TOPOLOGICAL PROPERTIES OF TILES AND DIGIT SETS

by

AVRA SOPHIA SASKIA LAARAKKER

Thesis
submitted in partial fulfillment of the requirements for
the Degree of Master of Science (Mathematics)

Acadia University
Fall Convocation, 2009

This thesis by AVRA SOPHIA SASKIA LAARAKKER was defended successfully in an oral examination on June 30, 2009.

The examining committee for the thesis was:

Dr. Elhadi Shakshuki, Chair

Dr. Tara Taylor, External Reader

Dr. Jeff Hooper, Internal Reader

Dr. Franklin Mendivil, Co-Supervisor

Dr. Eva Curry, Supervisor

Dr. Paul Stephenson, Head, Department of Mathematics and Statistics

This thesis is accepted in its present form by the Division of Research and Graduate Studies as satisfying the thesis requirements for the degree Master of Science (Mathematics).

. . . . . .

This thesis by AVRA SOPHIA SASKIA LAARAKKER was defended successfully in an oral examination on June 30, 2009.

The examining committee for the thesis was:

Dr. Elhadi Shakshuki, Chair

Dr. Tara Taylor, External Reader

Dr. Jeff Hooper, Internal Reader

Dr. Franklin Mendivil, Co-Supervisor

Dr. Eva Curry, Supervisor

Dr. Paul Stephenson, Head, Department of Mathematics and Statistics

This thesis is accepted in its present form by the Division of Research and Graduate Studies as satisfying the thesis requirements for the degree Master of Science (Mathematics).

_____
Author

_____
Supervisor

_____
Date

Table of Contents

List of Tables

List of Figures

Abstract

This thesis investigates topological properties of tiles, $T(A, D)$, arising from iterated function systems generated by dilation matrices, A, and associated digit sets, D. We develop a new method for investigating connectedness of $T(A, D)$ via level sets of the digit set and matrix similarity. We conjecture that there exists digit set D_A for any dilation matrix A, for which $T(A, D_A)$ is connected. We prove this conjecture for several important cases. While it has previously been shown in the two dimensional case that digit sets exist for which $T(A, D)$ is connected [5], we provide a new, constructive method that we expect will easily generalize to higher dimensions, where no results of this form are currently known.

We also develop new computational tools for investigating tiles $T(A, D)$, including an implementation of the neighbour-finding algorithm of Scheicher and Thuswaldner [20].

List of Symbols

Acknowledgements

Thank you to Dr. Eva Curry for her patience, guidance and support during this long process, and for giving me a chance. I would also like to thank Acadia University and especially the Mathematics & Statistics Department for giving me the opportunity to complete this thesis. Thank you to Duane Currie for all of the technical assistance, technical support, and everlasting patience with my never-ending technical problems. Thank you also to the Sage Development Team for ongoing support. A special thanks to NSERC for funding a portion of this research project.

Dedication

This thesis is for my family and all the regulars at 33 Cherry Lane.

1 Introduction

In this thesis, we examine the topological properties of connectedness and disk-likeness of tiles $T(A, D)$ arising from dilation matrices A and associated digit sets D in $\mathbb{R}^n$. In this chapter, we describe the mathematical setting we are working in, and provide a motivation as to why we are looking at this problem.

A *lattice* Γ is a discrete subgroup of $\mathbb{R}^n$ which spans the real vector space $\mathbb{R}^n$. Every lattice in $\mathbb{R}^n$ can be generated from an appropriate basis for the vector space by forming all linear combinations with integral coefficients. An *integer lattice* Γ is a discrete subgroup of $\mathbb{Z}^n$ given by integral linear combinations of some basis $\{v_1, v_2, ..., v_n\}$ where $v_i \in \mathbb{Z}^n$. $M_n(\mathbb{Z})$ is the set of $n \times n$ matrices with entries in $\mathbb{Z}$. For a lattice Γ and an $n \times n$ matrix A, we say that A is a *dilation matrix* if $A(\Gamma) \subset \Gamma$ and if all eigenvalues λ of A satisfy $|\lambda| > 1$, in other words, if A is expanding. Let A be a dilation matrix in $M_n(\mathbb{Z})$, set $q = |\det A|$, and let $D = \{d_1, ..., d_q\} \in \mathbb{Z}^n$ be a set of q distinct vectors. If D is a complete set of coset representatives of $\mathbb{Z}^n / A(\mathbb{Z}^n)$, then we call D a *q-digit set*, or simply a *digit set*.

Definition 1.1. *[7] Let A be a dilation matrix. We say that the matrix A* yields a radix representation with digit set D *if for every $x \in \mathbb{Z}^n$ there exists a nonnegative integer $N = N(x)$ and a sequence of digits $d_0, d_1, ..., d_N$ in D such that*

$$x = \sum_{j=0}^{N(x)} A^j d_j.$$

That is, a dilation matrix A yields a radix representation with digit set D if every $x \in \mathbb{Z}^n$ has a radix representation with radix A and digit set D.

For example, in one dimension, the radix $A = 3$ yields a radix representation with digit set $D_1 = \{-1, 0, 1\}$, but does not yield a radix representation with digit set $D_2 = \{0, 1, 2\}$ as no negative numbers can be represented with D_2.

Formally, an *iterated function system (IFS)* is a finite set of contraction mappings on a complete metric space. Symbolically $\{f_i : X \to X | i = 1, 2, ..., N\},\ N \in \mathbb{N}$ is an iterated function system if each f_i is a contraction on the complete metric space X. We are considering the metric space of closed, compact sets in $\mathbb{R}^n$, with the Hausdorff metric, $X = \mathcal{H}(\mathbb{R}^n)$. If we set $f_d(x) := A^{-1}(x + d)$ for each $d \in D$, then these are contractive maps under the Hausdorff metric [6].

Definition 1.2. *The* Hausdorff distance $h(A, B)$ *between two sets A and B in $\mathbb{R}^n$ is defined to be*

$$h(A, B) = \max\{d(A, B), d(B, A)\},$$

where the distance between two sets A and B is the non-symmetric function

$$d(A, B) = \max_{a \in A} \min_{b \in B} d(a, b).$$

It is well known that for a family of contractions $f_1, ..., f_q$ on $\mathcal{H}(\mathbb{R}^n)$, there is a unique non-empty compact set $T \subset \mathbb{R}^n$ with $T = \bigcup_i f_i(T)$ [6]. The Contraction Mapping Theorem says that every contractive mapping has a fixed point; in particular, every IFS has a fixed point, which is some compact, closed set in $\mathbb{R}^n$, called the *attractor* of the IFS [6]. In our case, the attractor T of the IFS is given

by

$$T = T(A, D) := \{\sum_{j=1}^{\infty} A^{-j} d_{-j} : d_{-j} \in D\}.$$

Note that T is a compact, closed set in $\mathbb{R}^n$ [11].

We would like to be able to think of the elements of T as the fractional parts of vectors in $\mathbb{R}^n$ in the same way that the fractional parts of real numbers lie in $[0, 1]$. This is an accurate interpretation if T is congruent modulo $\mathbb{Z}^n$ to $\mathbb{R}^n/\mathbb{Z}^n$. Consider the well known base 10 system. Here $A = 10$ and $D = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. We can build up all positive integers by considering polynomials of the form:

$$k = 10^n d_n + 10^{n-1} d_{n-1} + ... + 10 d_1 + d_0, \quad \text{with } d_i \in D.$$

By taking the polynomials with negative powers of 10, we have our usual decimal representation. We are able to represent every real number in the interval $[0, 1]$, so integer translations of that interval will give all of $\mathbb{R}$. So, thought of another way, we would like T to tile $\mathbb{R}^n$ under translation by $\mathbb{Z}^n$.

Definition 1.3. *A measurable set $T \subset \mathbb{R}^n$ gives a* self-affine tiling *of $\mathbb{R}^n$ under translation by $\mathbb{Z}^n$ if:*

1. *(**Tiling**) $\bigcup_{k \in \mathbb{Z}^n} (T + k) = \mathbb{R}^n$, and the intersection $(T + k_1) \cap (T + k_2)$ has measure zero for any two distinct $k_1, k_2 \in \mathbb{Z}^n$, and*
2. *(**Self-Affine**) there is a collection of $q = |\det A|$ vectors $d_1, ..., d_q \in \mathbb{Z}^n$ that are distinct coset representatives of $\mathbb{Z}^n/A(\mathbb{Z}^n)$ such that*

$$A(T) \simeq \cup_{i=1}^{q} (T + d_i).$$

When $T = T(A, D) := \{\sum_{j=1}^{\infty} A^{-j} d_{-j} : d_{-j} \in D\}$ is our attractor, we can write $T = \bigcup_{d \in D} f_d(T)$, since

$$\begin{aligned} f_d(T) &= \left\{ A^{-1} \left(\sum_{j=1}^{\infty} A^{-j} d_{-j} \right) + A^{-1} d : d_{-j} \in D \right\} \\ &= \left\{ A^{-1} d + \left(\sum_{j=2}^{\infty} A^{-j} d_{-j} \right) : d_{-j} \in D \right\}. \end{aligned}$$

It is clear from this that T is self-affine,

$$T = \bigcup_{d \in D} A^{-1}(T + d).$$

Also, in many cases, for example when A yields a radix representation for every $k \in \mathbb{Z}^n$ with digit set D, T tiles $\mathbb{R}^n$ under translation by $\mathbb{Z}^n$ [7]. That is, $\bigcup_{k \in \mathbb{Z}^n} T + k = \mathbb{R}^n$, and $m(T \cap (T + k)) = 0$ for all $k \in \mathbb{Z}^n$, $k \neq 0$.

At this point a connection between these radix representation and tiling properties is becoming evident. First, we can ask the following questions:

- What dilation matrices A yield radix representations for every vector $k \in \mathbb{Z}^n$?
- When do we have a radix representation for every vector $x \in \mathbb{R}^n$?

The answers to both of these questions depend strongly on the choice of our digit set D. A digit set is any set of coset representatives, so many potential digit sets are possible. In general, if one takes any fundamental domain F for $\mathbb{Z}^n$ (that is, any set $F \subset \mathbb{R}^n$ congruent to $\mathbb{R}^n/\mathbb{Z}^n$), then $A(F) \cap \mathbb{Z}^n$ will be a set of coset representatives of $\mathbb{Z}^n/A(\mathbb{Z}^n)$, and thus a candidate for a digit set.

In the base 10 example we do not have a true radix representation since we cannot express any negative integers. Also, we run into problems since the origin $\mathbf{0}$ is on the 'edge' of the digit set in some sense. In general, choosing the smallest possible coset representatives as digits, and choosing digits that can be generated by the intersection of a convex set with $\mathbb{Z}^n$, seems to produce digit sets that are easier to work with and more likely to yield a radix representation.

Dr. Curry [7] has shown that good qualities in choices of digit sets arise when we are centered about the origin. Thus in this thesis we will consider

$$D = A\left[-\frac{1}{2},\frac{1}{2}\right)^n \cap \mathbb{Z}^n \text{ or } D = A\left(-\frac{1}{2},\frac{1}{2}\right]^n \cap \mathbb{Z}^n,$$

where $F = \left[-\frac{1}{2},\frac{1}{2}\right)^n$ or $\left(-\frac{1}{2},\frac{1}{2}\right]^n$ is a fundamental domain for the lattice $\mathbb{Z}^n$.

We can also now look at the 'level sets' of the digit set D. The level sets D_n are defined as follows:

$$D_n := \{k \in \Gamma \big| k = \sum_{j=0}^{n-1} A^j d_j,\ d_j \in D\}.$$

With this in mind we can consider both the skewness of the dilation matrix A or the skewness of the level sets of the digits. For skewness of the dilation matrix A, we mean the skew of the matrix A. For the skewness of the level sets of the digit sets we mean the horizontal and/or vertical spread of the digits on the integer lattice. In this research project we considered the skewness of the level sets. It will be explained in later sections why this approach is useful.

In the one-dimensional setting, a digit set that would yield a radix representation is also known as a basic digit set. Matula [18] has shown that D is a basic digit

set for A if and only if:

1. D is a residue digit set for A (i.e., a complete residue system modulo $|A|$ with $0 \in D$), and
2. D_n, the n^{th} level set of the digit set D, contains no nonzero multiple of $A^n - 1$ for any $n \geq 1$.

When $n > 1$, a sufficient requirement for a radix representation is as follows:

Theorem 1.4. *[7] Let A be a dilation matrix with digit set*

$$D = A\left[-\frac{1}{2}, \frac{1}{2}\right)^n \cap \mathbb{Z}^n.$$

If all singular values σ of A satisfy $\sigma > 2$ then A yields a radix representation for all vectors $k \in \mathbb{Z}^n$.

Suppose we have a dilation matrix A, a digit set D, and the IFS defined by A and D. Let Γ denote a discrete subset of $\mathbb{R}^n$ and T be a compact set in $\mathbb{R}^n$ which coincides with the closure of its interior. If translates by Γ of T cover $\mathbb{R}^n$ and the interiors of two distinct translates have no intersection, then we say that T satisfies the *tiling condition*. In short, we say that T is a tile.

From the previous examples we see that radix representations are closely related to self-affine tilings of $\mathbb{R}^n$.

Theorem 1.5. *[7] Let A be a dilation matrix with digit set $D = A\left[-\frac{1}{2}, \frac{1}{2}\right)^n \cap \mathbb{Z}^n$.*

If A yields a radix representation for all of $\mathbb{Z}^n$, then a tile

$$T = T(A, D) := \{\sum_{j=1}^{\infty} A^{-j} d_{-j} : \ d_{-j} \in D\}$$

is congruent to $[0, 1]^n$, and thus every vector $x \in \mathbb{R}^n$ has a radix representation

$$x = \sum_{-\infty}^{N(x)} A^j d_j$$

with $d_j \in D$.

Noticing that A and D define an iterated function system helps us study both the number theory of radix representations as well as the topology of the set $T(A, D)$. In our case, the set mapping for $T \subset \mathbb{R}^n$ defined by this particular IFS is:

$$T \to \bigcup_{d \in D} A^{-1}(T + d).$$

We are particularly interested in the topology of these tiles.

Some important questions about the set $T(A, D)$ include:

1. Is $T(A, D)$ connected?
2. What does it mean for the digit set D to be connected, and does this give us additional conditions for $T(A, D)$ to be connected?
3. Is $T(A, D)$ homeomorphic to the unit disk in $\mathbb{R}^n$ ('disklike')?

Answers to the first and third questions depend on the choice of dilation matrix A and digit set D. When the digit set has 'gaps', or is too skew (or spread) the

answer to both questions is likely to be "no." Testing examples indicates that if D is the 'nice' digit set $D = A\left[-\frac{1}{2}, \frac{1}{2}\right)^n \cap \mathbb{Z}^n$, A yields a radix representation with digit set D, and A is not too skew, then the answer to both questions seems more likely to be "yes". An answer to the second question will come from examining the lattice connectedness of the level sets, D_n, as it appears to be that when the sets D_n is not lattice connected, the tile T is no longer connected.

To actually answer these questions, people have taken geometric, topological, graph theoretic, analytical, wavelet theoretic, and number theoretic approaches. Our approach is more topological, with the goal of establishing conditions such that connectedness and/or disklikeness is guaranteed; but, as we will describe in the next few chapters, influences from all of these fields have been important in investigating this problem.

In Chapter 2, we survey results from a variety of mathematical perspectives that provide insight into the types of problems one can look at when studying tiles $T(A, D)$.

In Chapter 3, we present results from the literature that were specifically used in this project in more detail as well as in consistent notation. The notation introduced in this chapter will be adopted in the following chapters.

In Chapter 4, we establish new results on the connectedness of T and the lattice connectedness of D_n. As well, we provide a new method for investigating connectedness and proving some previously established results in two dimensions that we expect will more readily generalize to three and higher dimensions.

We focussed our investigation on matrices of the form

$$A = \begin{bmatrix} \lambda & k \\ 0 & \lambda \end{bmatrix}.$$

We noted that rotation and dilation transformations alone do not affect lattice connectivity of the level sets D_n, nor the overall connectivity of the tile $T(A, D)$. It is the skewness of the matrix, and thus the digit set and the level sets D_n, that will determine connectivity of $T(A, D)$. Thus we considered matrices that give a purely skew transformation. We follow Kirat and Lau in this approach [11].

In order to minimize problems resulting from skew, we show that it is more useful to consider less skew matrices that are similar to A. Specifically, we looked at the Jordan Normal form J of A, and the canonical digit set D_J for J. We prove the following results.

Lemma. *The tile $T(A, D_A)$ is connected if and only if $T(J, D_J)$ is connected.*

Theorem. *For matrices of the form $J = \begin{bmatrix} \lambda & 1 \\ 0 & \lambda \end{bmatrix}$, such that $\lambda \geq 3$ an integer, we have that $T(J, D_J)$ is connected.*

Thus we prove the following conjecture in all but one sub-case.

Conjecture. *There exists a digit set D_A for which $T(A, D_A)$ is connected.*

In addition to laying out a program for investigating topological properties of tiles $T(A, D)$ via matrix similarity, we extend previous results that use the IFS structure of the radix representation to prove results on connectedness of $T(A, D)$, and

to show that connectedness of $T(A, D)$ is closely related to lattice connectedness of the level sets D_n.

We begin by giving a new proof of the following lemma.

Lemma. *Suppose that T_n is a sequence of compact, connected subsets of $\mathbb{R}^m$, and that in the Hausdorff metric $T = \lim_{n\to\infty} T_n$. Then T is connected.*

We also prove the following new results.

Proposition. *Let S be the set of neighbours of $T = T(A, D)$, and B a basis of $\mathbb{Z}^m$ such that $B \subset S$. If D is B-connected, then T is connected.*

The set S of neighbours of a tile T is defined in Chapter 2.

Lemma. *The set T_n is connected if and only if the level set D_n is lattice connected.*

In Chapter 5, we describe the implementation of an important algorithm from the literature that had not previously been implemented. We illustrate several examples which also demonstrate our results and the Maple package and Sage procedures written throughout the course of this project. This project made extensive use of the experimental mathematics approach of testing carefully chosen examples to help formulate conjectures.

In Chapter 6, we outline several interesting experimental observations that have led us to formulate several unresolved conjectures, including exploring experimentally shown bounds on λ and k determining connectedness of T, and indicate future research stemming from this project.

2 Literature Review

Other authors have approached the problem of connectedness and disklikeness of tiles in a variety of ways from many different areas of mathematics as indicated in the Introduction. In this chapter, we survey results from several areas of mathematics to give an idea of the types of results currently known about connectedness and disklikeness of a tile $T(A, D)$. In studying connectedness, authors such as Akiyama, Gjini, Kirat, Lau, Leung, Ngai, Rao, and Tang have considered polynomials associated with the dilation matrix A, mainly the characteristic polynomial. Authors including Bandt, Gelbrich, Gröchenig, Haas, Ngai, Tang, and Wang have taken a more topological approach looking at neighbours of a tile. Several authors made use of a graph theoretic approach including Kirat, Lau, Scheicher, and Thuswaldner. We introduce this approach briefly in this chapter, but develop it in greater detail in Chapter 3. In studying disklikeness, authors have tended to take a primarily topological approach. We survey the work of Akiyama, Lagarias, Luo, Ngai, Rao, Tan, Tang, Thuswaldner, and Wang.

