using software conversion of regularly acquired images. Converting an image represented in one lattice to another lattice representation is called resampling. The typical procedure is to first reconstruct a continuous image from the original samples by using an interpolation function. This reconstructed image is then resampled with the new lattice. In an implementation reconstruction and resampling are usually combined in a single step.

[MS05] generate hexagonally sampled images according to this interpolation scheme and compare nearest neighbor, bilinear and bicubic interpolation. They conclude that a bilinear filter kernel is sufficient for the purpose of generating hexagonally sampled images.
when the source material has a similar resolution to the target lattice. [VWPL02] propose an algorithm for image resampling between orthogonal and hexagonal images with similar sampling resolutions, which is based on the least squares approach, thus minimizing loss of information by incorporating a spline transform. [CVFH08] present a new grid conversion approach in order to resample between hexagonal and Cartesian lattices with the same sampling density. This method allows for recovering the initial data exactly, when the same algorithm is applied for the converse operation.

To compute rank-1 lattice textures for the experiments in the results section we have chosen the very simple generation approach of downsampling a high resolution source image to a rank-1 lattice image of maximal $\frac{1}{16}$ of the source image resolution. This corresponds to the “box-filter” interpolation and therefore facilitates the comparison of rank-1 to square lattice textures, since both kinds of textures are computed in quite a similar way. For other applications the resampling algorithms summarized at the beginning of this section can be applied to generate rank-1 lattice textures.

6.2 Results

In Section 6.1 the data structure for textures on rank-1 lattices is described and particularly classic access mechanisms are established, as shown in Figure 6.2. We implemented textures on rank-1 lattices in a ray tracing system by taking into account the anisotropy of the source images. This means that depending on the original texture either a MMD or an anisotropic rank-1 lattice is chosen. In the following we visually compare both kinds of rank-1 lattice texture to textures using square pixels and explain how to determine the anisotropy for the lattice choice.

6.2.1 Maximized Minimum Distance Rank-1 Lattices

Due to their isotropic structure maximized minimum distance lattices lend themselves to represent isotropic textures, which are thus called MMD rank-1 lattice textures. We implemented such textures in a ray tracing system. To verify the advantage of these textures compared to square lattices we chose four different textures at different resolutions using nearest neighbor sampling. The rendering system creates high dynamic range images and the comparison computes the mean square error between the reference image and the downsampled texture images. The reference image was computed using 1024 random samples per pixel at a resolution of $512 \times 512$ pixels (using a high resolution texture on the square lattice). The comparison images were computed with 128 random samples per pixel. Since the comparison images use fewer samples, there is a constant error in each image that can be seen in the graphs.

The resolution of the source image was $2048 \times 2048$ and we vary the target resolution from $128 \times 128$ to $512 \times 512$ in steps of $16 \times 16$. The target texture is always a downsampled version of the source image using a simple box filter in both cases. The rank-1 lattice texture contains exactly the same number of points as the square lattice texture. Example textures are shown in Figure 6.4 for a number of $64 \times 64$ picture elements. Figure 6.5
Figure 6.4: Comparison of standard textures with rank-1 lattice textures where each image contains the same number of picture elements (64 × 64). Rank-1 lattices can very closely approximate the hexagonal lattice. This results, among other things, in an advantage for non axis-parallel lines as can be seen clearly in the middle image.

shows the results of the comparison. Except for the checker board texture, rank-1 lattice textures consistently outperform the square lattice textures. The reason for the odd graph in the checker board scene is the axis aligned structure of the source texture. When the target resolution is a divider of the original checker board size we get a perfectly downsampled texture and only the constant error due to fewer pixel samples remains.

The addressing on rank-1 lattices is more expensive than the addressing of square lattice textures. In our rendering system the performance drop was about 5% for all tested scenes. Since these scenes contain only simple geometry and lighting calculations the relative performance cost drops when a more complex scene is rendered.

6.2.2 Anisotropic Rank-1 Lattices

If an image, or more specifically a texture, exposes anisotropy, anisotropic rank-1 lattices can be used to further improve the visual appearance, i.e. the approximation quality. This is illustrated in Figure 6.6 for a wood grain texture, which exposes one main direction with large variance. Here maximized minimum distance lattices lose details in comparison to the anisotropic ones.

The parameters for determining the anisotropic rank-1 lattice are computed from the structure tensor (see for example [Jäh02]) of each pixel. Without loss of generality let \( \lambda_{1,i} > \lambda_{2,i} \) be the eigenvalues of the structure tensor and \( \mathbf{v}_{1,i} \) and \( \mathbf{v}_{2,i} \) the corresponding eigenvectors for each pixel \( i \in [0, x_{\text{Res}} \cdot y_{\text{Res}}) \). Then the main direction \( \mathbf{d} = \frac{1}{x_{\text{Res}} \cdot y_{\text{Res}}} \sum_{i=0}^{x_{\text{Res}} \cdot y_{\text{Res}}-1} \mathbf{v}_{1,i} \)
Figure 6.5: Results of the comparison of square lattice textures to rank-1 lattice textures. The top rows show the reference images and the bottom rows the scaled mean square error at different resolutions.
Figure 6.6: Magnifications of the highlighted squares in the texture on the left represented on the regular grid, MMD rank-1 lattice, and anisotropic rank-1 lattice by 16384 pixels each. Note that for the anisotropic rank-1 lattice the mean square error (MSE) to the high resolution reference on the right is about half of the regular and MMD rank-1 lattice.

is computed by averaging the eigenvector of the largest eigenvalue. The width

\[ w = 1.0 - \frac{1}{A_{\text{max}}} \cdot \sum_{i=0}^{xRes \times yRes - 1} \frac{\lambda_{1,i}}{\lambda_{2,i}} \]

subtracts the normalized texture anisotropy from 1, since 0 means maximum anisotropy for the search algorithm from Section 3.3. The normalization constant \( A_{\text{max}} \) must be determined experimentally for a set of textures.

In Figure 6.7 isotropic rank-1 lattice textures are compared to anisotropic ones by means of the \( L_2 \)-error of the test images to a reference solution for an increasing number of lattice points for the source image of Figure 6.6. As can be seen from the error graph, the anisotropic rank-1 lattice textures are superior, as they are able to capture even small details, which are lost in the isotropic case.
Figure 6.7: Comparison of MMD and anisotropic rank-1 lattice textures for the wood grain example.
7 Anti-Aliasing by Rank-1 Lattices

The mathematical task of image synthesis consists in computing the intensity $I(k, l)$ for each pixel on the display medium, where $(k, l)$ represents the pixel location and additional dependencies like time or wavelength are possible.

Typically, an image is represented as a two dimensional arrangement of pixels. Then the pixel color is determined by projecting the image function onto the display basis:

$$ I(k, l) := \frac{1}{|P_{k,l}|} \int_{P_{k,l}} f(x) \, dx. $$

(7.1)

So the intensity $I(k, l)$ results as the average radiance flux through the corresponding pixel by computing an integral over the pixel area, where $x$ in (7.1) represents a location inside the pixel $P_{k,l}$. This means to perform pixel-wise integration.

Due to the correlation between the pixels an image also can be considered as a functional, which means to solve an integro-approximation problem

$$ I(k, l) := \int_{P} f(x, k, l) \, dx \approx \frac{1}{n} \sum_{i=0}^{n-1} f(x_i, k, l). $$

(7.2)

Here the two-dimensional image function $I(k, l)$ is given by a parametric integral. Thinking of $(k, l)$ as pixel coordinates on the screen, the above algorithm simultaneously computes a color for each pixel on the screen by averaging samples of the integrand at positions $x_i$ [Kel01].

Since usually analytic solutions are hardly available due to unpredictable discontinuities, we are interested in efficient numerical schemes to approximate the image function. However, the curse of dimension and a lack of smoothness of the integrands usually cause the standard tensor product quadrature to fail. A suitable alternative is given by numerical integration using Monte Carlo and quasi-Monte Carlo methods (see Section 1.1), which means that the integrals are estimated by sampling and averaging. As in practice we can only use a finite amount of samples, Shannon’s sampling theorem requires the image function to be band-limited under the assumption of being sufficiently continuous. This condition hardly ever can be fulfilled, though, since for example a simple triangle edge already introduces infinitely high frequencies. This means that for an increasing number of samples aliasing artifacts will reappear [Kel06], which also applies to random sampling. Particularly, this implicates that for an increasing number of samples random sampling cannot hide aliasing in noise anymore, such that we can sample deterministically, especially using rank-1 lattices, just as well.
Figure 7.1: Infinite checker board rendered with 16 samples per pixel. The left image uses the same MMD rank-1 lattice $L_{16,(1,4)}$ for all pixels, while in the right image an anisotropic rank-1 lattice adapted to the spectrum of each pixel is used. Clearly some aliases below the horizon become much more attenuated.

So what can we do when using a finite number of sampling points? In the following we present a new sampling technique for rank-1 lattices, in which the sampling pattern is adapted to the spectrum of the image function. Then we explore a new stratification approach for jittered sampling based on rank-1 lattices.

### 7.1 Spectrally Adaptive Sampling

In [GZD08] Goldberg et. al. present a technique for spectral noise synthesis in which they construct band-limited noise images according to a desired spectrum width and orientation. Whereas this method has applications in fast rendering of procedural noise textures for interactive rendering, we introduce a sampling scheme for anti-aliasing which is based on adapting the quadrature rule to the spectrum of the integrand, such that it is filtered in a better way. For this purpose we use anisotropic rank-1 lattices which are searched according to the spectrum of the integrand in order to reduce aliasing artifacts. The improved anti-aliasing is illustrated by a comparison in Figure 7.1 for the example of rendering an infinitely expanded checker board. Due to perspective projection and as many light and dark gray checker board cells fall within a single pixel, aliasing artifacts appear especially near the horizon. According to Figure 7.2, which visualizes the Fourier transform of some pixels near the horizon, there is more variance in the vertical than in the horizontal direction, which cannot be captured sufficiently by an isotropic sampling pattern.

Given the algorithm from Section 3.3, an anisotropic rank-1 lattice is specified by the main direction $d$ and the width $w$. Due to the spectrum of the checker board below the horizon, as displayed in Figure 7.2, we globally assume maximum anisotropy by fixing $w = 0$. The main direction $d$ is determined by projecting the normal of the first object intersected by a ray through the center of a pixel onto the image plane and normalizing the resulting vector.

This way the samples from the anisotropic rank-1 lattice in the pixel become isotropic and more uniform, when projected onto the surface seen in the scene (see Figure 7.3 on the left). As a consequence the texture is averaged more efficiently resulting in reduced aliasing. Note that for this argument, we assumed only one plane perpendicular to the
normal seen through a pixel, which is a useful approximation in many cases.

As the perspective projection does not have an impact on the variance of the checker board until a certain distance from the camera, anisotropic rank-1 lattices are used only for those pixels for which the hit point of a ray through a pixel midpoint and the checker board exceeds a certain distance to the camera (which is determined experimentally for this special setting). Otherwise isotropic MMD rank-1 lattices are used for each pixel.

In Figure 7.3 on the right, we compared the anisotropic rank-1 lattices to MMD rank-1 lattices, jittered grid and Larcher-Pillichshammer points by computing the $L_2$-norm of a converged reference image to the corresponding test images for an increasing number of sampling points per pixel. Note that both axes in the error graph are scaled logarithmically and that the reference image was computed by applying a jittered grid sampling pattern with $1024 \times 1024$ samples at each pixel. We observe that using the anisotropic rank-1 lattice outperforms the other sampling patterns especially for lower sampling rates. In contrast to the MMD rank-1 lattices, the error curve of the anisotropic lattices does not expose a strong oscillation any more. However, for larger numbers of samples the Larcher-Pillichshammer points still show the fastest convergence. We speculate that this is due to fixing the width $w$.

### 7.2 Jittering Rank-1 Lattices

In Monte Carlo integration stratification is used as a method for variance reduction, which has been shown to be never worse, but not necessarily better than pure Monte Carlo either [Nie92, Sob94]. The integration domain is partitioned into $k$ disjoint subdomains, i.e. strata,
Figure 7.3: Left: The arrows indicate the pixels and directions for which anisotropic rank-1 lattices are used. Right: Comparison of the anisotropic rank-1 lattices, to MMD rank-1 lattices, jittered grid and Larcher-Pillichshammer points.

\[ \int_{p} f(x) dx = \sum_{j=0}^{k-1} \int_{A_j} f(x) dx \approx \sum_{j=0}^{k-1} \lambda_s(A_j) \frac{n_j}{n} \sum_{i=0}^{n_j-1} f(x_{i,j}), \quad (7.3) \]

where \( \lambda_s(A_j) \) is the Lebesgue-measure of \( A_j \) and \( n = \sum_{j=1}^{k} n_j \) is the total number of samples. A variance reduction is achieved for choosing \( n_j = \lambda_s(A_j)n \), \( 1 \leq j \leq k \) \cite{Nie92}.

In the case of jittered sampling the stratification is done for each axis separately by dividing each axis into \( n_l \) intervals for a total number of \( n = \prod_{s=1}^{l} n_l \) samples. This means that the domain is divided into \( n \) equally sized subdomains and one random sample is drawn inside each subdomain. Then according to equation (7.3) \( n_j = 1 \) and \( \lambda_s = \frac{1}{n} \) and we have

\[ \int_{p} f(x) dx = \sum_{j=0}^{n-1} \int_{A_j} f(x) dx \approx \frac{1}{n} \sum_{j=0}^{n-1} f(x_{A_j}), \quad (7.4) \]

where \( x_{A_j} \) lies in the \( j \)-th stratum \( A_j \).

The method of stratification has been shown to increase the uniformity of random samples and to improve the convergence rate \cite{Nie92, Sob94}. However, since \( n \) must be factorized for stratified sampling, not every choice of \( n \) results in a good division of the unit cube in all dimensions (e.g. for \( n \) being prime). Therefore a good pattern is not not available for every number of points. Moreover, the samples can still lie arbitrarily close together along strata boundaries, preventing a large minimum distance of the sampling points.

Jittered grid sampling was proposed in computer graphics by Cook \cite{CPC84, Coo86}. In this context Mitchell \cite{Mit96} studied the effects of stratification on the convergence to the mean value of image pixels and observed an improved convergence, since even in the worst case the results are not better, but not worse than random sampling either. Chiu et al. presented a method to improve the two-dimensional jittered grid sampling by combining
the properties of latin hypercube (LHS) and jittered grid sampling, which is called multi-jittered sampling [CSW94]. In the case of \( n \times m \) strata each stratum is again subdivided into \( N \times M \) subdomains, such that the samples on this refined grid satisfy the LHS conditions and the pattern fulfills the jittered grid property for the \( n \times m \) strata at the same time. Thus the uniformity of the sampling points is increased, a maximized minimum distance property can not be guaranteed, however. In [Kel04], Keller studied jitted sampling by rank-1 lattices, where the domain discretization relies on the fundamental parallelepiped of the rank-1 lattice. As no minimum distance property is assured either, Keller proposes correlated sampling in order to improve convergence, where for each stratum in one pixel the same random offset is used for the sampling point. Based on the stratification approach presented in that work, we examine a different scheme via the Voronoi diagram of the rank-1 lattice. We evaluate the resulting sampling pattern by performing a spectral analysis.

**Spectral Analysis of Sampling Patterns**

In order to analyze the spectral properties of the jitted rank-1 lattice point sets, we examine the frequency domain characteristics of the sample pattern according to the technique described in [Uli87, LD08, MF92]. Thereby, the same normalization as in [LD08] is used in order to enable a comparison of the results. The analysis relies on two one-dimensional graphs, namely the radial power and the radial anisotropy, which are obtained by radial statistics. The radial power determines the power at each frequency, whereas the radial anisotropy represents a measure for the radial symmetry. Both graphs are based on evaluating mean periodograms [Bar55].

For real signals the periodogram of a point set \( \{x_0, \ldots, x_{n-1}\} \) results as the square of the magnitude of the Fourier transform of the point set

\[
\left| \frac{1}{n} \sum_{i=0}^{n-1} \delta(x - x_i) \right|^2,
\]

where \( \mathcal{F} \) denotes the Fourier transform and \( \delta \) represents Dirac’s delta function. The mean periodogram, i.e. the estimated power spectrum \( \hat{P} \), is obtained by averaging \( K \) periodograms. Note that periodograms are computed by means of discrete periodic sample images, which are defined over the unit square. Each sample is represented as an impulse of value 1 and finite width, the location of which is rounded to the nearest grid location. Thus it is possible to evaluate periodograms by the discrete FT and the FFT respectively.