Recall that we are looking at a tile $T = T(A, D)$, determined by a dilation matrix A and a digit set D, where T is the attractor of the IFS $\{f_i\}_{i=1}^{q}$ and is given by

$$T(A, D) := \{\sum_{j=1}^{\infty} A^{-j} d_{-j} : d_{-j} \in D\}.$$

There are two conditions, the Open Set Condition [6] and the Tiling Condition [13], that are both very useful in discussing our IFS and are mentioned and used by all authors when looking at these tiles.

Definition 2.1. The Open Set Condition *If there exists a non-empty open set $V \subset \mathbb{R}^n$ such that $f_i(V) \cap f_j(V) = \emptyset$ for $i \neq j$ and $\bigcup_i f_i(V) \subset V$, then we say that $f_1, ..., f_q$ (or T) satisfy the open set condition.*

Definition 2.2. The Tiling Condition *Let K denote a discrete subset of $\mathbb{R}^n$, and T be a compact set in $\mathbb{R}^n$ which coincides with the closure of its interior. If translates of T by K cover $\mathbb{R}^n$ and the interiors of two distinct translates have no intersection, then we say that T satisfies the tiling condition. In short, we say that T is a tile.*

Kirat and Lau [11] looked at particular types of digit sets, characteristic polynomials, and determinant values for a dilation matrix $A \in M_n(\mathbb{Z})$. Specifically, they looked at *consecutive collinear* digit sets of the form $\{0, v, 2v, ..., (|q| - 1)v\} \in \mathbb{Z}^n$. These collinear digit sets are one straightforward way to generalize one-dimensional digit sets (for example, when $A = 10$, $D = \{0, 1, ..., 9\}$). As noted in the Introduction, such digit sets are in general not suitable when we want A to yield a radix representation for every integer vector (recall the requirement that a digit set be basic [18]). However, they provide the following criterion for the attractor T to be a tile by the following theorem:

Theorem 2.3. *Suppose $A \in M_n(\mathbb{Z})$ is an expanding matrix with $|\det A| = q$, $q \geq 2$ is a prime. Let $D = \{d_1 v, ..., d_q v\}$ with $v \in \mathbb{R}^n / \{0\}$, $d_i \in \mathbb{Z}$. Then T is a self-affine tile if and only if $\{v, Av, ...A^{(n-1)}v\}$ is a linearly independent set and $\{d_1, ..., d_q\} = q^l \{d'_1, ..., d'_q\}$ where $\{d'_1, ..., d'_q\}$ is a complete set of coset representatives of $\mathbb{Z}_q$.*

In particular, if $v \in \mathbb{Z}/\{0\}$, then $\{v, Av, ...A^{(n-1)}v\}$ is automatically a linearly independent set. Hence T is a self-affine tile if and only if the above $\{d'_1, ..., d'_q\}$ is a complete set of coset representatives of $\mathbb{Z}_q$.

In both cases, if q is not prime, additional conditions on A and v assure that $m(T) \geq 0$, and also that v can be chosen in such a way that D is always an integral digit set. This is shown in greater detail in [11]. With these consecutive collinear digit sets they describe a special class of tiles, and they have several results on connectedness and disklikeness of these tiles in $\mathbb{R}^2$.

They showed the following important result, which is a stepping stone for generalization to higher dimensions.

Theorem 2.4. *Let $A \in M_2(\mathbb{Z})$ be an expanding matrix with $|\det A| = q$ (not necessarily prime). Then there exists a digit set $D = \{d_1, ..., d_q\} \subset \mathbb{Z}^2$ such that T is a connected tile.*

They also looked at characteristic polynomials and other monic polynomials coming from the dilation matrix, and used results from these to determine whether the tile generated by their digit set is connected or not. They were able to determine if the tile T is connected for particular values of $|\det A|$. They also looked at the disklikeness of a tile T. A result on disklikeness coming from their polynomial approach is as follows:

Proposition 2.5. *Let $A \in M_2(\mathbb{Z})$ be an expanding matrix with $|\det A| = q$ and characteristic polynomial $p(x) = x^2 \pm q$. Then there exists a digit set $D = \{d_1, ..., d_q\} \subset \mathbb{Z}^2$ such that $T(A, D)$ is a disklike tile.*

In the proof of this proposition, it turns out that T is always homeomorphic to a parallelogram, or 'square-like', as will be defined shortly.

Kirat, Lau, and Rao [12] reexamined consecutive collinear digit sets and generalized the previous Theorem 2.4. They described the *height reducing property*, which is also used frequently by others. They define $\mathbb{Z}[x]$ to denote the ring of polynomials with integer coefficients. They say that a monic polynomial $f(x) \in \mathbb{Z}[x]$ with $|f(0)| = q$ has the height reducing (HR) property if there exists $g(x) \in \mathbb{Z}[x]$ such that

$$g(x)f(x) = x^k + a_{k-1}x^{k-1} + ... + a_1 x \pm q,$$

with $|a_i| \leq q - 1, i = 1, ..., k - 1$.

With this in mind, they showed the following result:

Theorem 2.6. *Let $A \in M_n(\mathbb{Z})$ be an expanding matrix with $q = |\det A|$ and let $D = \{0, v, 2v, ..., (|q| - 1)v\}$ be a collinear digit set in $\mathbb{Z}^n$. Suppose the characteristic polynomial $f(x)$ of A has the HR-property, then T is connected.*

Leung and Lau [15] also looked at characteristic polynomials. The main purpose of their paper was to study the disklikeness of the special class of self-affine tiles in $\mathbb{R}^2$ that are generated by the consecutive collinear digit sets as in [11], thus providing an extension of Kirat and Lau's work. Leung and Lau looked at characteristic polynomials of the form $f(x) = x^2 + px + q$, where $|p| \leq q$ if $q \geq 2$, or $|p| \leq |q + 2|$ if $q \leq -2$, and where $A \in M_2(\mathbb{Z})$ is assumed to be an expanding matrix, and $f(x)$ is the characteristic polynomial of A.

The following is a useful result from their work:

Theorem 2.7. *Let $A \in M_2(\mathbb{Z})$ be an expanding matrix with characteristic polynomial $f(x) = x^2+px+q$. Then for any q-digit set $D = \{0, v, 2v, ..., (|q|-1)v\} \in \mathbb{Z}^2$ such that v and Av are independent, T is a disklike tile if and only if $2|p| \leq |q+2|$.*

An immediate corollary from this result is that if we have the opposite inequality we will not have a disklike tile.

A useful refinement of Kirat and Lau's result is also provided:

Theorem 2.8. *Let $A \in M_2(\mathbb{Z})$ be an expanding matrix with characteristic polynomial $f(x) = x^2 + px + q$. Suppose $0 < 2|p| \leq |q+2|$. Then the tiles $T(A, D)$ with $D = \{0, v, 2v, ..., (|q|-1)v\}$ are homeomorphic to a hexagonal tile.*

A tile can be homeomorphic to a hexagonal tile, also described as 'hexagon-like', or it can be homeomorphic to a parallelogram or square tile, also described as 'square-like'. These terms will be defined more carefully in our discussion of the work of Bandt and Wang below. So we see that only when $p = 0$ do we have a tile that is homeomorphic to a parallelogram or a square.

Akiyama and Gjini [1] also used the HR-property to show that

Theorem 2.9. *If T is a self-affine tile in $\mathbb{R}^n$, $n \leq 4$, generated by an expanding integer matrix A with $q = |\det A|$ and a collinear digit set $D = \{0, v, 2v, ..., (|q|-1)v\}$, $v \in \mathbb{Z}^n \setminus \{0\}$, then T is connected.*

Gröchenig and Haas [9] took an alternative approach with more topological results. They considered a lattice $\Gamma \subset \mathbb{R}^2$ with basis $\{e_1, e_2\}$. They defined the set $\hat{R_0} := \{\pm e_1, \pm e_2\}$, and introduced the following definition:

Definition 2.10. *A point $k \neq 0$,* pairs faces *of the digit set D, if there is an $e \in \hat{R}_0$ so that $(Ak + D) \cap (e + D) \neq \emptyset$.*

Gröchenig and Haas explore connectedness of a tile T when the digit set D is also $\hat{R}_0$-connected. Their results are quite foundational in the approach that we have taken, thus will be described in greater detail in the next chapter.

The idea of pair faces naturally gives rise to the idea of *neighbours* of our tile T, where a *neighbour*, T', of a tile, T, in a tiling means simply that $T \cap T' \neq \emptyset$. Our tile T can have both *edge neighbours* and *vertex neighbours*, where one can refer to the number of shared edges or shared vertices with the neighbours of T. So the tile T' is a neighbour to T if it shares either an edge or only a point (since $m(T \cap T') = 0$), and is thus either an edge neighbour or a vertex neighbour.

In their paper "Disk-Like Self Affine Tiles in $\mathbb{R}^2$", Bandt and Wang [5] further explore the edge and vertex neighbours of tiles. They cite the following proposition from Bandt and Gelbrich [4].

Proposition 2.11. *Let T be a topological disk which tiles $\mathbb{R}^2$ by lattice translates of the lattice Γ. Then in the tiling $T + \Gamma$ one of the following must be true:*

1. *T has no vertex neighbours and six edge neighbours $T \pm \alpha, T \pm \beta, T \pm (\alpha + \beta)$ for some $\alpha, \beta \in \Gamma$, and $\mathbb{Z}\alpha + \mathbb{Z}\beta = \Gamma$; or*

2. *T has four edge neighbours $T \pm \alpha, T \pm \beta$, and four vertex neighbours $T \pm \alpha \pm \beta$ for some $\alpha, \beta \in \Gamma$, and $\mathbb{Z}\alpha + \mathbb{Z}\beta = \Gamma$.*

In the first case, the tile T is 'hexagon-like'. In the second case the tile is 'square-like'. Keeping this in mind aids in visualizing disklike tiles in $\mathbb{R}^2$. Bandt and Wang defined a notion of connectedness for digit sets D as follows. They considered any finite subset of $\mathbb{Z}^2$, call it $\mathcal{F}$. They defined a subset $\mathcal{E} \subset \mathbb{Z}^2$ to be $\mathcal{F}$-connected if for any $u, v \in \mathcal{E}$ there exist $u_0 = u, u_1, ..., u_n = v \in \mathcal{E}$ with $u_{i+1} - u_i \in \mathcal{F}$. This is again using the idea of path-connectedness as in the work of Gröchenig and Haas.

Related results were given by Ngai and Tang [19] by looking at the neighbours of a tile T in a tiling. They describe a point p of a connected set S to be a cut point of S if $S \setminus \{p\}$ is disconnected. They prove, for a tile T in $\mathbb{R}^2$ satisfying the Open Set Condition, that if we assume that there exists a connected subset Q of T without cut points such that, for each $i = 1, ..., q$, we have that $|f_i(Q) \cap Q| \geq 2$, then T is connected, has no cut points, and the closure of each component of the interior of T, denoted by T°, is a disk.

This provides a new technique to establish the disklikeness of the closure of each component of T°. One can apply this theorem to show that the Eisenstein set [19] has no cut points; that is, that the closure of every component of its interior is a disk. The Eisenstein set is a (strictly) self-similar set defined by the IFS $f_i(x) = (1/2)R_\pi(x) + d_i$, for $i = 1, 2, 3, 4$, where R_π is the counterclockwise rotation by π and $d_1 = (0, 0),\ d_2 = (1, 0),\ d_3 = (-1/2, \sqrt{3}/2)),\ d_4 = (-1/2, -\sqrt{3}/2)$.

Also, the fundamental domain of the *quadratic canonical number system (CNS)* with base $-2+i$ has no cut points, so the closure of every component of its interior is also disklike [19]. A quadratic CNS is defined as follows. For a polynomial $p(x) = x^2 + Bx + C \in \mathbb{Z}[x]$ with $C \geq 2$ and $\mathcal{N} = \{0, ..., C - 1\}$, the pair $(p, \mathcal{N})$ is a quadratic CNS if each $q \in \mathbb{Z}[x]/p(x)\mathbb{Z}[x]$ admits a representation of the shape

$q(x) = \sum_{j=0}^{n} c_j x^j$ for some n, with $c_j \in \mathcal{N}$. To each quadratic CNS we can associate a self-affine tile. This tile is defined by $T = T(A, D)$ with $A = \begin{bmatrix} 0 & -C \\ 1 & -B \end{bmatrix}$ and $D = \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, ..., \begin{bmatrix} C-1 \\ 0 \end{bmatrix} \right\}$ (see [2]).

An alternative graph theory approach was provided by Scheicher and Thuswaldner [20]. In their setting, they let $T = T(A, D)$ be a $\mathbb{Z}^n$-tile. For $s \in \mathbb{Z}^n$ they define $B_s := T \cap (T + s)$. They defined the set of neighbours of T by

$$S := \{s \in \mathbb{Z}^n \setminus \{0\} | B_s \neq \emptyset\}.$$

They also showed that the boundary can be determined by a graph $G(S)$ on the set of neighbours. They provided an algorithm using these graphs to determine whether T is disklike or not. This paper was of key importance to this project, and so a good portion of the following chapter will be devoted to describing the graph Scheicher and Thuswaldner define, and discussing their algorithm and how they developed it.

As previously mentioned, a question concerning the topology of a self-similar tile T is the disklikeness of the closure of the components of T°, the interior of the set T. Several authors have looked at this from a variety of perspectives.

Lagarias and Wang [14] have shown the following result which aids in examining the interior and boundary of our tile T, and thus concerns the property of disklikeness.

Theorem 2.12. *Let $A \in M_m(\mathbb{R})$ be an expanding matrix such that $q = |\det A|$ is an integer, let $D \subseteq \mathbb{R}^m$ have cardinality q, and suppose $0 \in D$. Then the following four conditions are equivalent.*

1. *$T(A, D)$ has positive Lebesgue measure.*
2. *$T(A, D)$ has nonempty interior.*
3. *$T(A, D)$ is the closure of its interior T°, and its boundary $\partial T := T - T^\circ$ has Lebesgue measure 0.*
4. *For each $n \geq 1$, all q^n expansions in D_n are distinct, and D_∞ is a uniformly discrete set.*

(Note: $D_\infty = \bigcup_{n \in \mathbb{Z}} D_n$; a uniformly discrete set *has a fixed minimum distance between elements.)*

Luo, Rao and Tan [17] addressed the question of connectedness and disklikeness by examining the specific topology of a tile T in $\mathbb{R}^2$. They looked at what it meant for the boundary of T, ∂T, the interior of T, T°, and their complements to be connected and arc-wise connected if the tile itself was connected. Connectedness of T is an obvious necessary condition for T to have a connected boundary. Their following theorem provides specific results for a connected tile T in $\mathbb{R}^2$, and their paper makes suggestions for how to show a similar result in higher dimensions.

Theorem 2.13. *Let $f_1, ..., f_q$ be injective contractions on $\mathbb{R}^2$ satisfying the Open Set Condition and $T = T(f_1, ..., f_q)$ the attractor. If T is connected then we have the following statements:*

1. *T's interior T° is either empty or has no holes;*

2. *T's boundary ∂T is connected; and*

3. *whenever T° is non-empty and connected, ∂T is a simple closed curve, thus T is homeomorphic to the unit disk.*

Luo, Akiyama, and Thuswaldner [16] extended this work to higher dimensions. In particular, they developed the second and third parts of Theorem 1.1 [17]. For the boundary of a tile T, ∂T, we have the following theorem.

Theorem 2.14. *Let $f_1, ..., f_q$ be injective contractions on $\mathbb{R}^d$ $(d \geq 2)$ satisfying the Open Set Condition and $T = T(f_1, ..., f_q)$ be the attractor. Then ∂T is connected whenever T is.*

They stated the following result for the Tiling Condition as well.

Theorem 2.15. *Let $f_1, ..., f_q$ be injective contractions on $\mathbb{R}^d$ $(d \geq 2)$ and assume that the attractor T satisfies the Tiling Condition. Then ∂T is connected whenever T is.*

For the interior, T°, they provided an extension of part 3 of *Theorem 1.1* [17].

Theorem 2.16. *Let $f_1, ..., f_q$ be injective contractions on $\mathbb{R}^d$ $(d \geq 2)$ satisfying the Open Set Condition. If the attractor T is connected, then the complement of T° is connected.*

They then stated the following as a corollary to the previous result.

Theorem 2.17. *Let $f_1, ..., f_q$ be injective contractions on $\mathbb{R}^d$ $(d \geq 2)$ satisfying the Open Set Condition. If the attractor T is connected, then the complement of T° is arcwise connected.*

After these numerous results using the Open Set Condition, they proceeded to show similar results for the Tiling Condition.

Theorem 2.18. *Let T be an arcwise connected compact set satisfying the Tiling Condition. Then ∂T is connected.*

They also have the following results:

Theorem 2.19. *Assume that the compact set T satisfies the Tiling Condition. If T is connected, then the complement of T° is connected.*

Theorem 2.20. *Assume that the compact set T satisfies the Tiling Condition. If T is arcwise connected, then the complement of T° is arcwise connected.*

These results provide an idea of the breadth of the research area pertaining to tiles and their many properties. In this research project, although we did assume some of these results, we did not directly refer to them or employ the methods surveyed in this chapter. The primary source literature used will be detailed in the next chapter.

3 Background

The previous section gave a general overview of results pertaining to the connectedness and disklikeness of tiles from a variety of perspectives. In this chapter we outline a particular progression of results that illustrate our motivation for this project, as well as clarify the context of our later examples and results. We outline, specifically, definitions, results, and constructions which were especially useful for our research project, and we restate them in consistent language and notation.

As is typical in the literature, we are looking at a measurable subset T of $\mathbb{R}^m$ (or $\mathbb{Z}^m$), a lattice Γ in $\mathbb{R}^m$, and a subset Γ' of Γ. We say that the Γ'-*translates of* T *tile* $\mathbb{R}^m$ if the following two conditions are satisfied:

1. $\bigcup_{k\in\Gamma'} (k+T) = \mathbb{R}^m$,
2. $(k+T)\cap(l+T) \simeq \emptyset$ for all $k, l \in \Gamma'$, $k \neq l$.

If Γ' is a lattice, then we speak of a lattice tiling of $\mathbb{R}^m$ by T. Note that this is a restatement of the Tiling Condition. Also, note that for sets S and S', we write $S \simeq S'$ when the measurable sets S and S' are equal up to a set of measure zero.

As before, let A be a dilation matrix and let $D = \{d_1, ..., d_q\}$ be the set of digits,

where $|D| = q = |\det A|$. Since A is a dilation matrix, the sum $\sum_{j=1}^{\infty} A^{-j} d_j$ converges for any sequence of $d_j \in D$ [7]. Given an expansive linear transformation A that induces an automorphism of Γ, we say that a set T is (Γ, A)-self-similar if AT is a union of translates of T by elements of Γ [9]. We can now write $AT = \bigcup_{i=0}^{q-1} (d_i + T)$ for $d_i \in D$, with $(d_i + T) \cap (d_j + T) \simeq \emptyset$ for $d_i \in D$, $i \neq j$. Since, by our definition of a digit set, D is a complete set of coset representatives of $\mathbb{Z}^m / A(\mathbb{Z}^m)$, then the compact set $T = T(A, D) = \left\{ x \mid x = \sum_{j=1}^{\infty} A^{-j} d_j,\ d_j \in D \right\}$ is the unique (Γ, A)-self-similar set with the digit set D. Note that our tile T can be written equivalently as $T = \bigcup_{i=0}^{q-1} A^{-1}(d_i + T)$ [9].

3.1 Love thy neighbour

It has been shown by Gröchenig and Haas [9] that there is always a sub-lattice $\Gamma' \subseteq \Gamma$ such that the Γ'-translates of $T(A, D)$ tile $\mathbb{R}^m$. This concept has been adopted by other researchers in this field.

In [9], Gröchenig and Haas fix a basis $\{e_1, ..., e_m\}$ for Γ, and set $\hat{R}_0 = \{\pm e_1, ..., \pm e_m\}$. They recursively define

$$\hat{R_n} := \left\{ k \in \mathbb{Z}^m \middle| (Ak + D) \cap (l + D) \neq \emptyset \ for\ l \in \hat{R}_{n-1} \right\}$$

and set $\hat{R} = \bigcup_{n=0}^{\infty} \hat{R}_n$.

In dimension two, they classify the dilation A as *elliptic*, *parabolic*, or *hyperbolic* if A has, respectively, no real eigenvalues, one real eigenvalue, or two real eigenvalues. Using this terminology, the orientation-preserving Euclidean similarities,

that is the rigid motions, translations and rotations, are all elliptic. They further distinguish the class of rational dilations, that is, those dilations with rational eigenvalues. All parabolic and some hyperbolic dilations are of this type.

For the elliptic case, Gröchenig and Haas considered a lattice Γ with basis $\{e_1, e_2\}$. They defined a set $S \subseteq \Gamma$ to be $\hat{R}_0$-connected if, for any $k, l \in S$ there is a sequence $k = h_0, h_1, ..., h_r = l$ of elements of S, called a *path* from k to l, such that $h_{j+1} - h_j \in \hat{R}_0$ [9]. This is illustrated in Figure 3.1.

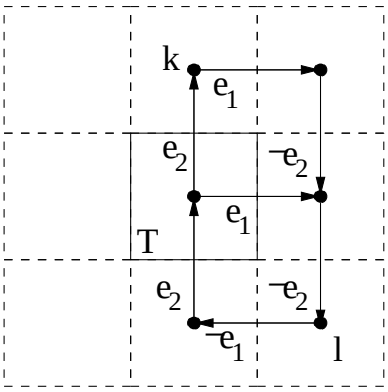


Figure 3.1: A *path* from k to l

Recall Definition 2.10 from the previous chapter. They say that $k \in \Gamma$, $k \neq 0$, *pairs faces* of the digit set D if there is an $e \in \hat{R}_0$ such that $(Ak+D) \cap (e+D) \neq \emptyset$ [9]. This is illustrated in Figure 3.2.

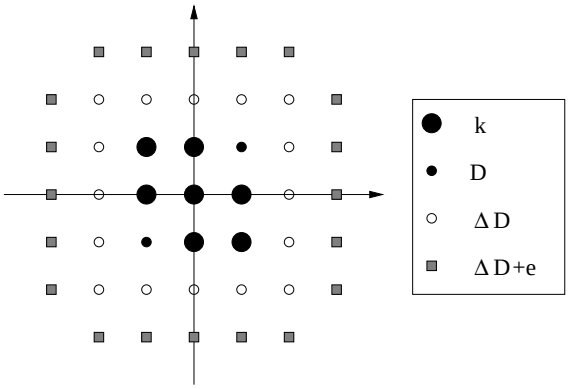


Figure 3.2: *k pairs faces* of D

Let T_0 be the parallelepiped spanned by the basis vectors of Γ. Recursively define

$$T_n := A^{-1}\Big(\bigcup_{k\in D}(T_{n-1}+k)\Big).$$

These sets are approximations to $T(A, D)$ and converge to T in the Hausdorff metric, and this iterative process is key to examining these tiles. As mentioned, we can view T as the attractor of an IFS, and the T_n arise by letting the system act on T_0.

In higher dimensions, they noted that a set S is $\hat{R}$-connected if and only if the interior of the set $\bigcup_{s\in S}(T_0+s)$ is a connected subset of $\mathbb{R}^m$.

They also extend the definition of pairs faces to say that a point $k \in \Gamma$, $k \neq 0$, *pairs faces* of T_n if there is an $e \in \hat{R}_0$ such that $(A^n k + D_n) \cap (e + D_n) \neq \emptyset$, that is, T_n and $T_n + k$ translated by a digit in D have an edge in common, and thus are neighbours. This is illustrated in Figure 3.3. They showed that k pairs faces of T_n if and only if $k \in \hat{R}_n$. They claimed that k pairs faces of T_n, if and only if $T_n \cap (T_n + k)$ is a non-empty $(m-1)$-dimensional piecewise linear sub-manifold of

$\mathbb{R}^m$. Note that 'pairs faces of the digit set D' and 'pairs faces of T_n' are distinct ideas.

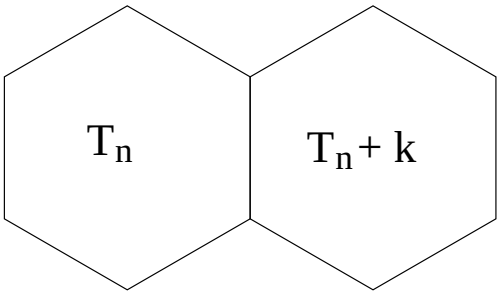


Figure 3.3: *k pairs faces* of T_n

We see that if a tile, $T + k$, pairs faces with T, then $T + k$ is a *neighbour* of our tile T, so that $T \cap (T + k) \neq \emptyset$. The tile $T + k$ is a neighbour to T if it shares either an edge or only a point, and is thus either an edge neighbour or a vertex neighbour.

Gröchenig and Haas explored connectedness of the tile T when the digit set D is also connected. They have results such as the following:

Theorem 3.1. *(cf. [[9], Theorem 2.5])) Let* $\{e_1, e_2\}$ *be a basis for* Γ *(a lattice of* $\mathbb{R}^2$*) and* D *be an* $\hat{R}_0$*-connected set of digits so that* e_1 *and* e_2 *both pair faces of* D*. Then* $T(A, D)$ *is a connected set and the* Γ*-translates of* $T(A, D)$ *tile* $\mathbb{R}^2$*.*

This result is generalized by Kirat and Lau to a general criterion for connectedness [11] which is further developed in this thesis.

The remaining focus of this section is to outline some of the fundamental ideas, approaches and results on tilings and connectedness of tilings that are prevalent in all related literature.

The following three results are imperative in showing that $T(A, D)$ always tiles. The first lemma presented refers to the notation introduced above, where Γ is the lattice in $\mathbb{R}^m$ that is invariant under multiplication by the dilation matrix A. (In our case, $\Gamma = \mathbb{Z}^m$, though the following three results are stated in greater generality.)

Lemma 3.2. *(cf. [[9], Lemma 3.1])*

1. $|D_n| = q^n$, *and for* $l \in \Gamma, l \neq 0$ *we have*

$$(A^n l + D_n) \cap D_n = \emptyset \quad \textit{and} \quad \bigcup_{k \in \Gamma} (A^n k + D_n) = \Gamma.$$

2. $A^n T_n = \bigcup_{k \in D_n} (k + T_0)$ *and* $A^n T(A, D) = \bigcup_{k \in D_n} (T(A, D) + k)$.

3. *The* Γ*-translates of* T_n *tile* $\mathbb{R}^m$ *for all* $n \geq 0$.

4. T_n *converges to* $T(A, D)$ *in the compact open topology.*

5. *The* Γ*-translates of* $T(A, D)$ *cover* $\mathbb{R}^m$*;* $\bigcup_{k \in \Gamma} (T(A, D) + k) = \mathbb{R}^m$.

Set

$$D_n := \left\{ k \in \Gamma \middle| k = \sum_{j=0}^{n-1} A^j d_j,\ d_j \in D \right\}.$$

Also, for a given set $S \subseteq \mathbb{R}^m$, denote the difference set by $\Delta S = \{x = s - s' | s, s' \in S\}$.

Proposition 3.3. *(cf. [[9], Proposition 3.2]) Let* D *be a set of digits. Then* $T(A, D)$ *is a* Γ*-tile if and only if* $\bigcup_{n+1}^{\infty} \Delta D_n = \Gamma$.

An immediate corollary from this result is the following:

Corollary 3.4. *(cf. [[9], Corollary 3.3]) Let D be a digit set for which $T(A, D)$ is a Γ-tile. Then the set $\{A^j(d_i - d_0),\ j \geq 0, i = 0, ..., q-1\}$ generates Γ.*

The following results are further criteria for lattice tiling. There is an important measure-theoretic counterpart to the disjointness condition, $(k+T) \cap (l+T) \simeq \emptyset$ for all $k, l \in \Gamma$, $k \neq l$, in the definition of a tiling. We already know that the Γ-translates of the self-similar set $T(A, D) \subseteq \mathbb{R}^m$ cover $\mathbb{R}^m$ (cf. [[9], Lemma 3.1]). Let T_0 be the parallelepiped spanned by a basis of Γ. Since a Γ-tile is a fundamental domain for Γ as a sub-lattice of $\mathbb{R}^m$ and the volume of a fundamental domain is an invariant, we have from [9] the following proposition:

Proposition 3.5. *(cf. [[9], Proposition 4.1]) $\lambda(T(A, D)) \geq \lambda(T_0)$ and the Γ-translates of $T(A, D)$ tile $\mathbb{R}^m$ if and only if $\lambda(T(A, D)) = \lambda(T_0)$.*

Here λ refers to the m-dimensional Lebesgue measure on $\mathbb{R}^m$. This tiling criterion is used frequently in the literature.

The following lemma shows that the $\hat{R}_n$ eventually stabilize, from which we infer that $\hat{R}$ is finite. This further supports that viewing the system as an IFS where T is the attractor is a useful perspective both computationally and conceptually.

Lemma 3.6. *(cf. [[9], Lemma 4.5])*

1. $\hat{R}_n = \hat{R}_n^*$, *where* $\hat{R}_n^* = \left\{ l \in \mathbb{Z}_0^m \middle| (A^n l + D_n) \cap (f + D_n) \neq \emptyset \text{ for } f \in \hat{R}_0 \right\}$, *where* $\mathbb{Z}_0^m = \mathbb{Z}^m \setminus \{0\}$.
2. $\hat{R}$ *is the smallest set such that* $\hat{R}_0 \subset \hat{R}$ *and* $D \cup (\hat{R} + D) \supseteq D \cup (A\hat{R} + D)$.
3. $D_n \cup (\hat{R} + D_n) \subseteq D_n \cup (A^n \hat{R} + D_n)$ *for* $n \geq 1$.

4. $\hat{R}$ is finite.

We can now begin to look at connected lattice tilings. The following result was an important starting point for generalizing Theorem 2.5 [9]:

Lemma 3.7. *(cf. [[9], Lemma 7.1]) Suppose that D is $\hat{R}_0$-connected and each $e \in \hat{R}_0$ pairs faces of T_1. Then, for each integer $n > 0$, D_n is $\hat{R}_0$-connected and each $e \in \hat{R}_0$ pairs faces of T_n.*

In the proof of this lemma, Gröchenig and Haas illustrate how to take a path in D_{n+1}, and show that it is $\hat{R}_0$-connected by taking any path in D_n (where D_n is $\hat{R}_0$-connected), and directly building on the path in D_n, that is, concatenating the two paths.

The remaining results are in $\mathbb{R}^2$, but exhibit the types of results we are interested in. Here

$$\Delta_0\hat{R}_0 = \{l_0 := e_1, l_1 := e_1 + e_2, l_2 := e_2, l_3 := e_2 - e_1, l_4 := -e_1,$$
$$l_5 = -e_1 - e_2, l_6 := -e_2, l_7 = -e_2 + e_1, l_8 := e_1\},$$

where the l_i are analogous to edges of $\hat{R}_0$ that are directed from the point that they are defined by, so that they form a cycle from e_1 back to e_1 again.

Proposition 3.8. *(cf. [[9], Proposition 7.2]) Suppose that F is a compact subset of $\mathbb{R}^2$ with piecewise linear boundary, and that both e_1 and e_2 pair faces of F. If $k \in \Gamma$ also pairs faces of F then $k \in \Delta\hat{R}_0$.*

Corollary 3.9. *(cf. [[9], Corollary 7.3]) Suppose that D is $\hat{R}_0$-connected and that e_1 and e_2 pair faces of T_1. Then $R \subset \Delta_0\hat{R}_0$.*

This next result is a step towards generalizing $\hat{R}_0$-connectedness to higher dimensions.

Lemma 3.10. *(cf. [[9], Lemma 7.4]) Suppose that D is $\hat{R}_0$-connected and each element of $\hat{R}_0$ pairs faces of T_1. Then $\Delta D_n \supset \Delta \hat{R}_0$ for $n \geq 4$.*

To summarize: the idea of pairs faces naturally gives rise to the idea of *neighbours* of our tile T, where a *neighbour* of a tile T in a tiling means simply a tile having a common boundary with T. Our tile T can have both *edge neighbours*, where the boundary between T and a neighbour contains infinitely many points, and *vertex neighbours*, where the boundary contains only one point.

3.2 And who is thy neighbour?

Kirat and Lau [11] also looked at connectedness of a tile T in $\mathbb{R}^m$. Some of their specific results for a tile in $\mathbb{R}^2$ are described in the previous section, but a particular result was of main interest.

They give a general criterion for connectedness by using a 'graph' argument on D. In their setting, they say that given a pair (A, D), we can define

$$S = \{(d_i, d_j) : (T + d_i) \cap (T + d_j) \neq \emptyset,\ d_i, d_j \in D\}$$

to be the set of 'edges' for the set D. They say that d_i and d_j are S-connected if there exists a finite sequence $\{d_{j_1}, ..., d_{j_k}\} \subseteq D$ such that $d_{j_1} = d_i,\ d_{j_k} = d_j$ and $(d_{j_1}, d_{j_k}) \in S,\ 1 \leq l \leq k-1$. Let $\Delta D := D - D$. They then give the following proposition:

Proposition 3.11. *(cf. [[11], Proposition 4.2]) Let $A \in M_m(\mathbb{Z})$ be expanding and let $D \subset \mathbb{R}^m$ be a digit set. Then $(d_i, d_j) \in S$ if and only if $d_i - d_j = \sum_{k=1}^{\infty} A^{-k} v_k$ with $v_k \in \Delta D$.*

The following result is the above-mentioned generalization of Gröchenig and Haas [9], and is a general criterion for connectedness.

Theorem 3.12. *(cf. [[11], Theorem 4.3]) Let $A \in M_m(\mathbb{Z})$ be an expanding matrix with $q = |\det A|$ and let $D = \{d_1, ..., d_q\} \subseteq \mathbb{R}^m$ be a q-digit set. Then T is connected if and only if for any two $d_i, d_j \in D$, d_i and d_j are S-connected.*

Kirat and Lau looked at skew matrices A as examples, and have several results demonstrating the important properties of this type of matrix. It is important to consider these matrices, as examples indicate that it is only very skewed matrices that will give rise to a disconnected digit set D and tile $T(A, D)$. This has been explored further in our research; the following theorem provides motivation for later results.

Theorem 3.13. *(cf. [[11], Theorem 3.4]) Suppose $A \in M_2(\mathbb{Z})$ is an expanding matrix with a repeated eigenvalue λ. Then λ is an integer and A is $\mathbb{Z}$-similar to a matrix of the form* $\begin{bmatrix} \lambda & a \\ 0 & \lambda \end{bmatrix}$.

Two matrices A and B are said to be $\mathbb{Z}$-similar if there exists a *unimodular* matrix $P^* \in GL_n(\mathbb{Z})$ such that $(P^*)^{-1} A P^* = B$. A unimodular matrix is a square integer matrix with determinant ± 1. Equivalently, it is an integer matrix that is invertible over the integers. Note that this is a stronger condition than simply saying that A and B are similar matrices.

In the two dimensional case, skew matrices $A = \begin{bmatrix} \lambda & k \\ 0 & \lambda \end{bmatrix}$ are similar to a Jordan form matrix $\begin{bmatrix} \lambda & 1 \\ 0 & \lambda \end{bmatrix}$, where the fact that λ must be an integer is implied by Theorem 3.4 [11], and can also be inferred from the form of the characteristic polynomial of A.

3.3 One should please their neighbour for their good, to build them up

The paper that we referred to the most is a graph theory approach to looking at tiles provided by Scheicher and Thuswaldner [20]. They viewed tiles as graphs, in a similar manner as Kirat and Lau [11], and showed how to actually find the neighbours from this graph. They provided an algorithm using these graphs to determine the set of neighbours S, which in turn can indicate whether T is disklike or not.

They explicitly defined a graph on a tile as follows:

- Let T be a $\mathbb{Z}^m$-tile. For $s \in \mathbb{Z}^m$, they let $B_s := T \cap (T + s)$, and they defined the set of neighbours of T by $S := \{s \in \mathbb{Z}^m \setminus \{0\} \,|B_s \neq \emptyset\}$. Note that S is a finite set because T is compact. One can write $\partial T = \bigcup_{s \in S} B_s$. It turns out that the boundary of T forms a *graph-directed system*. Recall, that a *directed graph* or *digraph* is an ordered pair $D := (V, E)$ with a set V whose elements are called vertices or nodes, and a set E of ordered pairs

of vertices, called arcs, directed edges, or arrows.

- They then labelled the elements of S as $S = \{s_1, ..., s_r\}$ and defined the graph $G(S) = (V, E)$ with the set of states $V := S$ in the following way. They let $E_{i,j}$ be the set of edges leading from s_i to s_j:

$$E_{i,j} := \left\{ s_i \xrightarrow{d} s_j \mid As_i + d' = s_j + d \text{ for some } d, d' \in D \right\}.$$

They showed that the boundary can be determined by the graph $G(S)$. One can write the boundary of a tile T, as ∂T. In fact, $\partial T = \bigcup_{s \in S} B_s$. It turns out that ∂T forms a *graph-directed system* [8].

From this definition of $G(S)$, Scheicher and Thuswaldner provided the following result.

Proposition 3.14. *(cf. [[20], Proposition 2.2]) ∂T is a graph-directed system determined by the graph $G(S)$. In particular, $\partial T = \bigcup_{s \in S} B_s$ where*

$$B_s = \bigcup_{\substack{d \in D, s' \in S \\ s \xrightarrow{d} s'}} A^{-1}(B_{s'} + d).$$

The union is extended over all d, s' such that $s \xrightarrow{d} s'$ is an edge in the graph $G(S)$.