In order to compute the radial power and the anisotropy the frequency domain is partitioned into a set of concentric annuli of width \( \Delta \), as displayed in Figure 7.4. Each annulus is defined by a central frequency \( f_r \) and covers \( N_r(f_r) \) frequency samples [LD08]. Then the radially averaged power spectrum

\[
P_r(f_r) = \frac{1}{N_r(f_r)} \sum_{i=1}^{N_r(f_r)} \hat{P}(f_i)
\]

results as the mean value of \( \hat{P}(f) \) within each annulus. The variance of the frequency
Figure 7.4: Visualization of the annuli which are used to average the power spectrum radially.

\[ s^2_r(f_r) = \frac{1}{N_r(f_r) - 1} \sum_{i=1}^{N_r(f_r)} (\hat{P}(f_i) - P_r(f_i))^2 \]  

(7.7)

is used to compute the radial anisotropy

\[ A_r(f_r) = \frac{s^2_r(f_r)}{P_r^2(f_r)}. \]  

(7.8)

These measures are based on the fact that the periodogram of a blue noise pattern, as for example the Poisson disc distribution, is radially symmetric. Such sampling patterns are characterized by a central DC peak which is surrounded by an annulus of low energy such that low frequencies are attenuated. This is illustrated in Figure 7.5 (a) for a blue noise sampling pattern of 16384 points obtained by applying Lloyd’s relaxation method to random samples [HDK01] using 16 iterations and in Figure 7.5 (b) for a sampling method based on polyominoes [Ost07] which allows for fast importance sampling with blue noise properties. However, due to construction (see C++ code referenced in the paper [Ost07]) the spectral analysis is only performed for \( K = 1 \). Accordingly, the typical power spectrum, the radially averaged power spectrum and the radial anisotropy of a jittered grid sampling pattern are shown in Figure 7.5 (c). The point sets on the unit square are plotted for \( n = 512 \) each, whereas \( n = 16384 \) points are used in order to compute the power spectrum.

Corresponding to the work of [Uli87, LD08] we use \( K = 10 \) periodograms in order to compute \( \hat{P}(f) \). This means that an anisotropy of \(-10dB\) implies background noise and suggests the sampling pattern to have a good radial symmetry. In the case of the polyominoes (see Figure 7.5 (b)) 0dB correspond to background noise, since \( K = 1 \). As in [LD08] the spectral analysis is performed for a periodogram resolution of \( 1024 \times 1024 \) and distributions of 16384 sampling points. The width of an annulus is chosen as the length of the vector plotted in Figure 7.4, divided by the number of annuli, namely \( \Delta = \frac{1}{2} \sqrt{xRes^2 + yRes^2} / \text{num} = \frac{1}{2} \sqrt{1024^2 + 1024^2} / 512 \approx 1.4 \), which corresponds to approximately one frequency sample. As the anisotropy may vary over a wide range, the graphs are plotted in decibels, according to the relation \( x_{db} = 10 \log_{10}(x) \). Moreover, as in [LD08]
Figure 7.5: Spectral analysis for several sampling pattern. First column: Pattern for \( n = 512 \). Second column: Power spectrum for \( n = 16384 \). Third column: Radially averaged power spectrum. Fourth column: Radial anisotropy.
the radially averaged power spectrum is normalized such that the mean value equals

$$\frac{1}{N_r(f_r)} \sum_{i=1}^{N_r(f_r)} P_r(f_i) = 1$$

and the power spectra are plotted without the DC peak.

### 7.2.1 Mapping in an Isotropic Way

In [Kel04], the domain stratification is induced by the fundamental parallelepiped of the rank-1 lattice which is spanned by the basis vectors $B = (b_1, \ldots, b_s)$ and the origin. The strata are defined as a partition of the unit cube by the $n$ parallelepipeds anchored at each lattice point

$$R_i : I^s \rightarrow A_i$$

$$\mathbf{x} \mapsto \left( \frac{i}{n} \cdot \mathbf{g} + B\mathbf{x} \right) \mod I^s,$$

(7.9)

This mapping is called reduced Cranley-Patterson rotation, an example of which is shown in Figure 7.5 (d) for the lattice $L_{512,(1,200)}$. However, the spectral analysis, which has been performed for $n = 16384$ points, reveals the anisotropy of the power spectrum reflecting the sheared strata.

In our approach we directly take advantage of the isotropic structure of MMD rank-1 lattices. For that purpose the strata are induced by the Voronoi diagram of the rank-1 lattice, i.e. we define the stratum $A_i$ as the Voronoi cell at the corresponding lattice point. However, as in [Kel04] the samples might still get arbitrarily close to each other. This means that the minimum distance property of the lattice is destroyed. A simple solution consists in placing the sampling points within the central area of a stratum [Ost07]. We determine the central area of the fundamental Voronoi cell as a circle of certain radius

$$r_L(k) = \frac{d_{\text{min}}(L_n, k)}{k}, \quad k \in \mathbb{N}$$

(7.10)

depending on the minimum distance of the lattice. Note that $r_L(2)$ corresponds to the in-circle radius of the fundamental Voronoi cell. Hence, it is reasonable to choose $k \geq 2$.

This domain discretization can be formalized by

$$R_i : I^s \rightarrow A_i$$

$$\mathbf{x} \mapsto \left( \frac{i}{n} \cdot \mathbf{g} + r_L(k) \cdot H(\mathbf{x}) \right) \mod I^s$$

(7.11)

where $H(\mathbf{x})$ represents a mapping of an $s$-dimensional point $\mathbf{x} \in I^s$ into the unit hypersphere.

In two dimensions

$$H(\mathbf{x}) = r \begin{pmatrix} \cos(2\pi x_2) \\ \sin(2\pi x_2) \end{pmatrix}, \quad \text{for } r = \sqrt{x_1}, \quad \mathbf{x} = (x_1, x_2)^	op$$

(7.12)
yields a point in the unit disk. For this mapping we have $\cup_i A_i \subset I'$ instead of $\cup_i A_i = I'$. This can be resolved by using a Cranley-Patterson rotation [CP76] for each instance of the sampling pattern.

Applying the discretization defined by (7.11) and for (7.4) we get

$$\int_{I'} f(x)dx \approx \frac{1}{n} \sum_{i=0}^{n-1} f(R_i(\vartheta_i) + \zeta \mod 1)$$

(7.13)

Thereby one independent uniform random sample $\vartheta_i \in I'$ is used for each $R_i$ and the pattern is shifted randomly on the unit torus by $\zeta \in I'$. The new jittered sampling scheme based on (7.11) is illustrated in Figure 7.6 for several $k$ and $s = 2$. Applying a random shift (see equation 2.22) to any fixed point set enables an unbiased integral estimation. Therefore equation 7.13 yields an unbiased estimator which can be used for any square integrable function [Kel04]. Similar to jittered grid sampling the replication of several random instances results in the same white noise and the new sampling scheme is not worse but not necessarily better than random sampling.

![Figure 7.6: Jittered sampling by rank-1 lattices based on the domain discretization defined in (7.11).](image)

**Adjusting the Radius**

Now we consider two-dimensional jittered sampling for pixel anti-aliasing. For that purpose we run experiments on the size of the domains $R_i$ defined by (7.11) and on the distribution of the random samples $\vartheta_j \in I'$. The size of $R_i$ is determined by the radius $r_L(k), k \in \mathbb{N}$, which therefore impacts the minimum distance and thus influences the spectral properties of the resulting point set. Figure 7.7 displays jittered rank-1 lattices $L_{512, (1.200)}$ for $k$ chosen from the set $\{1, 2, 3, 4, 10\}$. The spectral analysis is performed for $n = 16384$ points, as indicated in Section 7.2. The smaller $k$, the more a random pattern is approximated, the larger $k$ the more the resulting pattern resembles a rank-1 lattice, since $d_{\min}(L_{n,g})/k \to 0$ for $k \to \infty$, i.e. by means of $r_L(k)$ it is possible to “interpolate” between the rank-1 lattice and the random pattern. Altogether $r_L(3) = d_{\min}(L_{n,g})/3$ seems a good compromise between the dual lattice, i.e. the Fourier transform of the rank-1 lattice (see Section 3.3), and noise appearing in the spectrum of the sampling pattern.
Due to the construction the new discretization guarantees that $d_{\min}(P_n) \geq \frac{d_{\min}(L_{\Delta n})}{k}(k-2)$, for $k \geq 2$. In order to further increase the minimum distance of the stratified lattice points over the domain discretization we additionally performed experiments on the sample distribution. For example instead of uniformly sampling the disc of radius $r_{L}(k)$ the samples can be forced to concentrate near their anchor points $\frac{I}{n} \cdot g$ by changing the mapping function $H(x)$ of equation (7.12) to

$$H_p(x) = r^p \cdot \left( \frac{\cos(2\pi x_2)}{\sin(2\pi x_2)} \right), \quad \text{for } r = \sqrt{x_1}. \quad (7.14)$$

This means that $R_i$ (see equation 7.11) now depends on the two parameters $k$ and $p$. Figure 7.9 illustrates the resulting pattern for $r_{L}(3)$ and $p \in \{1, 2, 3, 4, 5\}$. The sampling points fall nearer to the original lattice points, as can be seen from the radial power graphs and the spectra become more anisotropic.