From this we can say that the boundary can be determined by the graph $G(S)$. Unfortunately though, this approach has the disadvantage that it involves the graph $G(S)$ which is not so easy to construct. On the other hand, $G(S)$ is interesting because knowing this graph implies knowledge of all neighbours of T. This is of great interest for deciding whether T is connected and disklike or not.

As mentioned, Scheicher and Thuswaldner have shown that there is also a second graph $G(R)$ which determines the boundary of T. This graph is a subgraph of $G(S)$, as will be shown. $G(R)$ has the advantage that it is easier to compute than $G(S)$, but there is no way to read the set S of neighbours of T directly from this graph. However, $G(R)$ can be used to construct $G(S)$. In order to do this, they defined the reduced graph of a graph G, denoted by $Red(G)$, where $Red(G)$ is the graph constructed from G by removing all states that are not the starting point of a walk of infinite length.

In order to describe the subgraph $G(R)$, as well as other components of this graph directed system, we need some additional definitions. Let T_0 be a parallelepiped spanned by a basis of the lattice $\mathbb{Z}^m$ and set

$$T_n = \bigcup \left\{ A^{-n}T_0 + x \middle| x \in \sum_{i=1}^{n} A^{-i}D \right\} \ (n \in \mathbb{N}).$$

Notice that $\lim_{n\to\infty} T_n = T$ in the Hausdorff metric, and also that T_n tiles $\mathbb{R}^m$ under translation by $\mathbb{Z}^m$ for each $n \in \mathbb{N}$.

Let $\{e_1, ..., e_m\}$ be a basis of the lattice $\mathbb{Z}^m$, set $R_0 := \{0, \pm e_1, ..., \pm e_m\}$, and define R_n recursively by

$$R_n := \left\{ k \in \mathbb{Z}^m \middle| Ak + D \cap l + D \neq \emptyset \ for \ l \in R_{n-1} \right\}.$$

If $R = \bigcup_{n\geq 0} R_n$ then R is a finite set. In particular, we have $R_{n-1} = R_n$ for n large enough (cf. [[9], Section 4]). Note that $R_0 \setminus \{0\} = \hat{R}_0$, $R_n \setminus \{0\} = \hat{R}_n$, and $R = \hat{R}$, from Gröchenig and Haas. Also, T_0 is the parallelepiped used in [[9], Section 3].

Now we define the graph $G(R) = (V, E)$ with set of states R in the following way. There exists an edge $r \xrightarrow{d} r'$ from r to r' labelled by d if $Ar + d' = r' + d$ holds for some $d' \in D$.

Before we state the algorithm, we give some definitions which will be used throughout the remainder of this thesis.

Definition 3.15. *(cf. [[20], Definition 3.2]) Let $G(\mathbb{Z}^m)$ be a labelled directed graph with set of states $\mathbb{Z}^m$ and set of labels $D \times D$ whose elements are written as $l|l'$. The labelled edges connecting two states s_1 and s_2 are defined by $s_1 \xrightarrow{l|l'} s_2$ if and only if $As_1 - s_2 = l - l'$ $(s_1, s_2 \in \mathbb{Z}^m,\ l, l' \in D)$. Since l' is uniquely determined by s_1, s_2 and A, we sometimes omit it and write just $s_1 \xrightarrow{l} s_2$ for the edges in $G(\mathbb{Z}^m)$ and its subgraphs. A subgraph of $G(\mathbb{Z}^m)$ with set of states V will be denoted by $G(V)$.*

Let G be a subgraph of $G(\mathbb{Z}^m)$. For abbreviation we will write $s_1 \xrightarrow{l|l'} s_2 \in E(G)$ if $s_1 \xrightarrow{l|l'} s_2$ is an edge in G. Note that $G(R)$ and $G(S)$ are subgraphs of $G(\mathbb{Z}^m)$. Thus the notation of the previous graphs agrees with the above definition.

Definition 3.16. *(cf. [[20], Definition 3.3]) Let $M \subset \mathbb{Z}^m$. We say that a subgraph $G(M)$ of $G(\mathbb{Z}^m)$ has property (C) if:*

- *(**C**) for each pair $(s_2, l) \in M \times D$ there exists a unique pair $(s_1, l') \in M \times D$ such that $s_1 \xrightarrow{l|l'} s_2$ is an edge of $G(M)$.*

Property (C) says that we have predecessors for all of the states of the graph $Red(G(R))$.

Definition 3.17. *(cf. [[20], Definition 3.4]) Let G be a graph. We denote by $Red(G)$ the graph that emerges from G if all states of G which are not the starting point of a walk of infinite length are removed.*

Definition 3.18. *(cf. [[20], Definition 3.5]) Let G_1 and G_1' be subgraphs of $G(\mathbb{Z}^m)$. The product $G_2 := G_1 \otimes G_1'$ is defined in the following way. Let r_1, s_1 be states of G_1 and r_1', s_1' be states of G_1'. Furthermore, let $l_1, l_1', l_2 \in D$.*

- *r_2 is a state of G_2 if $r_2 = r_1 + r_1'$*
- *There exists an edge $r_2 \overset{l_1|l_1'}{\to} s_2$ in G_2 if there exist*
  $r_1 \overset{l_1|l_1'}{\to} s_1 \in E(G_1)$ and $r_1' \overset{l_1'|l_2}{\to} s_1' \in E(G_1)$
  with $r_1 + r_1' = r_2$ and $s_1 + s_1' = s_2$ or there exist
  $r_1 \overset{l_1'|l_2}{\to} s_1' \in E(G_1)$ and$r_1' \overset{l_1|l_1'}{\to} s_1' \in E(G_1)$
  with $r_1 + r_1' = r_2$ and $s_1 + s_1' = s_2$.

Furthermore, if G is a subgraph of $G(\mathbb{Z}^m)$, we use the abbreviation

$$G^p := \otimes_{j=1}^{p} G = \underbrace{G \otimes ... \otimes G}_{p-times}.$$

The algorithm now reads as follows.

Algorithm 3.19. *(cf. [[20], Algorithm 3.6]) The graph $G(S)$, and with it the set S, can be determined by the following algorithm starting from the graph $G(R)$.*

p := 1

A[1] := Red(G(R))

repeat

p := p + 1

$A[p] := Red(A[p-1] \otimes A[1])$

until A[p] = A[p-1]

$G(S) := A[p] \setminus \{0\}$

This algorithm always terminates after finitely many steps.

For proving that this algorithm yields $G(S)$ after finitely many steps we need a list of preparatory results. To demonstrate the relation between $G(R)$ and $G(S)$, Scheicher and Thuswaldner used a series of arguments bounding $G(S)$. In particular, they 'bound' $G(S)$ from below and from above in terms of $G(R)$. They begin with a very easy result on $G(R)$.

Lemma 3.20. *(cf. [[20], Lemma 4.1]) Each state $r \in R'$ of the graph $G(R') := Red(G(R))$ has infinitely many predecessors and infinitely many successors. Thus $G(R')$ is a union of cycles of $G(\mathbb{Z}^m)$ and of walks connecting two of these cycles. Furthermore, $G(R')$ has property (C).*

The fact that each $r \in R'$ has infinitely many successors follows from the definition of $Red(\cdot)$. They showed that each $r \in R'$ has a predecessor, and then the result follows inductively. By the definition of $G(R')$, we have to find an $r' \in \mathbb{Z}^m$ with $r+d = Ar'+d'$ $(d, d' \in D)$. The definition of R implies that such an r' is contained in R and that it is a predecessor of r in $G(R)$. But since $r \in R'$, r has infinitely many successors in $G(R)$. Since r' is a predecessor of r in $G(R)$, the same is true for r'. This implies that $r' \in R'$ as well, and r' is a predecessor of r in $G(R')$.

They also showed that there exists an $r' \in \mathbb{Z}^m$ with $r + d = Ar' + d'(d, d' \in D)$, and used the fact that D is a complete set of coset representatives of $\mathbb{Z}^m/A(\mathbb{Z}^m)$ to show that for each pair $(r, d) \in R' \times D$, there exists a unique $d' \in D$ such

that $r + d' \equiv d' \mod (A(\mathbb{Z}^m))$. Hence, there exists a unique $r' \in \mathbb{Z}^m$ such that $r + d = Ar' + d'$. Thus $r' \in R'$ is a predecessor of r. The assertion concerning the structure of $G(R')$ follows immediately since R' is a finite set.

The next result shows a similar property of the graph $G(S \cup \{0\})$.

Lemma 3.21. *(cf. [[20], Lemma 4.2]) The graph $G(S \cup \{0\})$ is the union of all cycles of $G(\mathbb{Z}^m)$ and all walks connecting two of these cycles.*

The above lemmas have the following consequence.

Corollary 3.22. *(cf. [[20], Corollary 4.3]) $Red(G(R)) \subset G(S \cup \{0\})$.*

This corollary indicates that all the vertices in $Red(G(R))$ are neighbours, just that we may not yet have all of them. This is how we begin to construct $G(S)$ starting from $G(R') := Red(G(R))$. We have to be sure that the set R' contains a basis of the lattice $\mathbb{Z}^m$. The following lemma takes care of this.

Lemma 3.23. *(cf. [[20], Lemma 4.4]) Let $G(R') := Red(G(R))$. Then the set R' contains a basis $\{e'_1, ..., e'_m\}$ of the lattice $\mathbb{Z}^m$. By symmetry and because $0 \xrightarrow{0} 0$ is a cycle in $G(R)$ it even contains a set of the shape $\{0, \pm e'_1, ... \pm e'_m\}$.*

Scheicher and Thuswaldner showed that each of the e_j can be written as a $\mathbb{Z}$-linear combination of elements of R' in the following way:

$$e_j = t_1 + \sum_{i=2}^{l} (t_i - t_{i-1}) + (e_j - t_l),$$

where $t_1, t_2 - t_1, ..., t_l - t_{l-1}, e_j - t_l \in R'$, and where the t_i are $\mathbb{Z}^m$-translates of our tile T_n.

Note that, by Lemma 3.23, in the construction of the graph $G(R)$ we can always select the basis $\{e_1, ..., e_m\}$ in a way such that $Red(G(R)) = G(R)$, that is, $R' = R$.

Lemma 3.24. *(cf. [[20], Lemma 4.6]) Let G_1 and G_2 be subgraphs of $G(\mathbb{Z}^m)$ having property (C). Then $H := Red(G_1 \otimes G_2)$ has property (C) and there exists an edge $m_1 \overset{d|d'}{\to} m_2$ in $G_1 \otimes G_2$ if and only if $Am_1 + d' = m_2 + d$.*

The preceding lemmas have the following result as a consequence. It complements Corollary 3.22 (cf. [[20], Corollary 4.3]).

Corollary 3.25. *(cf. [[20], Corollary 4.7]). There exists a positive integer p_0 such that $Red(G(R)^{p_0}) \supset G(S \cup 0)$. Furthermore, $Red(G(R)^{p_0})$ has property (C).*

There are further lemmas and propositions on taking multiple products (multiple $\otimes$'s), as well as taking multiple reductions, and also multiple reductions of products and so on. These are procedures that need to be shown as valid to justify Algorithm 3.19.

After these preparations, they finally showed that Algorithm 3.19 yields $G(S)$, and does so in finitely many steps.

3.4 Summary

A progression of results has been made. To start, Gröchenig and Haas set up a general framework for research in this area and introduced the concept of 'neighbours' of a tile. Then, Kirat and Lau suggested that a graph method could

describe the set of neighbours, which would tell us more about connectedness. Scheicher and Thuswaldner provided a precise way of describing the graph of a tile T, and provided an explicit algorithm to find the neighbours of our tile. Further results on connectedness and disklikeness are linked to these results as they are either results stemming from this work in a similar vein, or they are results that still hold true in this setting.

All of the results outlined in this chapter were both motivating and frequently used over the course of this project.

4 Results

We have implemented Sheicher and Thuswaldner's *Algorithm 3.6* [20] as our Algorithm 3.19, and have used the knowledge of the neighbours to re-examine connectedness. We have also used the software from Sage to view, graphically, the level sets D_n.

4.1 Results on Connectedness

The following lemma has never been explicitly stated in the literature, so we separate it out in this exposition for clarity.

Lemma 4.1. *Suppose that T_n is a sequence of compact, connected subsets of $\mathbb{R}^m$, and that in the Hausdorff metric $T = \lim_{n\to\infty} T_n$. Then T is connected.*

Proof. We know that $T = \lim_{n\to\infty} T_n$ in the Hausdorff metric. That is, given $\epsilon > 0$, there exists N such that $h(T_n, T) = \max\{d(T, T_n), d(T_n, T)\} < \epsilon$ for all $n > N$. This means that both $d(T, T_n) < \epsilon$ and $d(T_n, T) < \epsilon$. Applying the definition of distance from one set to another, this means that $\max_{x\in T} d(x, T_n) < \epsilon$ and $\max_{y\in T_n} d(y, T) < \epsilon$.

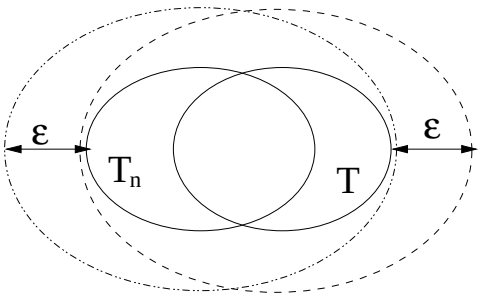


Figure 4.1: $h(T_n, T) < \epsilon$ for all $n > N$.

Suppose that T is closed and compact but not connected. Then there exists a *separation* of T, that is, two open sets U and V such that $T \subseteq U \cup V$, $T \cap U \neq \emptyset$, $T \cap V \neq \emptyset$, and $T \cap U \cap V = \emptyset$. We construct a specific separation of T.

Since T is not connected, it has at least two connected components. Let C be a component of T. Then C is closed and compact as well as connected. Let $B = T \setminus C$. B is also closed and compact, though not necessarily connected (B may be the union of multiple components if T has three or more connected components).

Note that in $\mathbb{R}^m$ there is always a non-zero, positive distance between nearest points of two compact sets. In particular,there exist δ_1, δ_2 such that for all $c \in C$, $d(c, B) \geq \delta_1 > 0$ and for all $b \in B$, $d(b, C) \geq \delta_2 > 0$. Thus $h(B, C) \geq \max\{\delta_1, \delta_2\} > 0$. Let $\delta = min\{\delta_1, \delta_2\}$; then for all $c \in C$, $b \in B$, $d(c, b) \geq \delta > 0$.

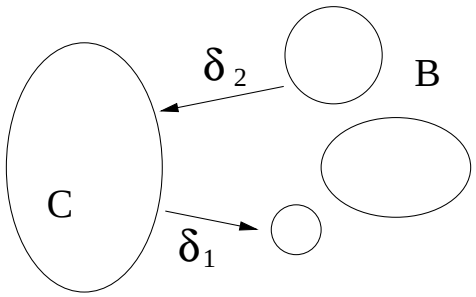


Figure 4.2: For all $c \in C$, $d(c, B) \geq \delta_1 > 0$ and for all $b \in B$, $d(b, C) \geq \delta_2 > 0$.

We will construct a specific separation of T based on the sets C and B. In order to do this, we must first describe a particular subset of $\mathbb{R}^m$. Let A be any subset of $\mathbb{R}^m$. For any $\epsilon > 0$, define $A + \epsilon := \{x \in \mathbb{R}^m \mid d(x, A) < \epsilon\}$.

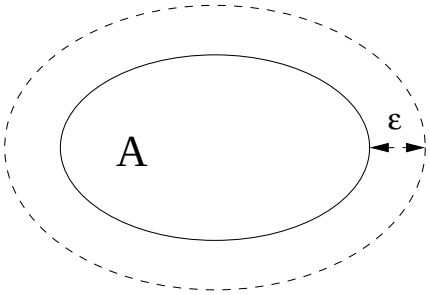


Figure 4.3: $A + \epsilon := \{x \in \mathbb{R}^m \mid d(x, A) < \epsilon\}$.

Set $\epsilon = \delta/4$, and let $U = C + \epsilon$ and $V = B + \epsilon$. Note that U and V are both open. We can compute that $d(U, V) > \delta/2$ and that $d(V, U) > \delta/2$ (see Figure 4.4). Thus $U \cap V = \emptyset$, and therefore $T \cap U \cap V = \emptyset$. Note also that $T \cap U \neq \emptyset$ because $T \cap U \supset C$, and that $T \cap V \neq \emptyset$ because $T \cap V \supset B$. Thus U and V provide a separation of T where, specifically, $h(U, V) \geq \delta/2 = 2\epsilon > 0$.

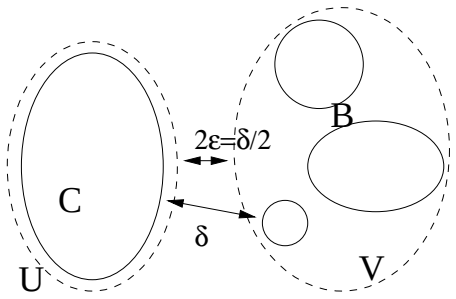


Figure 4.4: U and V provide a separation of T where $h(U,V) \geq \delta/2 = 2\epsilon > 0$.

Since $\lim_{n\to\infty} T_n = T$, there exists an index N such that

$$h(T_n, T) = \max\{d(T, T_n), d(T_n, T)\} < \epsilon \text{ for all } n > N,$$

and therefore both $d(T, T_n) < \epsilon$ and $d(T_n, T) < \epsilon$. This implies that $T \subset T_n + \epsilon$ and that $T_n \subset T + \epsilon$. Since $T + \epsilon = (C + \epsilon) \cup (B + \epsilon) = U \cup V$, one of the following two cases must apply. Case one is that $T_n \cap U \neq \emptyset$ and $T_n \cap V \neq \emptyset$. This implies that U and V are a separation of T_n, contradicting that T_n is connected. Case two is that $T_n \cap U = \emptyset$ and $T_n \cap V \neq \emptyset$ (or that $T_n \cap V = \emptyset$ and $T_n \cap U \neq \emptyset$; without loss of generality we can pick the first case). Then

$$h(T_n, T) = \max\{(d(T_n, T), d(T, T_n)\} \geq d(T_n, U) \geq \delta/2 = 2\epsilon > 0$$

contradicting that $h(T_n, T) < \epsilon$.

In either case we get a contradiction, thus the assumption that T is disconnected must be false. Therefore T is connected. □

Note that this theorem holds for arbitrary sets T satisfying the hypotheses. In our case, we see that the tile $T = T(A, D)$ is connected if there exists an N such

that T_n is connected for all $n > N$.

This following result was motivated by the experimental data from the Maple program which implemented Sheicher and Thuswaldner's *Algorithm 3.6*. It is a refinement of Kirat and Lau's result (cf. [[11], Theorem 4.3]), but we came to this result independently over the course of this research project. It became clear that if $S \cap \Delta D$ contained a basis for $\mathbb{Z}^m$, then the digit set would be lattice connected with this basis, and thus the tile T would be connected.

Proposition 4.2. *Let S be the set of neighbours of $T = T(A, D)$, and B a basis of $\mathbb{Z}^m$ such that $B \subset S$. If D is B-connected, then T is connected.*

Proof. To show that T is connected, we first show that if any subset Q of $\mathbb{R}^m$ that is congruent to the unit cube is connected, then $A^{-1}(Q + D)$ is connected.

Let $d, d' \in D$. We want to show that there exists a sequence from $Q+d$ to $Q+d'$. That is $d = d_1, ..., d_r = d'$, such that $(Q + d_j) \cap (Q + d_{j+1}) \neq \emptyset$. This means that the path from $Q + d$ to $Q + d'$ is a path through neighbour translates of our tile (these are translates by elements of S).

We know that for any $d, d' \in D$, there exists a path $d = d_1, ..., d_r = d'$, such that $d_{j+1} - d_j \in B$, because D is B-connected. Also $B \subset S$ implies that $Q \cap (Q+b) \neq \emptyset$ for all $b \in B$. Thus $Q \cap (Q + d_{j+1} - d_j) \neq \emptyset$, and so $(Q + d_j) \cap (Q + d_{j+1}) \neq \emptyset$. We see that this same sequence gives the path that we need to connect $Q + d$ and $Q + d'$. Thus $Q + D$ is connected whenever Q is connected, and therefore $A^{-1}(Q + D)$ is connected.

Let T_0 be the unit cube in $\mathbb{R}^m$, a connected set. Let $T_{n+1} = A^{-1}(T_n + D)$, for each

integer $n \geq 0$. Note that these are the same as the sets T_n studied in Sheicher and Thuswaldner [20], who observe that $\lim_{n\to\infty} T_n = T$ in the Hausdorff metric.

By induction, with $Q = T_n$ for each $n \geq 0$, we see that each T_n is connected. Then by Lemma 4.1, $T = \lim_{n\to\infty} T_n$ is connected as well. □

We can also talk about connectedness of the level sets D_n. The following lemma shows that there is a correspondence between D_n being connected and T_n being connected.

Lemma 4.3. *The set T_n is connected if and only if the level set D_n is lattice connected.*

Proof. Let Γ be a lattice under which T tiles $\mathbb{R}^m$. Let F be a fundamental domain centered at the origin for the lattice Γ, such that the Lebesgue measure of F is equal to the Lebesgue measure of T, that is, $\lambda(F) = \lambda(T)$. Suppose that D is Γ-connected.

Let

$$\begin{aligned} T_0 &= F \\ T_1 &= A^{-1}(T_0 + D) \\ &= A^{-1}(F + D) \\ T_2 &= A^{-1}(T_1 + D) \\ &= A^{-1}(A^{-1}(F + D) + D) \end{aligned}$$

$$= A^{-2}(F) + A^{-2}(D) + A^{-1}(D)$$

$$\vdots$$

$$T_n = A^{-n}(F) + A^{-n}(D) + ... + A^{-2}(D) + A^{-1}(D),$$

where we know that $A^{-n}(F)$ converges to 0 as $n \to \infty$.

Applying A^n to T_n, we have the following equality:

$$\begin{aligned} A^n T_n &= F + D + ... + A^{n-2}(D) + A^{n-1}(D) \\ &= F + D_n. \end{aligned}$$

So we see that $A^n T_n$ is connected if and only if D_n is lattice connected. The lemma follows, since $A^n T_n$ is connected if and only if T_n is connected, and $F + D_n$ is connected if and only if D_n is Γ-connected. □

4.2 Results in Two Dimensions

For a moment, let us consider arbitrary two dimensional matrices,

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}.$$

For any dilation matrix A in $M_2(\mathbb{Z})$, there is an invertible matrix $P \in M_2(\mathbb{C})$ such that $A = PJP^{-1}$, where J is the Jordan form of the matrix A. As an example,

recall the $\mathbb{Z}$-similar matrices described by Kirat and Lau [11]. In general, however, A and J will only be similar, not $\mathbb{Z}$-similar.

We have the following three cases for J:

1. A has two distinct or non-distinct (but with one dimensional eigenspaces) integer eigenvalues λ_1 and λ_2, so that

$$J = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}.$$

2. A has two distinct non-integer eigenvalues r_1 and r_2, so that

$$J = \begin{bmatrix} r_1 & 0 \\ 0 & r_2 \end{bmatrix},$$

   where $r_1 = a + b\sqrt{c}$, and $r_2 = a - b\sqrt{c}$ with $a, b, c \in \mathbb{Q}$.

3. A has a single integer eigenvalue λ with a two dimensional eigenspace, so that

$$J = \begin{bmatrix} \lambda & 1 \\ 0 & \lambda \end{bmatrix}.$$

We do not have a fourth case, where J is of the form of the third case, but where λ is not an integer, since such an eigenvalue cannot be the root of a quadratic polynomial with integer coefficients (see also *Theorem 3.4*, from Kirat and Lau[11]).

For matrices of the form $A = \begin{bmatrix} \lambda & k \\ 0 & \lambda \end{bmatrix}$ that Kirat and Lau considered and that

comprised the majority of our examples, the Jordan form will always be as in the third case.

Although we have been denoting the digit set for a matrix A as simply D throughout this thesis, for the following discussion and two results, we will give every digit set D a subscript indicating which matrix it is a digit set for. Thus, the digit set for the matrix A will now be D_A.

For the following two results we are looking at matrices of the following form:

$$J = \begin{bmatrix} \lambda & 1 \\ 0 & \lambda \end{bmatrix}$$

We want to use an inductive argument to show that $D_{J,n}$ is lattice connected for every $n \geq 0$. The base case is $D_{J,1} = D$, which is lattice connected. As mentioned previously, for our examples we are using the fundamental domain for $\mathbb{Z}^m$, $F = \left(-\frac{1}{2}, \frac{1}{2}\right]^m$. Thus in $\mathbb{Z}^2$, $D_J = \left[-\left\lfloor \frac{\lambda-1}{2} \right\rfloor, \left\lfloor \frac{\lambda}{2} \right\rfloor\right] \cap \mathbb{Z}^2$. If λ is an even integer, $D_J = \left[-\frac{\lambda}{2}+1, \frac{\lambda}{2}\right]^2 \cap \mathbb{Z}^2$; while if λ is odd, $D_J = \left[-\frac{\lambda-1}{2}, \frac{\lambda-1}{2}\right]^2 \cap \mathbb{Z}^2$.

In determining a relationship between the level sets $D_{J,n}$ and $D_{J,(n-1)}$, we considered the values of the lattice points a_0, b_0, c_0, d_0, and e_0 in $D_{J,1} = D$. These points are shown in Figure 4.5 where $\lambda = 4$, and Figure 4.6 for $\lambda = 3$. We will view the lattice points a_0, b_0, c_0, d_0, and e_0 as vectors, and refer to the top and the bottom entries as shown in equation 4.1. The subscripts $x_{1,*}$ and $x_{2,*}$ will be useful notation for clearly indicating which entry of our vectors we are looking at, as well as generalizing our results to higher dimensions. The general formulas

(when λ even and when λ odd) for these points are shown in Table 4.1.

$$a_0 = \begin{bmatrix} a_{1,0} \\ a_{2,0} \end{bmatrix} \; b_0 = \begin{bmatrix} b_{1,0} \\ b_{2,0} \end{bmatrix} \; c_0 = \begin{bmatrix} c_{1,0} \\ c_{2,0} \end{bmatrix} \; d_0 = \begin{bmatrix} d_{1,0} \\ d_{2,0} \end{bmatrix} \; e_0 = \begin{bmatrix} e_{1,0} \\ e_{2,0} \end{bmatrix} \tag{4.1}$$

	λ even	λ odd
a_0	$\begin{bmatrix} \frac{\lambda}{2} \\ \frac{\lambda}{2} \end{bmatrix}$	$\begin{bmatrix} \frac{\lambda-1}{2} \\ \frac{\lambda-1}{2} \end{bmatrix}$
b_0	$\begin{bmatrix} \frac{\lambda}{2} \\ -\frac{\lambda}{2}+1 \end{bmatrix}$	$\begin{bmatrix} \frac{\lambda-1}{2} \\ -\left(\frac{\lambda-1}{2}\right) \end{bmatrix}$
c_0	$\begin{bmatrix} -\frac{\lambda}{2}+1 \\ \frac{\lambda}{2} \end{bmatrix}$	$\begin{bmatrix} -\left(\frac{\lambda-1}{2}\right) \\ \frac{\lambda-1}{2} \end{bmatrix}$
d_0	$\begin{bmatrix} -\frac{\lambda}{2}+1 \\ -\frac{\lambda}{2}+1 \end{bmatrix}$	$\begin{bmatrix} -\left(\frac{\lambda-1}{2}\right) \\ -\left(\frac{\lambda-1}{2}\right) \end{bmatrix}$
e_0	$\begin{bmatrix} -\frac{\lambda}{2}+1 \\ 1 \end{bmatrix}$	$\begin{bmatrix} -\left(\frac{\lambda-1}{2}\right) \\ 1 \end{bmatrix}$

Table 4.1: Starting values for a_0, b_0, c_0, d_0, and e_0, for λ even and odd.

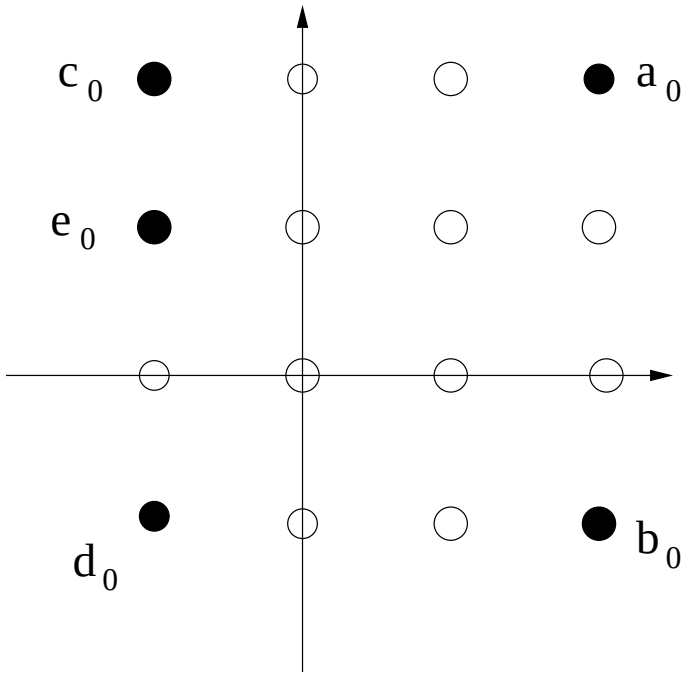


Figure 4.5: Starting values for a_0, b_0, c_0, d_0, and e_0, for $\lambda = 4$.

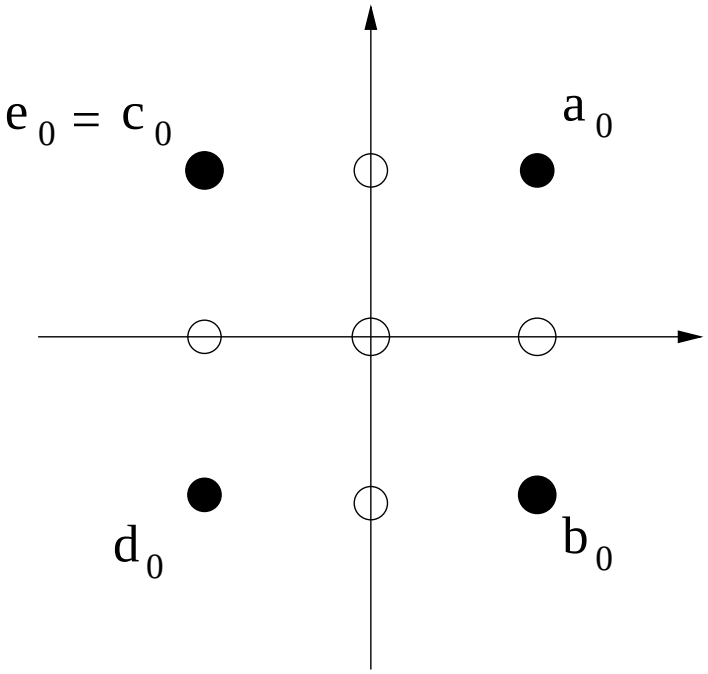


Figure 4.6: Starting values for a_0, b_0, c_0, d_0, and e_0, for $\lambda = 3$.

For the inductive step, we suppose that $D_{J,(n-1)}$ is lattice connected, and show that then $D_{J,n}$ is lattice connected. To do this, we look at the points a_n, b_n, c_n, d_n, and e_n. To show that $D_{J,n}$ is lattice connected whenever $D_{J,(n-1)}$ is lattice connected, we need to show that horizontal rows of digits in $D_{J,n}$ have sufficient overlap. This requires comparing the point e_n in $D_{J,n}$ with the point c_{n-1} in $D_{J,(n-1)}$.

We find that

$$c_1 = \begin{bmatrix} \lambda & 1 \\ 0 & \lambda \end{bmatrix} \cdot c_0 + c_0,$$

and

$$c_n = \begin{bmatrix} \lambda & 1 \\ 0 & \lambda \end{bmatrix} \cdot c_{n-1} + c_0.$$

Similarly,

$$e_1 = \begin{bmatrix} \lambda & 1 \\ 0 & \lambda \end{bmatrix} \cdot e_0 + d_0,$$

and

$$e_n = \begin{bmatrix} \lambda & 1 \\ 0 & \lambda \end{bmatrix} \cdot e_{n-1} + d_0.$$

Figure 4.7 shows c_1 and e_1, and c_2 and e_2 for $\lambda = 4$. Figure 4.8 shows c_1 and e_1, and c_2 and e_2 for $\lambda = 3$. Table 4.2 shows the explicit calculation and form of c_0, c_1, c_2, and general c_n and e_0, e_1, e_2, and general e_n for λ even and odd.

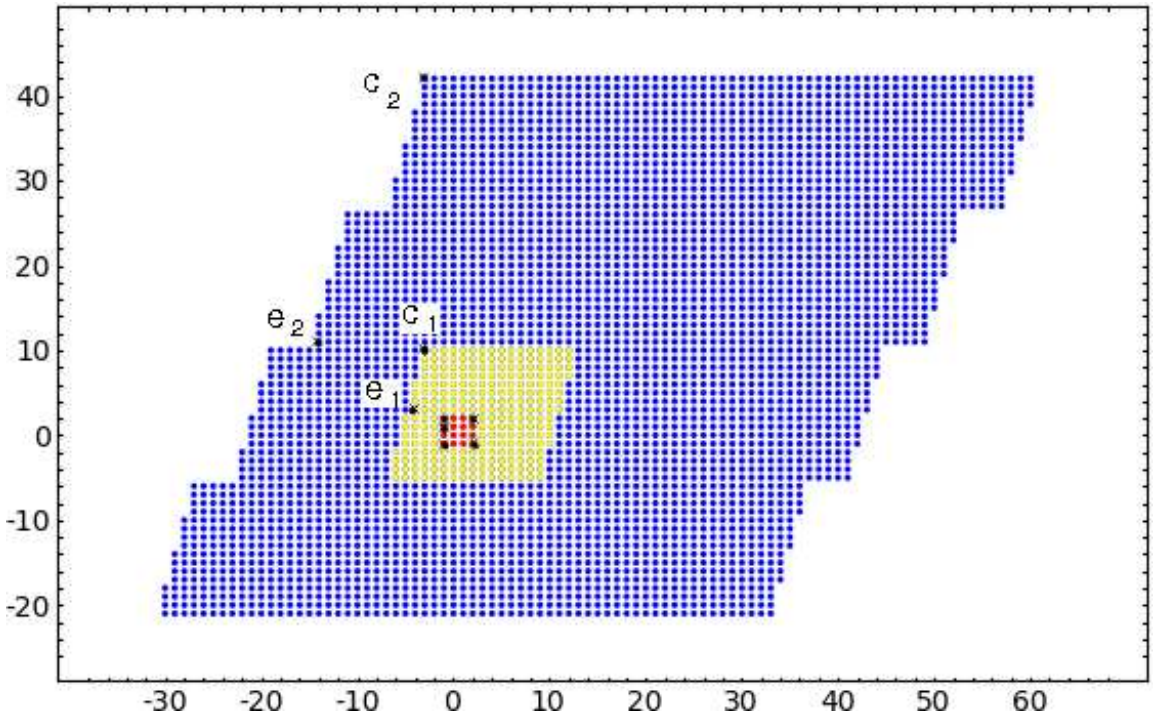


Figure 4.7: c_1 and e_1, and c_2 and e_2 for $\lambda = 4$.

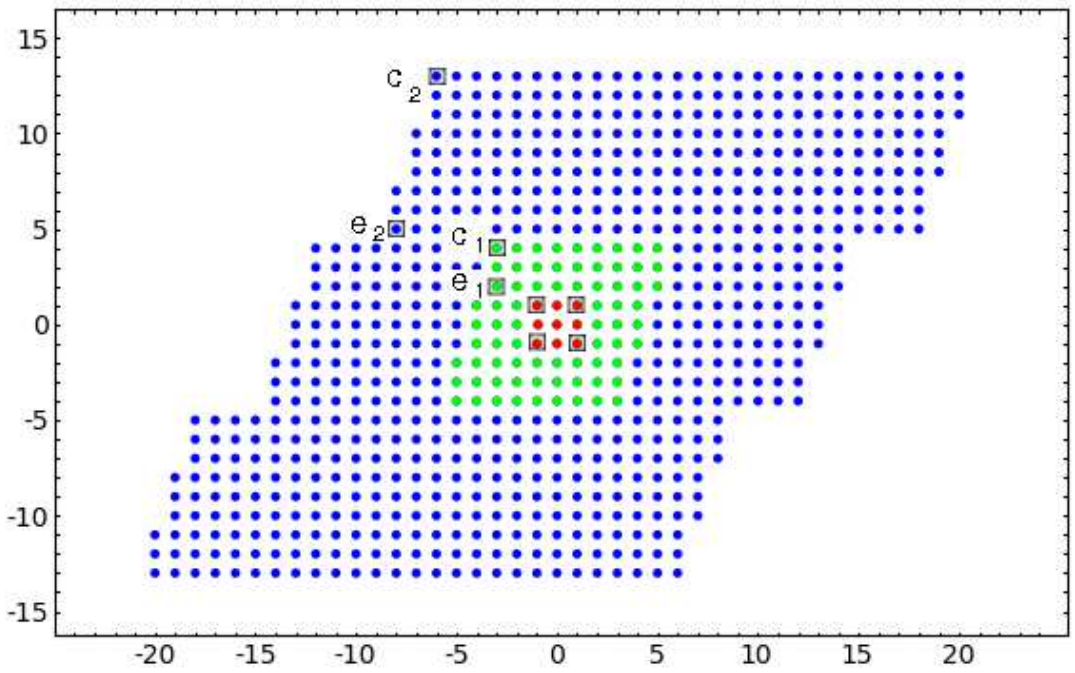


Figure 4.8: c_1 and e_1, and c_2 and e_2 for $\lambda = 3$.