Another possibility to increase the minimum distance consists in sampling the disc of radius $r_{L}(k)$ by a normal distribution of mean $\mu = 0$ and variance $\sigma = \frac{1}{3}$, using the Box-Müller transform [BM58]:

$$G(x) = \frac{r}{3} \cdot \left( \frac{\cos(2\pi x_2)}{\sin(2\pi x_2)} \right), \quad \text{for } r = \sqrt{-2\ln(1-x_1)}. \quad (7.15)$$

As the interval of $[-3\sigma, 3\sigma]$ covers 99.7% of the energy of the density function, for $\sigma = \frac{1}{3}$ the jittered samples lie in disc of radius $r_{L}(k)$ with this probability. Note that we do not discard samples falling outside the disc. Figure 7.8 shows the corresponding sampling pattern and the associated spectral analysis. In comparison to Figure 7.7 we obtain a larger minimum distance at approximately equal anisotropy.

**Approximating Blue Noise Properties**

Using Lloyd’s relaxation method a minimum distance and thus blue noise properties can be imposed on a sampling pattern [MF92, HA90, HDK01]. As rank-1 lattices are invariant under Lloyd’s relaxation method this method forces the jittered samples back to the original lattice points underlying the stratification, when applied to the new jittering method. Therefore we perform only one relaxation step after jittering, and explore the spectra for different $k$ again. As before, the bigger $k$, i.e. the radius $r_{L}(k)$, the more the dual lattice reappears in the spectrum. Moreover the area of low frequency around the DC peak takes a hexagonal form, as well as the ringing appearing for increasing $k$. Thus a blue noise spectrum is approximated. For $k \to \infty$ the hexagonal structure resolves into the dual lattice, which is indicated by the spikes in the radial power and anisotropy graphs.

**7.2.2 Results**

In Figure 7.11 we compare the new jittered rank-1 lattice schemes using one instance of a Cranley-Patterson rotation to jittered grid sampling, MMD rank-1 lattices and the reduced Cranley Patterson rotation by means of integro-approximation for the zone plate test pattern.
Figure 7.7: $L_{512, (1,200)}$, using the mapping $H(x)$ (7.12) in the domain stratification (7.11).
Figure 7.8: \( L_{512, (1, 200)} \), using the mapping \( G(x) \) (7.15) in (7.11).
Figure 7.9: $L_{512, (1, 200)}$, using the mapping $H_p(x)$ (7.14) in (7.11).
Figure 7.10: $L_{512(1,200)}$, One Lloyd relaxation step after jittering by mapping $H(x)$ (7.12) in (7.11).
$\sin(x^2 + y^2)$, i.e. the sampling points are applied over the whole screen (see equation (7.2)). A B-spline filter of degree 2 is used for reconstruction. For each pattern we determined the $L_2$-error in decibel to a reference image which was computed by applying a jittered grid sampling pattern with $1024 \times 1024$ samples at each pixel.

The MMD rank-1 lattice results in a smooth image with the smallest error to the reference image, however exposing some aliasing artifacts due to correlation. The jittered grid and reduced Cranley-Patterson rotation hide aliasing by noise, but results in a larger error. Using the mapping $G(x)$ for stratification in our new sampling scheme outperforms the mapping $H(x)$ for the same radius $r_L(3)$ and yields approximately the same (visual and error) results as using the mapping $H_5(x)$. Applying the Lloyd relaxation almost achieves the same quality as in (b) with respect to the error, while hiding the aliasing artifacts in noise at the same time. Note that although both the visual quality and error is improved with regard to (a), some aliasing artifacts in the upper right corner remain and will become more noticeable with larger $k$ in $r_L(k)$. Although the sampling pattern based on polyominoes results in a slightly lower error than (h) the visual quality suffers from the regular structure.

In summary, the jittered rank-1 lattices outperform the classical jittered grid techniques by reducing the error and increasing the visual quality. The generation of the new sampling pattern can be performed in a very fast and simple way and contrary to other methods the stratified patterns are available for any number of points. When additionally applying one Lloyd relaxation step, blue noise properties are approximated and the resulting sampling patterns yield results comparable to the polyomino sampling scheme which represents a state of the art blue noise technique.

Please note that jittering in the central circular area of the strata of the jittered grid sampling pattern [Ost07](d) also causes both a visual improvement and reduces the $L_2$-error, but does not solve the problem that good patterns cannot be guaranteed for any number of points, as mentioned at the beginning of this section.
Figure 7.11: Zone plate $\sin(x^2 + y^2)$ for $512 \times 512$ sample points in a resolution of $512 \times 512$, filtered with a B-spline filter of degree 2. The error to the reference image is given in decibel.
In this thesis we have examined rank-1 lattices within the scope of computer graphics. For that purpose we have first provided and, where necessary, extended the theoretical background, concerning addressing, administration, and search of such lattices and lattice sequences. Concentrating on low dimensional applications in computer graphics, we have both developed an efficient search algorithm for two-dimensional isotropic, i.e. maximized minimum distance rank-1 lattices, which is mathematically motivated by an error bound for the numerical integration of Lipschitz-continuous functions, and have introduced the concept of and a search method for anisotropic rank-1 lattices. The theoretical results have then been applied in various areas of computer graphics, ranging from simulation algorithms which rely on the fast Fourier transform, to image and texture representation and anti-aliasing:

- Independent of dimension \( s \) rank-1 lattices require only a one-dimensional Fourier transform. Therefore spectral synthesis and simulation on rank-1 lattices can be implemented efficiently. Additionally, the approximation can be more accurate than on a tensor product lattice. It is also notable that the isotropic measure of maximized minimum distance can replace the classical measures like e.g. discrepancy often used in connection with rank-1 lattices. For isotropic spectra this measure provides best visual quality, too.

- Thanks to a higher sampling efficiency images represented on rank-1 lattices provide a better image quality than images on the square lattice at the same storage cost. In fact the sampling efficiency of maximized minimum distance rank-1 lattices very closely approximates the one of the hexagonal lattice, such that almost no visual difference is noticeable. As rank-1 lattices are not restricted to the isotropic spectrum, they are also able to capture anisotropic structures. We have taken advantage of this fact in order to enable improved image representation by anisotropic textures.

Image and texture representation by rank-1 lattices can be easily integrated in ray tracing and real-time rendering systems. The periodicity of rank-1 lattices allows for a simple tiled image representation, with the number \( n \) of lattice points per tile being independent of the original image resolution. Therefore \( n \) always can be chosen freely, according to the need of the application, as a prime number or a power of 2 for example.

Using rank-1 lattices for the pixel layout makes it necessary to adjust the classical rasterization algorithms to this kind of lattice. We have extended the work of [Ski05]
to Voronoi cells and we have proposed a method to perform the midpoint algorithm on rank-1 lattices.

- In the context of anti-aliasing we have explored two new approaches using rank-1 lattices. By adapting the sampling pattern to the spectrum of the integrand by means of anisotropic rank-1 lattices, MMD rank-1 lattices are outperformed for anisotropic structures especially for lower sampling rates. Moreover, we have presented a new jittering method in order to hide aliasing which restricts the domain discretization to circles enclosed by those Voronoi cells. Thus it is possible to vary the resulting sampling pattern from nearly random to strictly deterministic rank-1 lattices in a very simple way.

8.1 Conclusion

Rank-1 lattices can be generated in a very simple and fast way. As parameters can be found both for isotropic and anisotropic spectra and as they are not restricted to the unit square due to weighted norms, they can be constructed and applied in a very flexible way. Thus, images can be represented for any number of pixels on rectangular, square and parallelepiped-shaped domains, for example. Further convex shapes are possible. However, this flexibility results in a more complicated addressing scheme, neighborhood computations and rasterization algorithms. Due to the lack of explicit constructions the generator vector has to be determined by computer search, which still is performed in an exhaustive way in the anisotropic case and thus makes the search process very slow.

In the setting of anti-aliasing the regular structure of rank-1 lattices enforces correlations between the function to be displayed, which results in distorting artifacts. On the other hand, these lattices have been proven to improve the image quality, when they are applied carefully, as for example in the case of spectrally adaptive and jittered sampling.

8.2 Future Work

On the theoretical side future work will concentrate on accelerating the search of generator vectors for the anisotropic rank-1 lattices and on extending the low dimensional search principles to higher dimensions in a component-by-component manner [NC06]. So far, MMD rank-1 lattice sequences have only been explored with respect to search algorithms. Possible applications comprise adaptive anti-aliasing, image pyramids and a hierarchical approach of the jittered rank-1 lattice sampling scheme. As the spectrally adaptive sampling has only been tested in the simple checker board scene we would like to extend this sampling technique to general scenes in rendering.

Within the framework of texture representation we have proposed a fast construction for rank-1 lattice B-splines, such that the structure of the Voronoi cells of rank-1 lattices is preserved. These B-splines can be used for texture resampling and access. Since our rank-1 lattice B-spline definition only represents an approximation to the correct solution, we intend to construct rank-1 lattice splines in analogy to the hex-splines approach [VBU+04]
as future work. It would be interesting to compare the approximation presented in this chapter to the mathematically correct B-splines with respect to quality and computational complexity.