	λ even	λ odd
c_0	$\frac{1}{2}\begin{bmatrix} -\lambda+2 \\ \lambda \end{bmatrix}$	$\frac{1}{2}\begin{bmatrix} -\lambda+1 \\ \lambda-1 \end{bmatrix}$
c_1	$\frac{1}{2}\begin{bmatrix} -\lambda^2+2\lambda+2 \\ \lambda^2+\lambda \end{bmatrix}$	$\frac{1}{2}\begin{bmatrix} -\lambda^2-\lambda \\ \lambda^2-1 \end{bmatrix}$
c_2	$\frac{1}{2}\begin{bmatrix} -\lambda^3+3\lambda^2+2\lambda+2 \\ \lambda^3+\lambda^2+\lambda \end{bmatrix}$	$\frac{1}{2}\begin{bmatrix} -\lambda^3+2\lambda^2-\lambda \\ \lambda^3-1 \end{bmatrix}$
$\vdots$	$\vdots$	$\vdots$
c_n	$\frac{1}{2}\begin{bmatrix} -\lambda^{n+1}+(n+1)\lambda^n+n\lambda^{n-1}+...+2\lambda+2 \\ \lambda^{n+1}+\lambda^n+...+\lambda^2+\lambda \end{bmatrix}$	$\frac{1}{2}\begin{bmatrix} -\lambda^{n+1}+n\lambda^n-\lambda^{n-1}-\lambda^{n-2}-...-\lambda^2-\lambda \\ \lambda^{n+1}-1 \end{bmatrix}$
e_0	$\frac{1}{2}\begin{bmatrix} -\lambda+2 \\ 1 \end{bmatrix}$	$\frac{1}{2}\begin{bmatrix} -\lambda+1 \\ 1 \end{bmatrix}$
e_1	$\frac{1}{2}\begin{bmatrix} -\lambda^2+\lambda+4 \\ \lambda+2 \end{bmatrix}$	$\frac{1}{2}\begin{bmatrix} -\lambda^2+3 \\ \lambda+1 \end{bmatrix}$
e_2	$\frac{1}{2}\begin{bmatrix} -\lambda^3+\lambda^2+4\lambda+4 \\ \lambda^2+\lambda+2 \end{bmatrix}$	$\frac{1}{2}\begin{bmatrix} -\lambda^3+3\lambda+2 \\ \lambda^2+1 \end{bmatrix}$
$\vdots$	$\vdots$	$\vdots$
e_n	$\frac{1}{2}\begin{bmatrix} -\lambda^{n+1}+\lambda^n+(n+2)\lambda^{n-1}+(n+1)\lambda^{n-2}+...+4\lambda+4 \\ \lambda^n+\lambda^{n-1}+...+\lambda+2 \end{bmatrix}$	$\frac{1}{2}\begin{bmatrix} -\lambda^{n+1}+(n+1)\lambda^{n-1}+\lambda^{n-2}+...+\lambda+2 \\ \lambda^n+1 \end{bmatrix}$

Table 4.2: Values of c_0, c_1, c_2, and c_n and e_0, e_1, e_2, and e_n for λ even and odd.

Although we can look at both $c_{1,n}$ and $c_{2,n}$, as well as $e_{1,n}$ and $e_{2,n}$, we always have $e_{2,(n+1)} - c_{2,n} = 1$, $e_{2,(n+1)} - e_{2,n} > 0$, and $c_{2,(n+1)} - c_{2,n} > 0$, so we only want to look at the relationship between $c_{1,n}$ and $e_{1,(n+1)}$.

This relationship is described in the proof of the following result.

Lemma 4.4. *For the Jordan matrices of the form* $J = \begin{bmatrix} \lambda & 1 \\ 0 & \lambda \end{bmatrix}$, *where* $\lambda \geq 3$ *is an integer, each* $D_{J,n}$ *is lattice connected.*

Proof. To show this, we use an inductive argument as indicated in the preceeding discussion. We noted already that $D_{J,1} = D_J$ is lattice connected. For the inductive step, we suppose that $D_{J,(n-1)}$ is lattice connected. To show that D_n is lattice connected, we will first show that $e_{1,n} < c_{1,(n-1)}$, that is, $c_{1,(n-1)} - e_{1,n} > 0$. This will be shown for both the even and odd case.

For λ even, when $n = 0$, $c_{1,0} = \frac{1}{2}(-\lambda + 2)$, and $e_{1,0} = \frac{1}{2}(-\lambda^2 + \lambda + 4)$, we see that

$$c_{1,0} - e_{1,0} = \frac{1}{2}(-\lambda + 2) - \frac{1}{2}\left(-\lambda^2 + \lambda + 4\right) = \frac{1}{2}\left(\lambda^2 - 2\lambda - 2\right),$$

where for $\lambda \geq 4$, this will always be greater than 0.

Suppose this holds up to $n - 1$, so that $c_{1,n-2} - e_{1,(n-1)} > 0$. Now

$$c_{1,(n-1)} = \frac{1}{2}\left(-\lambda^n + (n)\lambda^{n-1} + \lambda^{n-2} + ... + 2\lambda + 2\right),$$

and

$$e_{1,n} = \frac{1}{2}\left(-\lambda^{n+1} + \lambda^n + (n+2)\lambda^{n-1} + (n+1)\lambda^{n-2} + ... + 4\right),$$

and so

$$\begin{aligned} c_{1,(n-1)} - e_{1,n} &= \frac{1}{2}\left(\lambda^{n+1} - 2\lambda^n - 2\lambda^{n-1} - ... - 2\lambda - 2\right) \\ &= \frac{1}{2}\left[\lambda^{n+1} - 2\left(\frac{\lambda^{n+1}-1}{\lambda-1}\right)\right]. \end{aligned}$$

Now $c_{1,(n-1)} - e_{1,n}$ can be written as

$$c_{1,(n-1)} - e_{1,n} = \frac{1}{2}\left[\frac{\lambda^{n+1}(\lambda-3)+2}{\lambda-1}\right],$$

where for $\lambda \geq 4$, this is always greater than 0.

For λ odd, when $n = 0$, $c_{1,0} = \frac{1}{2}(-\lambda - 1)$, and $e_{1,0} = \frac{1}{2}(-\lambda^2 + 3)$, we see that

$$c_{1,0} - e_{1,0} = \frac{1}{2}(-\lambda - 1) - \frac{1}{2}\left(-\lambda^2 + 3\right) = \frac{1}{2}\left(\lambda^2 - \lambda - 4\right),$$

where for $\lambda \geq 3$, this will always be greater than 0.

Suppose this holds up to $n-1$, so that $c_{1,n-2} - e_{1,(n-1)} > 0$. Now

$$c_{1,(n-1)} = \frac{1}{2}\left(-\lambda^n + (n-1)\lambda^{n-1} - \lambda^{n-2} - ... - \lambda^2 - \lambda\right),$$

and

$$e_{1,n} = \frac{1}{2}\left(-\lambda^{n+1} + (n+1)\lambda^{n-1} + \lambda^{n-2} + ... + \lambda + 2\right),$$

and so

$$\begin{aligned} c_{1,(n-1)} - e_{1,n} &= \frac{1}{2}\left(\lambda^{n+1} - \lambda^n - 2\lambda^{n-1} - ... - 2\lambda - 2\right) \\ &= \frac{1}{2}\left[\lambda^{n+1} - \lambda^n - 2\left(\frac{\lambda^n - 1}{\lambda - 1}\right)\right]. \end{aligned}$$

Now $c_{1,(n-1)} - e_{1,n}$ can be written as

$$c_{1,(n-1)} - e_{1,n} = \frac{1}{2}\left[\frac{\lambda^n(\lambda^2 - 2\lambda + 1) + 2}{\lambda - 1}\right],$$

where for $\lambda \geq 3$, this is always greater than 0.

For λ even or odd, we also have that $-c_{1,(n-1)} + e_{1,n} > 0$. This indicates that horizontal rows of digits of D_n always overlap as shown in Figure 4.7 and Figure 4.8, and thus D_n is lattice connected. □

Combining Lemmas 4.1, 4.3, and 4.4, we may state the following theorem:

Theorem 4.5. *For matrices of the form* $J = \begin{bmatrix} \lambda & 1 \\ 0 & \lambda \end{bmatrix}$ *, such that* $\lambda \geq 3$ *is an integer, we have that* $T(J, D_J)$ *is connected.*

For $\lambda > 1$, the matrix J will also be a dilation matrix, but will always have the 'block' digit set $D_J = \left(-\left\lfloor \frac{\lambda}{2} \right\rfloor, \left\lfloor \frac{\lambda}{2} \right\rfloor\right] \cap \mathbb{Z}^2$. D_J is thus the canonical digit set for J.

We would like to consider a different digit set for A, however. Set $d_i = Pg_i$ for each $g_i \in D_J$, that is, set $D_A := PD_J$. Then we can rewrite the tile $T = T(A, D_A)$

in the following way:

$$\begin{aligned} T(A, D_A) &= \sum_{i=1}^{\infty} A^i d_i \quad \text{for } d_i \in D_A \\ &= \sum_{i=1}^{\infty} (PJP^{-1})^i d_i \\ &= \sum_{i=1}^{\infty} PJ^iP^{-1} d_i \\ &= P\left(\sum_{i=1}^{\infty} J^i (P^{-1} d_i)\right) \\ &= P\big(T(J, D_J)\big). \end{aligned}$$

The next lemma and theorem follow from this discussion.

Lemma 4.6. *The tile $T(A, D_A)$ is connected if and only if $T(J, D_J)$ is connected.*

Proof. We saw that $T(A, D_A) = P(T(J, D_J))$, where P is an invertible linear transformation. The mapping defined by P is thus continuous, and gives a homeomorphism between $T(A, D_A)$ and $T(J, D_J)$ where connectedness is preserved. □

Conjecture 4.7. *There exists a digit set D_A for which $T(A, D)$ is connected.*

Cases 1, 3, and part of 2. As shown above we can always find $D_A = PD_J$. It remains to show that $T(J, D_J)$ is always connected.

Recall, for matrices $A \in M_2(\mathbb{Z})$ there are three possibilities for its Jordan form. In the first case, A has two distinct or non-distinct (but with one dimensional

eigenspaces) integer eigenvalues λ_1 and λ_2, and so

$$J = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}.$$

Here we have two linearly independent eigenvectors $\vec{e}_1$ and $\vec{e}_2$. Thus, for any $\vec{x} \in \mathbb{R}^2$, we have that $\vec{x} = c_1\vec{e}_1 + c_2\vec{e}_2$ for $c_1, c_2 \in \mathbb{R}$. We see that $A^n\vec{x} = {\lambda_1}^n(c_1\vec{e}_1) + {\lambda_2}^n(c_2\vec{e}_2)$. The digit set D_J will still be the 'block' centered around 0, only now a rectangle, that is, $D_J = \left(\left(-\left\lfloor\frac{\lambda_1}{2}\right\rfloor, \left\lfloor\frac{\lambda_1}{2}\right\rfloor\right] \times \left(-\left\lfloor\frac{\lambda_2}{2}\right\rfloor, \left\lfloor\frac{\lambda_2}{2}\right\rfloor\right]\right) \cap \mathbb{Z}^2$, or $D_J = \left(\left[-\left\lfloor\frac{\lambda_1-1}{2}\right\rfloor, \left\lfloor\frac{\lambda_1}{2}\right\rfloor\right] \times \left[-\left\lfloor\frac{\lambda_2-1}{2}\right\rfloor, \left\lfloor\frac{\lambda_2}{2}\right\rfloor\right]\right) \cap \mathbb{Z}^2$. To be precise,

$$D_J = \left\{ \begin{bmatrix} -\left\lfloor\frac{\lambda_1-1}{2}\right\rfloor \\ -\left\lfloor\frac{\lambda_2-1}{2}\right\rfloor \end{bmatrix}, ..., \begin{bmatrix} \left\lfloor\frac{\lambda_1}{2}\right\rfloor \\ -\left\lfloor\frac{\lambda_2-1}{2}\right\rfloor \end{bmatrix}, \begin{bmatrix} -\left\lfloor\frac{\lambda_1-1}{2}\right\rfloor \\ -\left\lfloor\frac{\lambda_2-1}{2}\right\rfloor + 1 \end{bmatrix}, ..., \begin{bmatrix} \left\lfloor\frac{\lambda_1}{2}\right\rfloor \\ -\left\lfloor\frac{\lambda_2-1}{2}\right\rfloor + 1 \end{bmatrix}, ..., \begin{bmatrix} -\left\lfloor\frac{\lambda_1-1}{2}\right\rfloor \\ \left\lfloor\frac{\lambda_2-1}{2}\right\rfloor \end{bmatrix}, ..., \begin{bmatrix} \left\lfloor\frac{\lambda_1}{2}\right\rfloor \\ \left\lfloor\frac{\lambda_2-1}{2}\right\rfloor \end{bmatrix} \right\}.$$

Since the eigenspaces for λ_1 and λ_2 are orthogonal, we can now look at each one-dimensional component of our tile T. For λ_1 or λ_2 even,

$$D_{\lambda_1} = \left\{-\frac{\lambda_1 - 2}{2}, ..., \frac{\lambda_1}{2}\right\} \text{ or } D_{\lambda_2} = \left\{-\frac{\lambda_2 - 2}{2}, ..., \frac{\lambda_2}{2}\right\}.$$

Then

$$\begin{aligned} T_{\lambda_1} &= \left\{ \sum_{i=1}^{\infty} \frac{d_i}{\lambda_1}^i \ : \ d_i \in D_{\lambda_1} \right\} \\ &= \left[\sum_{i=1}^{\infty} \frac{-\left(\frac{\lambda_1 - 2}{2}\right)}{\lambda_1^i}, \sum_{i=1}^{\infty} \frac{\left(\frac{\lambda_1}{2}\right)}{\lambda_1^i} \right] \\ &= \left[-\frac{1}{2}\left(\frac{\lambda_1}{\lambda_1 - 1} - \frac{2}{\lambda_1 - 1}\right), \frac{1}{2}\left(\frac{\lambda_1}{\lambda_1 - 1}\right) \right] \\ &= \left[-\frac{1}{2}\left(\frac{\lambda_1 - 2}{\lambda_1 - 1}\right), \frac{1}{2}\left(\frac{\lambda_1}{\lambda_1 - 1}\right) \right], \end{aligned}$$

where it is now clear that T_{λ_1} is closed and connected, has measure equal to 1, and is centered about 0 for λ_1 even. Similarly T_{λ_2} is a closed and connected measure 1 interval centered at 0 for λ_2 even.

For λ_1 or λ_2 odd,

$$D_{\lambda_1} = \left\{ -\frac{\lambda_1 - 1}{2}, ..., \frac{\lambda_1}{2} \right\} \text{ or } D_{\lambda_2} = \left\{ -\frac{\lambda_2 - 2}{2}, ..., \frac{\lambda_2}{2} \right\}.$$

Then

$$\begin{aligned} T_{\lambda_1} &= \left\{ \sum_{i=1}^{\infty} \frac{d_i}{\lambda_1}^i \ : \ d_i \in D_{\lambda_1} \right\} \\ &= \left[\sum_{i=1}^{\infty} \frac{-\left(\frac{\lambda_1 - 1}{2}\right)}{\lambda_1^i}, \sum_{i=1}^{\infty} \frac{\left(\frac{\lambda_1 - 1}{2}\right)}{\lambda_1^i} \right] \\ &= \left[-\frac{1}{2}\left(\frac{\lambda_1}{\lambda_1 - 1} - \frac{1}{\lambda_1 - 1}\right), \frac{1}{2}\left(\frac{\lambda_1}{\lambda_1 - 1} - \frac{1}{\lambda_1 - 1}\right) \right] \\ &= \left[-\frac{1}{2}\left(\frac{\lambda_1 - 1}{\lambda_1 - 1}\right), \frac{1}{2}\left(\frac{\lambda_1 - 1}{\lambda_1 - 1}\right) \right] \\ &= \left[-\frac{1}{2}, \frac{1}{2} \right], \end{aligned}$$

where it is now clear that T_{λ_1} is closed and connected, has measure equal to 1, and is centered at 0 for λ_1 odd. Similarly T_{λ_2} is a closed and connected measure

1 interval centered at 0 for λ_2 odd.

So we see that $T(J, D_J) = T_{\lambda_1} \times T_{\lambda_2}$ is connected for λ_1 and λ_2 even or odd. Therefore by Theorem 4.6, $T(A, D_A)$ is connected with $D_A = PD_J$.

In the second case, A has two distinct non-integer eigenvalues r_1 and r_2, so that

$$J = \begin{bmatrix} r_1 & 0 \\ 0 & r_2 \end{bmatrix},$$

where $r_1 = a + b\sqrt{c}$, and $r_2 = a - b\sqrt{c}$ with $a, b, c \in \mathbb{Q}$. We then say that r_1 and r_2 are a conjugate pair, either real (when $c > 0$) or complex (when $c < 0$).

When r_1 and r_2 are a complex conjugate pair, that is $r_1 = \alpha + \beta i$ and $r_2 = \alpha - \beta i$ for $\alpha, \beta \in \mathbb{R}$, we see that J is similar to the rotation matrix

$$\rho = \begin{bmatrix} \alpha & -\beta \\ \beta & \alpha \end{bmatrix}.$$

whose eigenvalues are $\alpha \pm \beta i$. That is $J = S\rho S^{-1}$, where ρ is a mapping from $\mathbb{R}^2 \to \mathbb{R}^2$. Let $r = \sqrt{\alpha^2 + \beta^2}$; then we can write ρ as follows:

$$\rho = r \begin{bmatrix} \frac{\alpha}{r} & \frac{-\beta}{r} \\ \frac{\alpha}{r} & \frac{\beta}{r} \end{bmatrix} = \begin{bmatrix} r & 0 \\ 0 & r \end{bmatrix} \cdot \begin{bmatrix} cos\theta & -sin\theta \\ cos\theta & sin\theta \end{bmatrix},$$

so we see that R just scales and rotates.

We may choose $F_\rho = \left(-\frac{1}{2}, \frac{1}{2}\right]^2$, and we have that

$$D_\rho = \rho(F_\rho) \cap \mathbb{Z}^2.$$

Note that $r^2 = q$, so D_ρ has the correct number of digits and is the 'block' set of digits centered around 0. The level set $D_{\rho,n} = \rho^n(F_\rho) \cap \mathbb{Z}^2$, so is just a dilated and rotated 'block' of digits. Thus the $D_{\rho,n}$ is lattice connected, which tells us that $\rho^n T_{\rho,n}$ is connected and therefore so is $T_{\rho,n}$. Thus $T(\rho, D_\rho)$ is a connected tile.

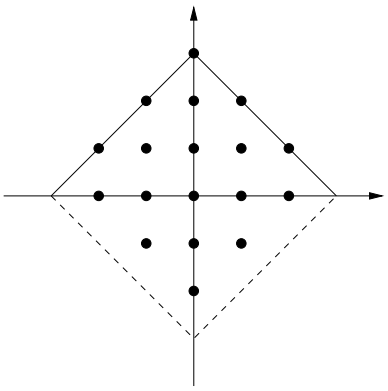


Figure 4.9: $D_\rho = \rho(F_\rho) \cap \mathbb{Z}^2$ dilated and rotated.

Now since J is similar to ρ, we know that $T(J, D_J)$ is connected, with $D_J = SD_\rho$. Also, as A is similar to J and thus similar to ρ, that is $A = PJP^{-1} = P(S\rho S^{-1})P^{-1}$, we have that $T(A, D_A)$ is connected with $D_A = PD_J = P(SD_\rho)$.

When r_1 and r_2 are a real conjugate pair our experimental evidence indicates that we can find a digit set D_A such that $T(A, D_A)$ is connected similarly as in the complex conjugate case, however, details remain to be worked out.

Finally, in the third case, A has a single integer eigenvalue λ with a two dimensional eigenspace, so that

$$J = \begin{bmatrix} \lambda & 1 \\ 0 & \lambda \end{bmatrix}.$$

As shown in Theorem 4.5, for J in this form we have that the tile $T(J, D_J)$ is always connected, and therefore by Lemma 4.6, $T(A, D_A)$ is connected with $D_A = PD_J$. □

While we have not yet completed the proof of this conjecture, we believe we have provided a strong argument for using similar matrices to investigate topological properties of $T(A, D)$.

5 Examples and Tools

5.1 New Computational Tools for Finding Neighbours

In this section we describe the implementation of the neighbour finding algorithm of Scheicher and Thuswaldner. Implementing this algorithm was a significant contribution to the field resulting from this research project. We also developed new visualization tools to facilitate our research.

5.1.1 Maple

Scheicher and Thuswaldner [20] defined two graphs $G(S)$ and $G(R)$ related to a $\mathbb{Z}^m$ tile $T = T(A, D)$. $G(S)$ is the graph on the set of neighbours S, of T. $G(R)$ is a subgraph of $G(S)$, that is useful and important in constructing $G(S)$.

They provided the following algorithm to construct $G(S)$ from $G(R)$.

Algorithm 5.1. *(cf. [[20], Algorithm 3.6])) The graph $G(S)$, and with it the set S, can be determined by the following algorithm starting from the graph $G(R)$.*

p := 1

A[1] := Red(G(R))

repeat

$p := p + 1$

$A[p] := Red(A[p-1] \otimes A[1])$

until A[p] = A[p-1]

$G(S) := A[p] \setminus \{0\}$

This algorithm always terminates after finitely many steps.

Since this algorithm always runs in finite time, it can be coded on a computer and used as a tool to more easily investigate properties of a tile T, such as neighbours, pairs faces, connectivity, and even disk-likeness. However, after contacting both Scheicher and Thuswaldner, as well as some of their collaborators, we found that this algorithm had not yet been implemented.

We coded this algorithm in Maple, and have designed it to be used like a Maple package, such as `Student[Calculus1]` and others. The package '`neighbours`', contains the following commands: `Alg36`, `CheckIn`, `RedG`, `Rprime`, `alg36`, `cart`, `condfunc`, `cube2ds`, `cube3ds`, `cube4ds`, `deltaDS`, `genRn`, `makeall`, `makeanylist`, `makeanyvector`, `makelist`, `makevector`, `pairsfaces`, `parelist`, `prod2GR`, `prodnGR`, `subDS`, and `sumDS`. Some of these are key procedures that are elements of the end algorithm, while others are smaller sub-procedures. They are all available separately so that we can examine subsets of the set of neighbours along the way. Table 5.1 gives a brief description of each command. More detailed explanations are provided in Appendix A.

Table 5.1: Maple Procedures for `neighbours` package.

Command	Description
`cart`	gives the Cartesian product as a single list
`deltaDS`	outputs the difference set, ΔD, of the digit set D
`sumDS`	gives the set resulting from all possible combinations of adding two elements (each from a different set)
`CheckIn`	compares a set of vectors to a single vector, and checks equality
`parelist`	removes redundancies from a set of vectors
`genRn`	determines R_n as in [20]
`makeall`	generates the graph from R_n
`pairsfaces`	finds R_1
`condfunc`	determines if fourth element in a list is a leaf of $G(R)$
`RedG`	iteratively removes leaves from $G(R)$
`Rprime`	picks off the fourth element of a list that is not a leaf
`prod2GR`	takes the product of two graphs G_1 and G_2
`makelist`	converts a list of 4×1-dimensional vectors into a list of four-tuple lists
`makeanylist`	makes a list of a $n \times 1$-dimensional vector into a list of n-tuple lists
`makevector`	makes a list of four-tuple lists into a list of 4×1-dimensional vectors
`makeanyvector`	makes a list of n-tuple lists into a list of $n \times 1$-dimensional vectors
`alg36`	finds the set of neighbours as four-tuples
`Alg36`	finds the set of neighbours, but is `Rprime` of `alg36`
`subDS`	like `deltaDS` but takes the difference of two sets
`prodnGR`	takes the product of $G(R)$ n-times
`cube2ds`	generates all two-tuples of a list of integers
`cube3ds`	generates all three-tuples of a list of integers
`cube4ds`	generates all four-tuples of a list of integers

5.1.2 Sage

Using the Sage programming environment, we wrote several procedures to aid in visualizing the tiles that we studied in this project. There are three separate groups of code that one can use to visualize D_n and T_n.

The first is for the one-dimensional case, where we are just looking at whether every $x \in \mathbb{Z}$ can be represented using our base A, and $|D| = |A|$ digits. Matula defines the digit set to be *basic* if every $x \in \mathbb{Z}$ is uniquely representable with A and D [18]. With the procedures written in Sage, we are able to generate $D_n := \{x \in \mathbb{Z} | k = \sum_{j=0}^{n-1)} A^j d_j,\ d_j \in D\}$, for any specified n. In the one dimensional case, it is also instructive to represent these D_n as graphs, where the nodes (or "vertices") are our digits and the digits of each level set D_n, and the edges between nodes exist if $A \cdot x_{i-1} + d_j = x_i$ for some $d_j \in D$ (in other words, if there is an edge from node x_{i-1} to x_i where $x_{i-1} \in D_{n-1}$ and $x_i \in D_n$). This is useful in that if the graph generates a tree, then we know we have unique representation. If the graph has cycles, then we know that some integers in D_n do not have a unique representation, and thus we do not have a basic, or radix, representation. The program generates a graph in 3D. The image is produced using graphics software JMol, which allows the user to click on the graph and move it around. This feature assists in determining if cycles are present.

In the two-dimensional case, again we have written a procedure to calculate the sets D_n. In two dimensions we have also written a procedure to calculate and visualize $T_n := \{\sum_{j=1}^{\infty} A^{-j} d_{-j} : d_{-j} \in D\}$. This allows us to visually determine if we have a unique radix representation for each $x \in D_n$ for n not too large. That is, we can compute and plot D_n up to where Sage is either computationally too

slow, or the plot is so large that we can no longer see if there are missing points. We have presented the two dimensional cases primarily as an interactive function, where there is a 'slider' allowing one to change the value of n in an applet-like environment. Being able to plot the T_n has been very useful in determining whether a tile is connected or disconnected.

In the three-dimensional case, we are able to use Sage in the same way as in the two-dimensional case. The difference here is in the visualization. Images are displayed using JMol again, and so we are able to rotate and better examine our 3D tiles and radix representations. Again, we used the interactive function format to aid in ease of visualization and use of the program.

5.2 Examples

In this section we present experimental results that we found using the above described computational tools.

5.2.1 Basic Examples

All of the two-dimensional examples that we investigated computationally used the following matrix form:

$$A := \begin{bmatrix} \lambda & k \\ 0 & \lambda \end{bmatrix}. \tag{5.1}$$

We will indicate which value of λ and k is being used in each example. Many of the examples come as a set of data. We will indicate what properties we were interested in for each of the data sets.

Recall the previous work done by Bandt and Wang for tiles in $\mathbb{R}^2$ [5]. They determined that if a tile T has six neighbours, then T is hexagon-like. If T has four edge neighbours and four vertex neighbours (or eight neighbours) then T is square-like. Our experimental data agrees with this property.

In all of the examples, we are interested in being able to calculate $S \cap \Delta D$, the intersection of the set of neighbours S with the difference set of the digit set D. We want to know if $S \cap \Delta D$ contains a basis for $\mathbb{Z}^m$, as this will indicate whether our tile T is connected or disconnected. Note that the $\mathbf{0}$ element is not removed in any of the examples, unlike Algorithm 3.19 indicates. If one mentally removes $\mathbf{0}$ from the list, we have the intersection of the set of proper neighbour, $S - \{0\}$, with ΔD. With $\mathbf{0}$ removed, the number of neighbours remaining agrees with previous work on the the shape of the tile.

Skew matrices were of particular interest (as the following examples exhibit), as it is the skewness of the matrix, and thus the digit set and the level sets D_n that will determine connectivity of the tile T. Rotation and dilation alone do not affect connectivity, as scaling and rotation do not change the connectivity of the level sets D_n, and thus the overall connectivity of the tile T.

The first series of twelve examples tested our initial hypothesis that if $S \cap \Delta D$ did not contain a basis of $\mathbb{Z}^m$ then our tile T would be disconnected. (Note that in our examples $m = 2$.)

Figure 5.1: $\lambda = 3$, $k = -1$.

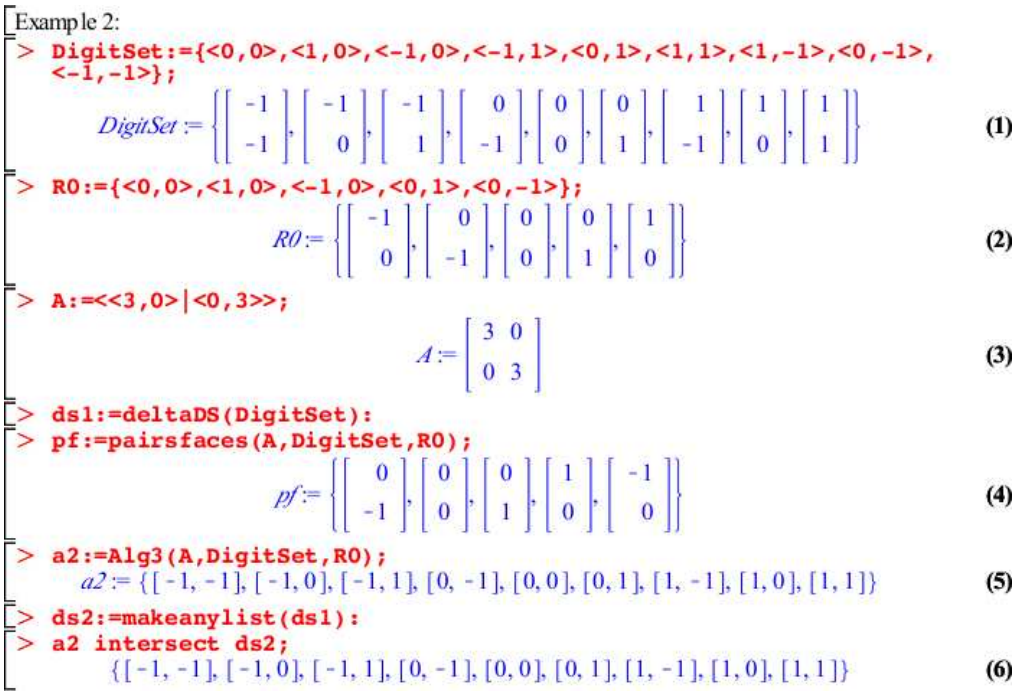


Figure 5.2: $\lambda = 3$, $k = 0$.

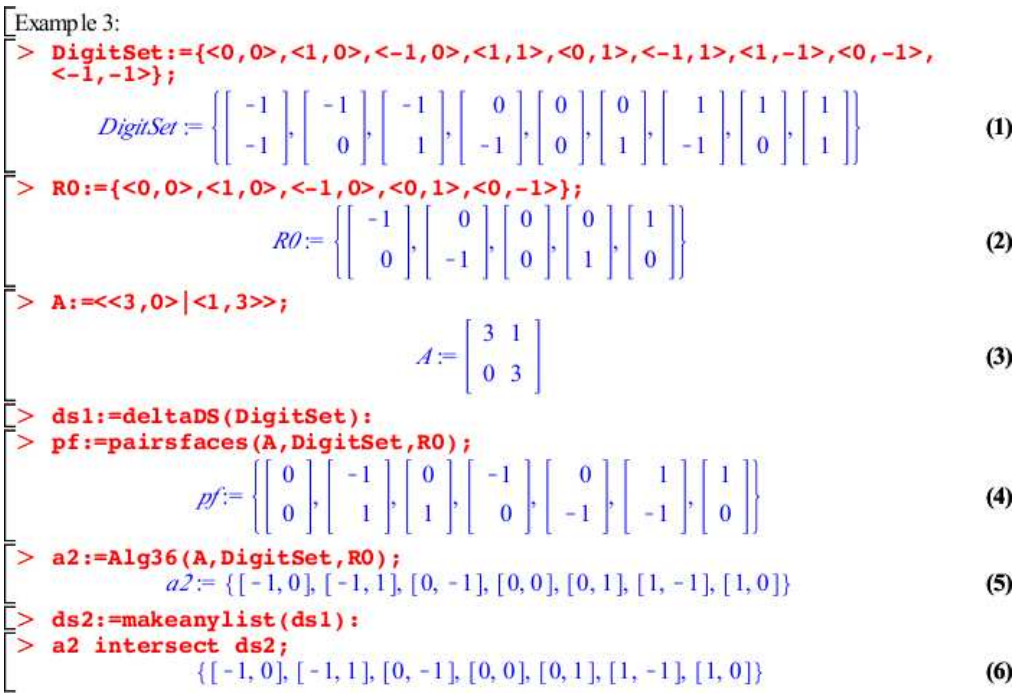


Figure 5.3: $\lambda = 3$, $k = 1$.

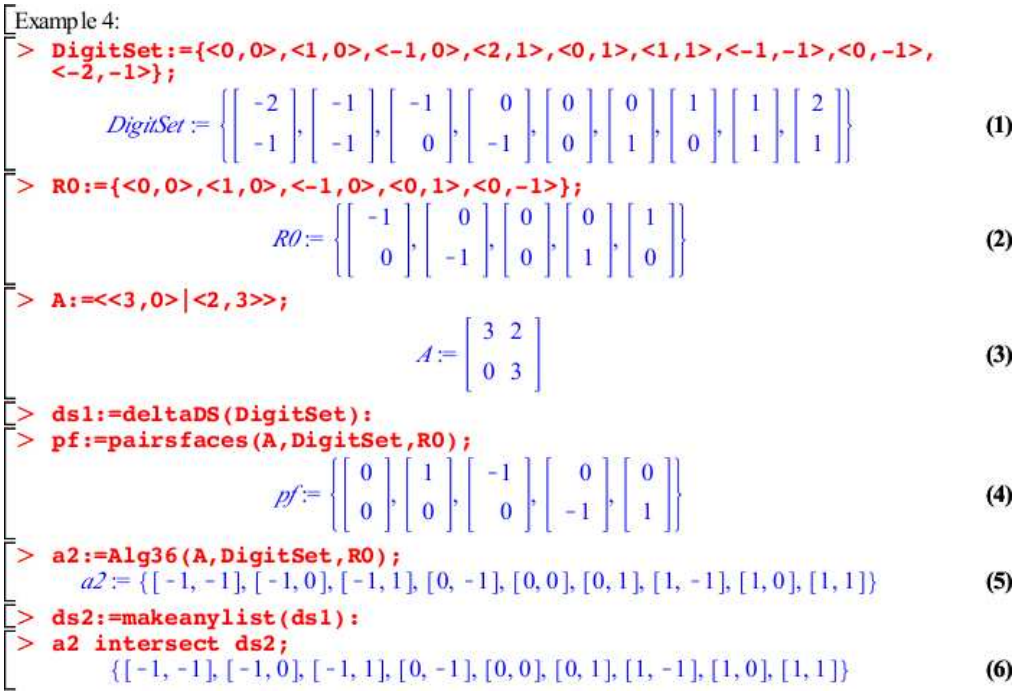


Figure 5.4: $\lambda = 3$, $k = 2$.

Example 5:

```
> DigitSet:={<0,0>,<1,0>,<-1,0>,<2,1>,<0,1>,<1,1>,<-1,-1>,<0,-1>,
  <-2,-1>};
```

$$DigitSet := \left\{ \begin{bmatrix} -2 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \end{bmatrix} \right\} \quad (1)$$

```
> R0:={<0,0>,<1,0>,<-1,0>,<0,1>,<0,-1>};
```

$$R0 := \left\{ \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right\} \quad (2)$$

```
> A:=<<3,0>|<3,3>>;
```

$$A := \begin{bmatrix} 3 & 3 \\ 0 & 3 \end{bmatrix} \quad (3)$$

```
> ds1:=deltaDS(DigitSet):
> pf:=pairsfaces(A,DigitSet,R0);
```

$$pf := \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\} \quad (4)$$

```
> a2:=Alg36(A,DigitSet,R0);
```

$$a2 := \{[-1, 0], [-1, 1], [0, -1], [0, 0], [0, 1], [1, -1], [1, 0]\} \quad (5)$$

```
> ds2:=makeanylist(ds1):
> a2 intersect ds2;
```

$$\{[-1, 0], [-1, 1], [0, -1], [0, 0], [0, 1], [1, -1], [1, 0]\} \quad (6)$$

Figure 5.5: $\lambda = 3$, $k = 3$.

Example 6:

```
> DigitSet:={<0,0>,<1,0>,<-1,0>,<2,1>,<0,1>,<1,1>,<-1,-1>,<0,-1>,
  <-2,-1>};
```

$$DigitSet := \left\{ \begin{bmatrix} -2 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \end{bmatrix} \right\} \quad (1)$$

```
> R0:={<0,0>,<1,0>,<-1,0>,<0,1>,<0,-1>};
```

$$R0 := \left\{ \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right\} \quad (2)$$

```
> A:=<<3,0>|<4,3>>;
```

$$A := \begin{bmatrix} 3 & 4 \\ 0 & 3 \end{bmatrix} \quad (3)$$

```
> ds1:=deltaDS(DigitSet):
> pf:=pairsfaces(A,DigitSet,R0);
```

$$pf := \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\} \quad (4)$$

```
> a2:=Alg36(A,DigitSet,R0);
```

$$a2 := \{[-2, 1], [-1, 0], [-1, 1], [0, -1], [0, 0], [0, 1], [1, -1], [1, 0], [2, -1]\} \quad (5)$$

```
> ds2:=makeanylist(ds1):
> a2 intersect ds2;
```

$$\{[-1, 0], [-1, 1], [0, -1], [0, 0], [0, 1], [1, -1], [1, 0]\} \quad (6)$$

Figure 5.6: $\lambda = 3$, $k = 4$.