Further issues of future work include to study texture synthesis based on Fourier transform (which can be realized in shaders for example), to implement image processing algorithms, such as edge detection, using the framework of images on rank-1 lattices, and to build a prototype rank-1 lattice display device.
This chapter contains the Python source code for the search algorithms of Chapter 3.

A.1 Search Algorithms

Exhaustive Search

```python
def searchMMDR1Lattice(n):
    maxmin = 0
    gen_opt = []

    for gx in range(1, n):
        for gy in range(1, n):
            if (computeGCD(gx, gy, n) != 1):
                continue
            mindist = gaussianBasisReduction(n, gx, gy)
            if (mindist == maxmin):
                gen_opt.append([gx, gy])
            elif (mindist > maxmin):
                maxmin = mindist
                gen_opt = []
                gen_opt.append([gx, gy])

    return maxmin, gen_opt
```

Listing A.1: Exhaustive search for maximized minimum distance rank-1 lattices.

Approximate Search for MMD Rank-1 Lattices

```python
def searchLatticeApproxUsingGR(n):
    l = math.sqrt(math.sqrt(4.0/3.0) * float(n))
    maxmin = 0
    gen_opt = []
    gen_vec = []
    gen_vec = rasterizeCircle(l)

    for e in gen_vec:
        a, b = e

        if (computeGCD(computeGCD(n, a), b) != 1):
            continue
```

# take only valid generator vectors
```
mindist = gaussianBasisReduction(n, a, b)

if (mindist == maxmin):
    gen_opt.append([a, b])
else (mindist > maxmin):
    maxmin = mindist
    gen_opt = []
    gen_opt.append([a, b])

return maxmin, gen_opt

Listing A.2: Searching for maximized minimum distance rank-1 lattices by restricting the search space.

Searching for MMD Rank-1 Lattice Sequences

def searchMMDR1LatticeSequenceInitLat(n, gx, gy, m_max, b):
    maxmin = 0
    gen_opt = []  # saves parameter (i, j)
    # build valid generator vectors
    for i in range(0, n*m_max):
        for j in range(0, n*m_max-i):
            xTmp = gx + i*n
            yTmp = gy + j*n
            sumdist = 0
            md = []
            for m in range(1, m_max+1):
                nTmp = (b ** m) + n
                mindist = gaussianBasisReduction(nTmp, xTmp, yTmp)
                sumdist += float(mindist)/nTmp
                md.append(mindist)
            if (sumdist == maxmin):
                gen_opt.append([i, j, md])
            elif (sumdist > maxmin):
                maxmin = sumdist
                gen_opt = []
                gen_opt.append([i, j, md])
    return maxmin, gen_opt

Listing A.3: Search for maximized minimum distance rank-1 lattice sequence based on an initial maximized minimum distance lattice.

def searchMMDR1LatticeSequenceApprox(b, m_max):
    n_max = b ** m_max; maxmin = 0
    gen_opt = []  # saves parameter g = (i, j)
    # build valid generator vectors
    gen_vec = []
    for m in range(1, m_max+1):
nTmp = (b ** m)
maxLength = math.sqrt(math.sqrt(4.0/3.0) * float(nTmp) * float(lx) * float(ly))
gen_vec += rasterizeCircle(maxLength)

for e in gen_vec:
i = e[0]; j = e[1]
if (computeGCD(i, j) != 1):
    continue

sumdist = 0; md = []
for m in range(1, m_max+1):
nTmp = b ** m
mindist = gaussianBasisReduction(nTmp, i, j)
sumdist += float(mindist)/nTmp
md.append(mindist)

if (sumdist == maxmin):
gen_opt.append([i, j, md])
else if (sumdist > maxmin):
    maxmin = sumdist
    gen_opt = []
    gen_opt.append([i, j, md])

return maxmin, gen_opt

Listing A.4: Approximate search for maximized minimum distance rank-1 lattice sequences with b and m_max as input parameters.

Search for Anisotropic Rank-1 Lattices

def searchAnisotropicLattice(dx, dy, width, n):
distD = float(dx*dx + dy*dy)
# set of scalars k, for which (k*g % n)
# generates identical lattices
phiVec = computePhiSet(n)
searchSpace = [0] * (n*n)
searchSpace[0] = 1
genvec = []

for gy in range(1, n):
    for gx in range(1, n):
        if (computeGCD(computeGCD(n, gx), gy) != 1):
            searchSpace[gx + n*gy] = 1
        if (searchSpace[gx + n*gy] == 0):
            # mark generator vector and gy which generate
            # identical lattices
            searchSpace[gx + n*gy] = 1
            for j in range(0, len(phiVec)):
                tx = (phiVec[j]*gx) % n
                ty = (phiVec[j]*gy) % n
                searchSpace[tx + n*ty] = 1

blx, bly, b2x, b2y = SearchBasisGR(n, gx, gy)
# compute reduced dual basis
Listing A.5: Searching for anisotropic rank-1 lattice. The main direction of the desired spectrum is given by \((dx, dy)\), and \(n\) denotes the number of lattice points.

### Searching for \((t, m, 2)\)-Nets from Shifted Rank-1 Lattices

```python
def search_tms_net_from_lattice(t, b, m):
    n = b * m
    generators = []

    # loop over all possible generators
    for a in range(1, (n+1)/2):
        b1, b2, b3, mindist = gaussianBasisReduction(n, 1, a)
        generators.append([mindist, [1, a], b1, b2, b3])

    # Sorting generator vectors by minimum distance
    generators.sort(lambda x, y: -cmp(x[0], y[0]))

    # Looping over sorted generator vectors and shifts
    for el in generators:
        g = el[1]
        bbmin, bbmax = computeBB([0, 0], el[2], el[3], el[4])

        # generate shifts and check tms property for each shift
        for sy in range(0, bbmax[1] - bbmin[1]):
            for sx in range(0, bbmax[0] - bbmin[0]):

```
if (isPointInLatticeCell(bbmin, sx, sy, n, el[2][0], el[2][1], el[3][0], el[3][1])):
    if (is_tm2_net(t, b, m, g, [sx, sy])):
        return [g, [sx, sy], el[0]]

return []

Listing A.6: Search a \((t, m, 2)\)-net from a shifted rank-1 lattice in basis \(b\). The source code for the auxiliary functions is given in the Listings A.10, A.11, A.12 and A.13.

A.2 Auxiliary Functions

Rasterization

```python
# rasterize circle of radius rad (in first quadrant)
# return list of points representing the rasterized circle of width 6
# use for search on unit square

def rasterizeCircle(rad):
    size = long(math.ceil(rad))
    cd = 1.25 - rad
cy = 0.0
    cx = rad
gx = long(math.floor(cx))
gy = long(math.floor(cy))
gen_vec = []
gen_vec.append([gx, gy])
gen_vec.append([gx+1, gy])
if ((gx-1) >= 0):
    gen_vec.append([gx-1, gy])
gen_vec.append([gx+2, gy])
if ((gx-2) >= 0):
    gen_vec.append([gx-2, gy])
gen_vec.append([gx+3, gy])
if ((gx-3) >= 0):
    gen_vec.append([gx-3, gy])

while (cx > cy):
    if (cd < 0.0):
        cd = cd + 2.0*cy + 3.0
    else:
        cd = cd + 2.0*cy - 2.0*cx + 5.0
    cx = cx - 1.0
cy = cy + 1.0
gx = long(math.floor(cx))
gy = long(math.floor(cy))
gen_vec.append([gx, gy])
gen_vec.append([gx+1, gy])
if ((gx-1) >= 0):
    gen_vec.append([gx-1, gy])
gen_vec.append([gx+2, gy])
if ((gx-2) >= 0):
    gen_vec.append([gx-2, gy])
```
gen_vec.append([gx+3, gy])
if ((gx-3) >= 0):
gen_vec.append([gx-3, gy])

return gen_vec

Listing A.7: Circle rasterization.

# rasterize ellipse with axis (a,0) and (0,b) (in first quadrant)
# return list of points representing the rasterized ellipse of width 6
# use for search on unit square
def rasterizeEllipse(a, b):
d1 = b*b - (a+a*b) + (0.25*a*a)
x = 0.0
y = b
gx = long(math.floor(x))
gy = long(math.floor(y))
ellipse = []
ellipse.append([gx, gy])

while (a*a*(y-0.5) > b*b*(x+1.0)):
    if (d1 < 0.0):  
d1 += b*b* (2.0*x + 3.0)
    else:  
d1 += b*b* (2.0*x + 3.0) + a*a * (-2.0*y + 2.0)
y -= 1.0
x += 1.0
gx = long(math.floor(x))
gy = long(math.floor(y))
ellipse.append([gx, gy])

d2 = b*b * (x + 0.5)*x + 0.5) + a*a * ((y - 1.0)*y - 1.0) - a*a*b*b

while (y > 0):
    if (d2 < 0.0):  
d2 += b*b* (2.0*x + 2.0) + a*a * (-2.0*y + 3.0)
x += 1.0
    else:  
d2 += a*a * (-2.0*y + 3.0)
y += 1.0
gx = long(math.floor(x))
gy = long(math.floor(y))
ellipse.append([gx, gy])

gen_vec = []
width = 3
xRes = long(math.ceil(a+width)) + 1
yRes = long(math.ceil(b+width)) + 1
boundingBox = []
boundingBox = [0] * (xRes*yRes)

for e in ellipse:
    tx = e[0]
ty = e[1]

    for i in range(-width, width+1):
        for j in range(-width, width+1):
            tx1 = tx+i
ty1 = ty+j
\( \text{if } ((tx1>=0) \text{ and } (tx1<xRes) \text{ and } (ty1>=0) \text{ and } (ty1<yRes)):\)
\( \text{if } (\text{boundingBox}[tx1 + ty1*xRes] == 0):\)
\( \text{boundingBox}[tx1 + ty1*xRes] = 1;\)
\( \text{gen_vec}.append([tx1, ty1])\)

**Listing A.8:** Ellipse rasterization.

### Computing the Set \( \{ \phi(n) \} \)

```python
def computePhiSet(n):
    phiVec = []
    for j in range(1, n):
        if (computeGCD(n, j) == 1):
            phiVec.append(j)
    return phiVec
```

**Listing A.9:** Computes the set of integers \( \{ \phi(n) \} = \{ x | 1 \geq x < n, \gcd(n, x) = 1 \} \).

### \((t, m, 2)\)-Nets from Shifted Rank-1 Lattices

```python
def is_tmn2_net(t, b, m, g, shift):
    n = b**m
    num = b**t
    numEllInts = b**(m-t) # number of el. intervals per kind k
    elInterval = []
    # init elementary intervals
    elInterval = [0] * (m-t+1) * (b**(m-t))
    # count lattice points per elementary interval
    for i in range(0, n):
        px = (i*g[0] + shift[0]) % n
        py = (i*g[1] + shift[1]) % n
    # loop over elementary intervals
    for k in range(0, m-t+1):
        dx = b**k
        dy = b**(m-t-k) # (b**(m-t)) / dx
        tx = b**(m-k) # n / dx
        ty = b**(t+k) # n / dy
        # coordinates of elementary interval of type k
        # in which lattice point lies
        elx = px//tx
        ely = py//ty
    # dx = number of el. intervals in x direction;
    # number of el. intervals per type k = b**(m-t)
    idx = elx + (dx*ely) + (k*numEllInts)
    ellInterval[idx] += 1
```

125
Listing A.10: Check if the lattice $L_{n,g}$ with $g = (1,a)$ is a $(t,m,2)$-net.

```python
def transformPointIntoLatticeBasis(px, py, n, b1x, b1y, b2x, b2y):
    tmpB1x = float(b1x) / float(n)
    tmpB1y = float(b1y) / float(n)
    tmpB2x = float(b2x) / float(n)
    tmpB2y = float(b2y) / float(n)
    tx = px * tmpB2y - py * tmpB2x
    ty = py * tmpB1x - px * tmpB1y
    denom = tmpB1x * tmpB2y - tmpB2x * tmpB1y
    tpx = tx / denom
    tpy = ty / denom
    return tpx, tpy
```

Listing A.11: Transform a point $(px, py) \in [0,1)^2$ into the lattice basis, given by $b1x, b1y, b2x, b2y$.

```python
def determineLatticeCell(px, py):
    tmpX = long(px)
    tmpY = long(py)
    if (tmpX <= px):
        cx = tmpX
    else:
        cx = tmpX - 1
    if (tmpY <= py):
        cy = tmpY
    else:
        cy = tmpY - 1
    return cx, cy
```

Listing A.12: Determine the lattice cell induced by the fundamental parallelepiped to which the point $(px, py)$ in lattice coordinates belongs. The lattice cell is described by its anchor point $(cx, cy)$.

```python
def isPointInLatticeCell(bxmin, px, py, n, b1x, b1y, b2x, b2y):
    qx = float(px + bbmin[0]) / float(n)
    qy = float(py + bbmin[1]) / float(n)
    tpx, tpy = transformPointIntoLatticeBasis(qx, qy, n, b1x, b1y, b2x, b2y)
    bx, by = determineLatticeCell(tpx, tpy)
```

Listing A.13: Check if a point $(px, py)$ in lattice coordinates is in the lattice cell $(cx, cy)$. If the transformed point $(tx, ty)$ is within the fundamental parallelepiped of the lattice cell, then the point $(px, py)$ belongs to the lattice cell.
Listing A.13: Check if the point $p = (px, py)$ in the bounding box $(bbmin, bbmax)$ lies within the inscribed lattice cell induced by the fundamental parallelepiped. $bbmin$ corresponds to the shift vector by which the bounding box is shifted such that the lower left corner of the bounding box coincides with the origin. The basis of the lattice with $n$ points is given by $b1x$, $b1y$, $b2x$, $b2y$. 

```python
if ((bx == 0) and (by == 0)):
    return True
else:
    return False
```
B.1 MMD Rank-1 Lattices not available in Korobov Form

The table lists the generator vectors $g = (g_1, g_2)$ for two-dimensional maximized minimum distance lattices which can not be represented in Korobov form (see Section 3.2.1). Note that the squared minimum distance $d_{\text{min}}(L_n, g)$ is scaled to $[0, n)^2$, such that it can be expressed in integers.

<table>
<thead>
<tr>
<th>$n$</th>
<th>$g_1$</th>
<th>$g_2$</th>
<th>$(d_{\text{min}}(L_n, g) \cdot n)^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>2</td>
<td>3</td>
<td>13</td>
</tr>
<tr>
<td>30</td>
<td>3</td>
<td>5</td>
<td>34</td>
</tr>
<tr>
<td>40</td>
<td>4</td>
<td>5</td>
<td>41</td>
</tr>
<tr>
<td>56</td>
<td>4</td>
<td>7</td>
<td>64</td>
</tr>
<tr>
<td>66</td>
<td>2</td>
<td>9</td>
<td>72</td>
</tr>
<tr>
<td>70</td>
<td>5</td>
<td>7</td>
<td>74</td>
</tr>
<tr>
<td>80</td>
<td>5</td>
<td>8</td>
<td>89</td>
</tr>
<tr>
<td>90</td>
<td>5</td>
<td>9</td>
<td>100</td>
</tr>
<tr>
<td>132</td>
<td>3</td>
<td>32</td>
<td>145</td>
</tr>
<tr>
<td>138</td>
<td>2</td>
<td>57</td>
<td>148</td>
</tr>
<tr>
<td>140</td>
<td>7</td>
<td>10</td>
<td>149</td>
</tr>
<tr>
<td>154</td>
<td>7</td>
<td>11</td>
<td>170</td>
</tr>
<tr>
<td>168</td>
<td>7</td>
<td>12</td>
<td>193</td>
</tr>
<tr>
<td>182</td>
<td>7</td>
<td>13</td>
<td>196</td>
</tr>
<tr>
<td>192</td>
<td>2</td>
<td>27</td>
<td>205</td>
</tr>
<tr>
<td>198</td>
<td>2</td>
<td>15</td>
<td>225</td>
</tr>
<tr>
<td>200</td>
<td>2</td>
<td>15</td>
<td>221</td>
</tr>
<tr>
<td>208</td>
<td>8</td>
<td>13</td>
<td>233</td>
</tr>
<tr>
<td>210</td>
<td>3</td>
<td>55</td>
<td>234</td>
</tr>
<tr>
<td>222</td>
<td>2</td>
<td>69</td>
<td>244</td>
</tr>
<tr>
<td>240</td>
<td>8</td>
<td>15</td>
<td>256</td>
</tr>
<tr>
<td>306</td>
<td>2</td>
<td>135</td>
<td>328</td>
</tr>
<tr>
<td>308</td>
<td>11</td>
<td>14</td>
<td>317</td>
</tr>
<tr>
<td>340</td>
<td>10</td>
<td>17</td>
<td>389</td>
</tr>
<tr>
<td>370</td>
<td>2</td>
<td>55</td>
<td>421</td>
</tr>
<tr>
<td>374</td>
<td>11</td>
<td>17</td>
<td>410</td>
</tr>
<tr>
<td>380</td>
<td>4</td>
<td>75</td>
<td>416</td>
</tr>
<tr>
<td>384</td>
<td>2</td>
<td>57</td>
<td>421</td>
</tr>
<tr>
<td>392</td>
<td>4</td>
<td>21</td>
<td>449</td>
</tr>
<tr>
<td>402</td>
<td>2</td>
<td>21</td>
<td>445</td>
</tr>
<tr>
<td>418</td>
<td>11</td>
<td>19</td>
<td>482</td>
</tr>
<tr>
<td>420</td>
<td>2</td>
<td>77</td>
<td>452</td>
</tr>
<tr>
<td>426</td>
<td>2</td>
<td>135</td>
<td>477</td>
</tr>
<tr>
<td>438</td>
<td>2</td>
<td>159</td>
<td>493</td>
</tr>
<tr>
<td>1008</td>
<td>3</td>
<td>92</td>
<td>1103</td>
</tr>
<tr>
<td>1020</td>
<td>17</td>
<td>30</td>
<td>1156</td>
</tr>
<tr>
<td>1050</td>
<td>2</td>
<td>93</td>
<td>1192</td>
</tr>
<tr>
<td>1054</td>
<td>2</td>
<td>493</td>
<td>1160</td>
</tr>
<tr>
<td>1060</td>
<td>4</td>
<td>35</td>
<td>1184</td>
</tr>
<tr>
<td>1062</td>
<td>2</td>
<td>219</td>
<td>1189</td>
</tr>
<tr>
<td>1064</td>
<td>4</td>
<td>35</td>
<td>1220</td>
</tr>
<tr>
<td>1065</td>
<td>3</td>
<td>130</td>
<td>1189</td>
</tr>
<tr>
<td>1092</td>
<td>3</td>
<td>152</td>
<td>1225</td>
</tr>
<tr>
<td>1116</td>
<td>18</td>
<td>31</td>
<td>1285</td>
</tr>
<tr>
<td>1122</td>
<td>2</td>
<td>525</td>
<td>1249</td>
</tr>
<tr>
<td>1130</td>
<td>2</td>
<td>365</td>
<td>1256</td>
</tr>
<tr>
<td>1135</td>
<td>5</td>
<td>98</td>
<td>1274</td>
</tr>
<tr>
<td>1160</td>
<td>2</td>
<td>225</td>
<td>1301</td>
</tr>
<tr>
<td>1164</td>
<td>2</td>
<td>273</td>
<td>1312</td>
</tr>
<tr>
<td>1176</td>
<td>7</td>
<td>36</td>
<td>1345</td>
</tr>
<tr>
<td>1178</td>
<td>19</td>
<td>31</td>
<td>1322</td>
</tr>
<tr>
<td>1206</td>
<td>2</td>
<td>369</td>
<td>1377</td>
</tr>
<tr>
<td>1216</td>
<td>19</td>
<td>32</td>
<td>1385</td>
</tr>
<tr>
<td>1245</td>
<td>3</td>
<td>95</td>
<td>1354</td>
</tr>
<tr>
<td>1250</td>
<td>2</td>
<td>395</td>
<td>1384</td>
</tr>
<tr>
<td>1254</td>
<td>19</td>
<td>33</td>
<td>1444</td>
</tr>
<tr>
<td>1260</td>
<td>2</td>
<td>265</td>
<td>1421</td>
</tr>
<tr>
<td>1344</td>
<td>3</td>
<td>172</td>
<td>1521</td>
</tr>
<tr>
<td>1350</td>
<td>5</td>
<td>102</td>
<td>1521</td>
</tr>
<tr>
<td>1378</td>
<td>2</td>
<td>507</td>
<td>1376</td>
</tr>
<tr>
<td>1386</td>
<td>2</td>
<td>183</td>
<td>1585</td>
</tr>
<tr>
<td>1392</td>
<td>2</td>
<td>219</td>
<td>1557</td>
</tr>
<tr>
<td>1404</td>
<td>3</td>
<td>143</td>
<td>1576</td>
</tr>
<tr>
<td>1428</td>
<td>2</td>
<td>34</td>
<td>1597</td>
</tr>
<tr>
<td>1430</td>
<td>2</td>
<td>225</td>
<td>1636</td>
</tr>
<tr>
<td>1452</td>
<td>3</td>
<td>148</td>
<td>1665</td>
</tr>
<tr>
<td>1470</td>
<td>3</td>
<td>265</td>
<td>1636</td>
</tr>
<tr>
<td>1480</td>
<td>5</td>
<td>112</td>
<td>1649</td>
</tr>
</tbody>
</table>
The table provides a list of the parameters, i.e. the generator vector \( \mathbf{g} \) and a lattice basis \( \mathbf{B} \), of all maximized minimum distance lattices in two dimensions with \( n = 2^i \) points for \( i = 2, \ldots, 31 \) in order to ease the implementation of the spectral synthesis of ocean waves in Section 4.2. Note that the basis vectors are given in integer precision and have to be divided by the number of lattice points.
<table>
<thead>
<tr>
<th>$i$</th>
<th>$n = 2^i$</th>
<th>generator $\mathbf{g} = (g_1, g_2)$</th>
<th>basis vectors $\mathbf{B} = (b_1 b_2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>64</td>
<td>(1, 28) (1, 36)</td>
<td>(7, 4), (2, -8) (7, -4), (2, 8)</td>
</tr>
<tr>
<td>7</td>
<td>128</td>
<td>(1, 12) (1, 116)</td>
<td>(11, 4), (1, 12) (11, -4), (1, -12)</td>
</tr>
<tr>
<td>8</td>
<td>256</td>
<td>(1, 30) (1, 226)</td>
<td>(9, 14), (17, -2) (9, -14), (17, 2)</td>
</tr>
<tr>
<td>9</td>
<td>512</td>
<td>(1, 200) (1, 312)</td>
<td>(18, 10), (23, -8) (18, -16), (23, 8)</td>
</tr>
<tr>
<td>10</td>
<td>1024</td>
<td>(1, 271) (1, 495) (1, 529) (1, 753)</td>
<td>(34, -21), (15, -31) (2, -34), (31, -15) (2, 34), (31, 15) (34, 21), (15, 31)</td>
</tr>
<tr>
<td>11</td>
<td>2048</td>
<td>(1, 592) (1, 1456)</td>
<td>(45, 16), (7, 48) (45, -16), (7, -48)</td>
</tr>
<tr>
<td>12</td>
<td>4096</td>
<td>(1, 70) (1, 4026)</td>
<td>(59, 34), (58, -36) (59, -34), (58, 36)</td>
</tr>
<tr>
<td>13</td>
<td>8192</td>
<td>(1, 1530) (1, 6662)</td>
<td>(91, -34), (75, 62) (91, 34), (75, -62)</td>
</tr>
<tr>
<td>14</td>
<td>16384</td>
<td>(1, 1435) (1, 6291) (1, 10093) (1, 14949)</td>
<td>(57, -125), (137, -13) (125, -57), (13, -137) (125, 57), (13, 137) (57, 125), (137, 13)</td>
</tr>
<tr>
<td>15</td>
<td>32768</td>
<td>(1, 15936) (1, 16832)</td>
<td>(183, -64), (146, 128) (183, 64), (146, -128)</td>
</tr>
<tr>
<td>16</td>
<td>65536</td>
<td>(1, 25962) (1, 39574)</td>
<td>(260, -88), (53, -270) (260, 88), (53, 270)</td>
</tr>
<tr>
<td>17</td>
<td>131072</td>
<td>(1, 49331) (1, 62899) (1, 68173) (1, 81541)</td>
<td>(172, -348), (217, 323) (348, -172), (323, 217) (348, 172), (323, -217) (172, 348), (217, -323)</td>
</tr>
<tr>
<td>18</td>
<td>262144</td>
<td>(1, 1990) (1, 260154)</td>
<td>(527, 154), (395, -382) (527, -154), (395, 382)</td>
</tr>
<tr>
<td>19</td>
<td>524288</td>
<td>(1, 86592) (1, 437696)</td>
<td>(775, -64), (442, 640) (775, 64), (442, -640)</td>
</tr>
<tr>
<td>20</td>
<td>1048576</td>
<td>(1, 195638) (1, 852938)</td>
<td>(134, 1092), (879, -662) (134, -1092), (879, 662)</td>
</tr>
<tr>
<td>21</td>
<td>2097152</td>
<td>(1, 193293) (1, 715835) (1, 1381317) (1, 1903859)</td>
<td>(1226, -958), (217, 1541) (958, 1226), (1541, -217) (958, -1226), (1541, 217) (1226, 958), (217, -1541)</td>
</tr>
<tr>
<td>22</td>
<td>4194304</td>
<td>(1, 1120786) (1, 3073518)</td>
<td>(363, -2170), (1699, 1398) (363, 2170), (1699, -1398)</td>
</tr>
<tr>
<td>23</td>
<td>8388608</td>
<td>(1, 1671221) (1, 3288547) (1, 5100061) (1, 6717387)</td>
<td>(1807, -2533), (3097, 301) (2533, 1807), (301, -3097) (2533, -1807), (301, 3097) (1807, 2533), (3097, -301)</td>
</tr>
<tr>
<td>24</td>
<td>16777216</td>
<td>(1, 7605516) (1, 9171700)</td>
<td>(2903, -3308), (1414, 4168) (2903, 3308), (1414, -4168)</td>
</tr>
<tr>
<td>25</td>
<td>33554432</td>
<td>(1, 1905545) (1, 14462279) (1, 19092153) (1, 31648887)</td>
<td>(405, -62111), (5582, -2754) (6211, 405), (2754, 5582) (6211, -405), (2754, -5582) (405, 6211), (5582, 2754)</td>
</tr>
</tbody>
</table>
### B.3 \((t, m, 2)\)-Nets from Shifted Rank-1 Lattices in Base \(b\)

The table summarizes the search experiments of Section 3.5 for shifted rank-1 lattices in Korobov form \(L^\Delta_{\omega, (1,a)}\) that are \((t, m, 2)\)-nets in base \(b\) by listing the generator vector \(g\), the shift vector \(\Delta\) and both the minimum distance \(d_{\min}(L_{n,g})\) and the maximal possible minimum distance MMD for rank-1 lattices with \(n = b^m\) lattice points. The rational shift \(\Delta\) is scaled by \(n\) for integer precision. The shift domain corresponds to the bounding box of the fundamental parallelepiped shifted to the origin (i.e. lower left corner is the origin). For this table no exclusion test for shifts outside the fundamental parallelepiped has been performed in order to be able to always capture the shift \(\mathbf{0}\) if feasible. As the search algorithm is stopped after the first valid \((t, m, 2)\)-lattice is found, only one possible shift is displayed in the table.

<table>
<thead>
<tr>
<th>(i)</th>
<th>(n = 2^i)</th>
<th>(\text{generator } g = (g_1, g_2))</th>
<th>(\text{basis vectors } B = (b_1b_2))</th>
</tr>
</thead>
<tbody>
<tr>
<td>26</td>
<td>67108864</td>
<td>((1, 22282116))</td>
<td>((6391, -6052), (8436, 2512))</td>
</tr>
<tr>
<td></td>
<td></td>
<td>((1, 44826748))</td>
<td>((6391, 6052), (8436, -2512))</td>
</tr>
<tr>
<td>27</td>
<td>134217728</td>
<td>((1, 58928436))</td>
<td>((9147, 8444), (2740, -12144))</td>
</tr>
<tr>
<td></td>
<td></td>
<td>((1, 75289292))</td>
<td>((9147, -8444), (2740, 12144))</td>
</tr>
<tr>
<td>28</td>
<td>268435456</td>
<td>((1, 86198508))</td>
<td>((682, 17592), (15577, 8204))</td>
</tr>
<tr>
<td></td>
<td></td>
<td>((1, 182236948))</td>
<td>((682, -17592), (15577, -8204))</td>
</tr>
<tr>
<td>29</td>
<td>536870912</td>
<td>((1, 8370742))</td>
<td>((11737, 21958), (24885, 814))</td>
</tr>
<tr>
<td></td>
<td></td>
<td>((1, 528500170))</td>
<td>((11737, -21958), (24885, -814))</td>
</tr>
<tr>
<td>30</td>
<td>1073741824</td>
<td>((1, 78999493))</td>
<td>((10221, -33695), (24071, 25699))</td>
</tr>
<tr>
<td></td>
<td></td>
<td>((1, 284281075))</td>
<td>((10221, 33695), (24071, -25699))</td>
</tr>
<tr>
<td></td>
<td></td>
<td>((1, 789460749))</td>
<td>((33695, 10221), (25699, -24071))</td>
</tr>
<tr>
<td></td>
<td></td>
<td>((1, 994742331))</td>
<td>((33695, -10221), (25699, 24071))</td>
</tr>
<tr>
<td>31</td>
<td>2147483648</td>
<td>((1, 940574718))</td>
<td>((38453, 31638), (8176, -49120))</td>
</tr>
<tr>
<td></td>
<td></td>
<td>((1, 1206908930))</td>
<td>((38453, -31638), (8176, 49120))</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(b)</th>
<th>(t)</th>
<th>(m)</th>
<th>(g = (g_1, g_2))</th>
<th>(\Delta)</th>
<th>(d_{\min}(L_{n,g}))</th>
<th>(\text{MMD})</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>((1, 1))</td>
<td>((0, 0))</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>2</td>
<td>((1, 3))</td>
<td>((0, 0))</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>3</td>
<td>((1, 5))</td>
<td>((0, 0))</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>4</td>
<td>((1, 7))</td>
<td>((1, 0))</td>
<td>8</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>5</td>
<td>((1, 9))</td>
<td>((0, 0))</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>6</td>
<td>((1, 50))</td>
<td>((0, 0))</td>
<td>61</td>
<td>65</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>7</td>
<td>((1, 34))</td>
<td>((0, 0))</td>
<td>80</td>
<td>137</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>8</td>
<td>((1, 66))</td>
<td>((8, 0))</td>
<td>80</td>
<td>277</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>((1, 1))</td>
<td>((0, 0))</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>2</td>
<td>((1, 4))</td>
<td>((0, 0))</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>3</td>
<td>((1, 8))</td>
<td>((2, 0))</td>
<td>18</td>
<td>26</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>4</td>
<td>((1, 24))</td>
<td>((2, 0))</td>
<td>85</td>
<td>85</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>5</td>
<td>((1, 57))</td>
<td>((0, 0))</td>
<td>241</td>
<td>261</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>6</td>
<td>((1, 327))</td>
<td>((0, 0))</td>
<td>810</td>
<td>810</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>((1, 1))</td>
<td>((0, 0))</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>b</td>
<td>t</td>
<td>m</td>
<td>(g = (g_1, g_2))</td>
<td>(\Delta)</td>
<td>(d_{\text{min}}(f_{\mathcal{g}}, g))</td>
<td>MMD</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>2</td>
<td>(1.5)</td>
<td>(0.0)</td>
<td>10</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>3</td>
<td>(1.17)</td>
<td>(0.0)</td>
<td>32</td>
<td>65</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>4</td>
<td>(1.220)</td>
<td>(0.0)</td>
<td>277</td>
<td>277</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>5</td>
<td>(1.380)</td>
<td>(7.0)</td>
<td>1088</td>
<td>1160</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>(1.2)</td>
<td>(0.0)</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>2</td>
<td>(1.7)</td>
<td>(1.0)</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>3</td>
<td>(1.26)</td>
<td>(0.0)</td>
<td>50</td>
<td>125</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>4</td>
<td>(1.185)</td>
<td>(1.0)</td>
<td>689</td>
<td>689</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
<td>(1.1)</td>
<td>(0.0)</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>2</td>
<td>(1.31)</td>
<td>(0.0)</td>
<td>26</td>
<td>36</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>3</td>
<td>(1.35)</td>
<td>(5.0)</td>
<td>72</td>
<td>233</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>4</td>
<td>(1.103)</td>
<td>(11.0)</td>
<td>1378</td>
<td>1460</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>1</td>
<td>(1.2)</td>
<td>(0.0)</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>2</td>
<td>(1.8)</td>
<td>(0.0)</td>
<td>37</td>
<td>49</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>3</td>
<td>(1.146)</td>
<td>(6.0)</td>
<td>98</td>
<td>370</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>4</td>
<td>(1.861)</td>
<td>(4.0)</td>
<td>2597</td>
<td>2725</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>1</td>
<td>(1.3)</td>
<td>(0.0)</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>2</td>
<td>(1.19)</td>
<td>(1.0)</td>
<td>58</td>
<td>65</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>3</td>
<td>(1.65)</td>
<td>(0.0)</td>
<td>128</td>
<td>580</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>1</td>
<td>(1.4)</td>
<td>(0.0)</td>
<td>5</td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>2</td>
<td>(1.34)</td>
<td>(6.0)</td>
<td>74</td>
<td>85</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>3</td>
<td>(1.80)</td>
<td>(8.0)</td>
<td>162</td>
<td>810</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>1</td>
<td>(1.3)</td>
<td>(0.0)</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>2</td>
<td>(1.9)</td>
<td>(9.0)</td>
<td>82</td>
<td>104</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>3</td>
<td>(1.99)</td>
<td>(9.0)</td>
<td>200</td>
<td>1105</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>1</td>
<td>(1.4)</td>
<td>(0.0)</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>2</td>
<td>(1.36)</td>
<td>(7.0)</td>
<td>109</td>
<td>121</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>3</td>
<td>(1.120)</td>
<td>(10.0)</td>
<td>242</td>
<td>1514</td>
</tr>
</tbody>
</table>
Bibliography