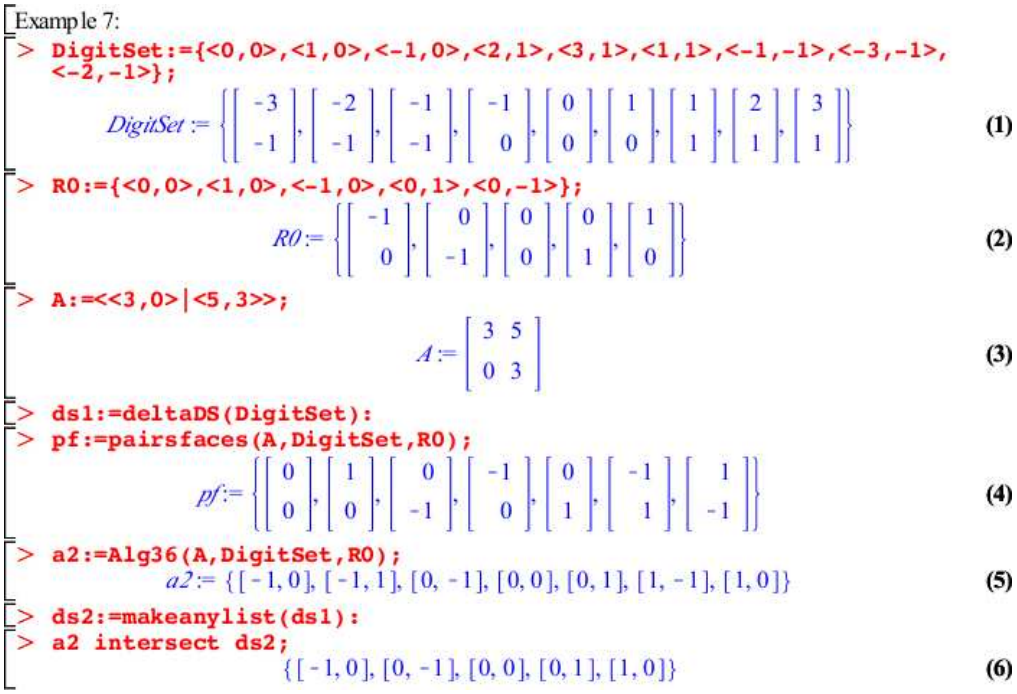


Figure 5.7: $\lambda = 3$, $k = 5$.

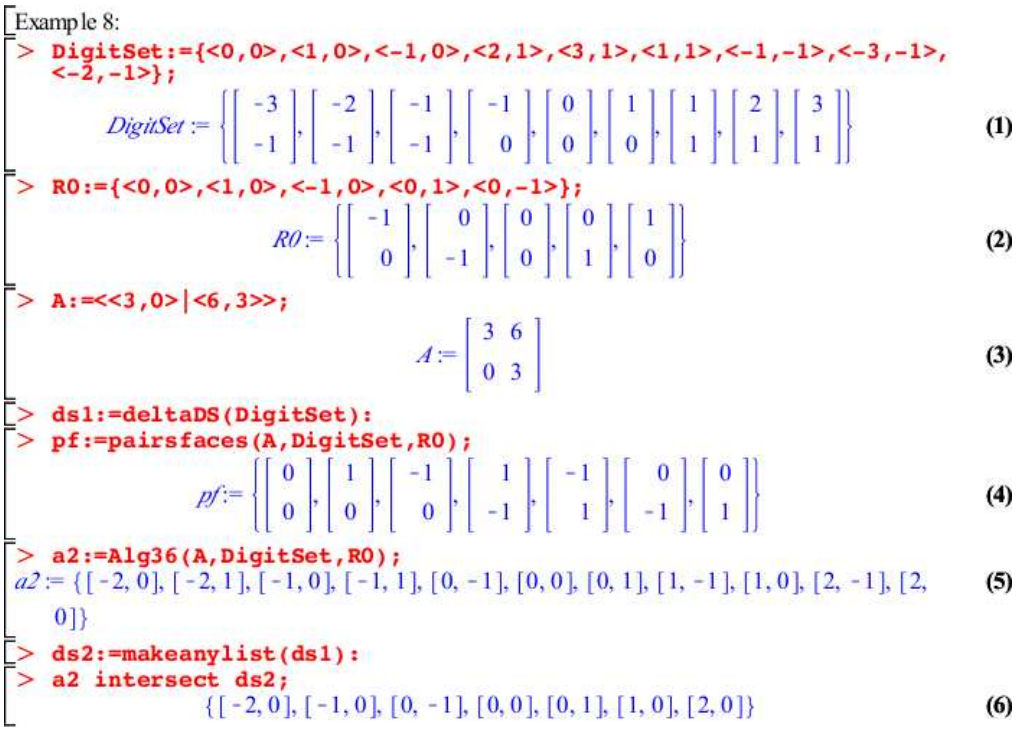


Figure 5.8: $\lambda = 3$, $k = 6$.

At $k = 6$, $S \cap \Delta D$ still contains a basis for $\mathbb{Z}^2$. When $k = 7$, however, $S \cap \Delta D$ no longer contains a basis. For $k = 6$, we have included a picture generated from the program written in Sage showing that the tile is connected. For $k = 7$, we have provided an image from Sage that shows that the tile has become disconnected.

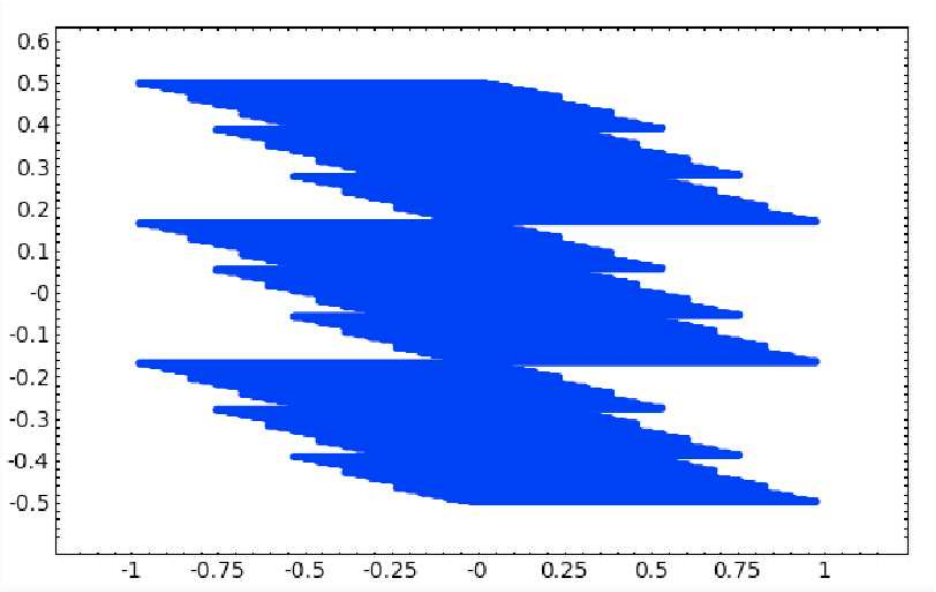


Figure 5.9: T_7 for $\lambda = 3$, $k = 6$.

Example 9:

```
> DigitSet:={<0,0>,<1,0>,<-1,0>,<2,1>,<3,1>,<1,1>,<-1,-1>,<-3,-1>,
<-2,-1>};
```

$$DigitSet := \left\{ \begin{bmatrix} -3 \\ -1 \end{bmatrix}, \begin{bmatrix} -2 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 3 \\ 1 \end{bmatrix} \right\} \quad (1)$$

```
> R0:={<0,0>,<1,0>,<-1,0>,<0,1>,<0,-1>};
```

$$R0 := \left\{ \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right\} \quad (2)$$

```
> A:=<<3,0>|<7,3>>;
```

$$A := \begin{bmatrix} 3 & 7 \\ 0 & 3 \end{bmatrix} \quad (3)$$

```
> ds1:=deltaDS(DigitSet):
> pf:=pairsfaces(A,DigitSet,R0);
```

$$pf := \left\{ \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right\} \quad (4)$$

```
> a2:=Alg36(A,DigitSet,R0);
```

$$a2 := \{[-2, 0], [-2, 1], [-1, 0], [-1, 1], [0, 0], [1, -1], [1, 0], [2, -1], [2, 0]\} \quad (5)$$

```
> ds2:=makeanylist(ds1):
> a2 intersect ds2;
```

$$\{[-2, 0], [-1, 0], [0, 0], [1, 0], [2, 0]\} \quad (6)$$

Figure 5.10: $\lambda = 3$, $k = 7$.

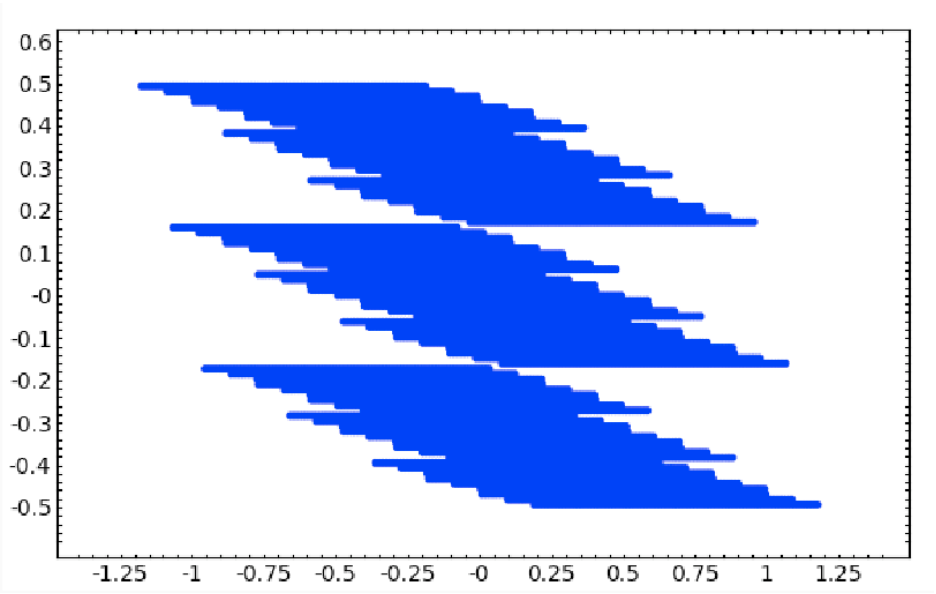


Figure 5.11: T_5 for $\lambda = 3$, $k = 7$.

Example 10:

```
> DigitSet:={<0,0>,<1,0>,<-1,0>,<2,1>,<3,1>,<4,1>,<-4,-1>,<-3,-1>,
<-2,-1>};
```

$$DigitSet := \left\{ \begin{bmatrix} -4 \\ -1 \end{bmatrix}, \begin{bmatrix} -3 \\ -1 \end{bmatrix}, \begin{bmatrix} -2 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 3 \\ 1 \end{bmatrix}, \begin{bmatrix} 4 \\ 1 \end{bmatrix} \right\} \quad (1)$$

```
> R0:={<0,0>,<1,0>,<-1,0>,<0,1>,<0,-1>};
```

$$R0 := \left\{ \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right\} \quad (2)$$

```
> A:=<<3,0>|<8,3>>;
```

$$A := \begin{bmatrix} 3 & 8 \\ 0 & 3 \end{bmatrix} \quad (3)$$

```
> ds1:=deltaDS(DigitSet):
> pf:=pairsfaces(A,DigitSet,R0);
```

$$pf := \left\{ \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \end{bmatrix} \right\} \quad (4)$$

```
> a2:=Alg36(A,DigitSet,R0);
```

$$a2 := \{[-2, 0], [-2, 1], [-1, 0], [-1, 1], [0, -1], [0, 0], [0, 1], [1, -1], [1, 0], [2, -1], [2, 0]\} \quad (5)$$

```
> ds2:=makeanylist(ds1):
> a2 intersect ds2;
```

$$\{[-2, 0], [-1, 0], [0, 0], [1, 0], [2, 0]\} \quad (6)$$

Figure 5.12: $\lambda = 3$, $k = 8$.

Example 11:

```
> DigitSet:={<0,0>,<1,0>,<-1,0>,<2,1>,<3,1>,<4,1>,<-4,-1>,<-3,-1>,
  <-2,-1>};
```

$$DigitSet := \left\{ \begin{bmatrix} -4 \\ -1 \end{bmatrix}, \begin{bmatrix} -3 \\ -1 \end{bmatrix}, \begin{bmatrix} -2 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 3 \\ 1 \end{bmatrix}, \begin{bmatrix} 4 \\ 1 \end{bmatrix} \right\} \quad (1)$$

```
> R0:={<0,0>,<1,0>,<-1,0>,<0,1>,<0,-1>};
```

$$R0 := \left\{ \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right\} \quad (2)$$

```
> A:=<<3,0>|<9,3>>;
```

$$A := \begin{bmatrix} 3 & 9 \\ 0 & 3 \end{bmatrix} \quad (3)$$

```
> ds1:=deltaDS(DigitSet):
> pf:=pairsfaces(A,DigitSet,R0);
```

$$pf := \left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right\} \quad (4)$$

```
> a2:=Alg36(A,DigitSet,R0);
```

$$a2 := \{[-2, 0], [-2, 1], [-1, 0], [-1, 1], [0, 0], [1, -1], [1, 0], [2, -1], [2, 0]\} \quad (5)$$

```
> ds2:=makeanylist(ds1):
> a2 intersect ds2;
```

$$\{[-2, 0], [-1, 0], [0, 0], [1, 0], [2, 0]\} \quad (6)$$

Figure 5.13: $\lambda = 3$, $k = 9$.

Example 12:

```
> DigitSet:={<0,0>,<1,0>,<-1,0>,<2,1>,<3,1>,<4,1>,<-4,-1>,<-3,-1>,
  <-2,-1>};
```

$$DigitSet := \left\{ \begin{bmatrix} -4 \\ -1 \end{bmatrix}, \begin{bmatrix} -3 \\ -1 \end{bmatrix}, \begin{bmatrix} -2 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 3 \\ 1 \end{bmatrix}, \begin{bmatrix} 4 \\ 1 \end{bmatrix} \right\} \quad (1)$$

```
> R0:={<0,0>,<1,0>,<-1,0>,<0,1>,<0,-1>}:
> A:=<<3,0>|<10,3>>;
```

$$A := \begin{bmatrix} 3 & 10 \\ 0 & 3 \end{bmatrix} \quad (2)$$

```
> ds1:=deltaDS(DigitSet):
> pf:=pairsfaces(A,DigitSet,R0);
```

$$pf := \left\{ \begin{bmatrix} 2 \\ -1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \begin{bmatrix} -2 \\ 1 \end{bmatrix} \right\} \quad (3)$$

```
> a2:=Alg36(A,DigitSet,R0);
```

$$a2 := \{[-3, 1], [-2, 0], [-2, 1], [-1, 0], [-1, 1], [0, 0], [1, -1], [1, 0], [2, -1], [2, 0], [3, -1]\} \quad (4)$$

```
> ds2:=makeanylist(ds1):
> a2 intersect ds2;
```

$$\{[-2, 0], [-1, 0], [0, 0], [1, 0], [2, 0]\} \quad (5)$$

Figure 5.14: $\lambda = 3$, $k = 10$.

After this first set of examples, we conjectured that when $k = \lambda^2 - \lambda$, we would still have a connected tile, but when $k = \lambda^2 - \lambda + 1$, the tile would no longer be connected. In the following six examples, we experimentally verified this conjecture for cases $\lambda = 4, 5$, and 6.

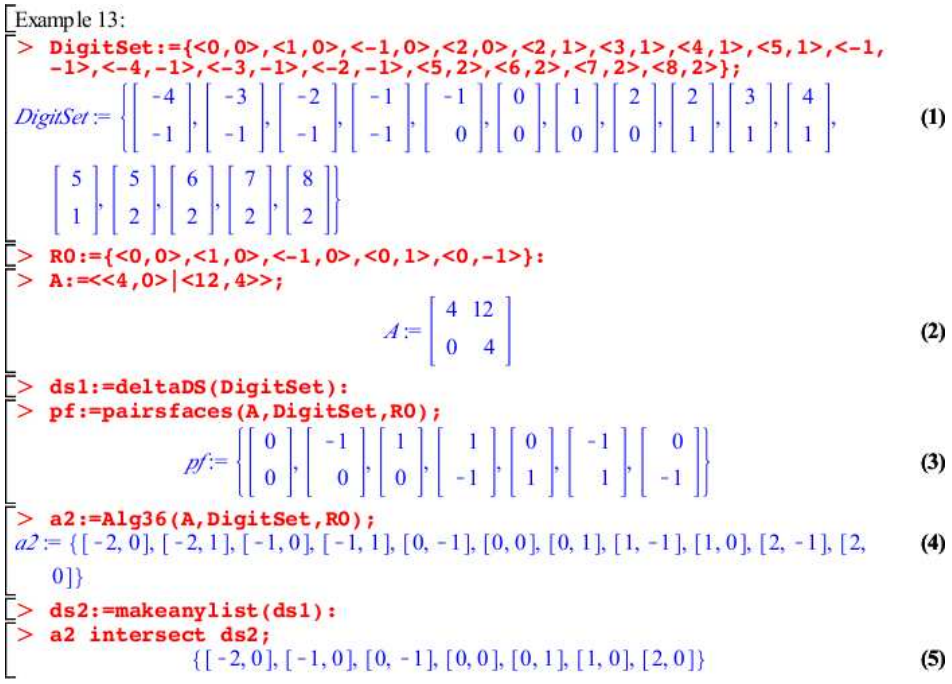


Figure 5.15: $\lambda = 4$, $k = 12$.

Example 14:

```
> DigitSet:={<0,0>,<1,0>,<-1,0>,<2,0>,<2,1>,<3,1>,<4,1>,<5,1>,<-1,
  -1>,<-4,-1>,<-3,-1>,<-2,-1>,<5,2>,<6,2>,<7,2>,<8,2>};
```

$$DigitSet := \left\{ \begin{bmatrix} -4 \\ -1 \end{bmatrix}, \begin{bmatrix} -3 \\ -1 \end{bmatrix}, \begin{bmatrix} -2 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 3 \\ 1 \end{bmatrix}, \begin{bmatrix} 4 \\ 1 \end{bmatrix}, \begin{bmatrix} 5 \\ 1 \end{bmatrix}, \begin{bmatrix} 5 \\ 2 \end{bmatrix}, \begin{bmatrix} 6 \\ 2 \end{bmatrix}, \begin{bmatrix} 7 \\ 2 \end{bmatrix}, \begin{bmatrix} 8 \\ 2 \end{bmatrix} \right\} \quad (1)$$

```
> R0:={<0,0>,<1,0>,<-1,0>,<0,1>,<0,-1>}:
> A:=<<4,0>|<13,4>>;
```

$$A := \begin{bmatrix} 4 & 13 \\ 0 & 4 \end{bmatrix} \quad (2)$$

```
> ds1:=deltaDS(DigitSet):
> pf:=pairsfaces(A,DigitSet,R0);
```

$$pf := \left\{ \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right\} \quad (3)$$

```
> a2:=Alg36(A,DigitSet,R0);
```

$$a2 := \{[-2, 0], [-2, 1], [-1, 0], [-1, 1], [0, 0], [1, -1], [1, 0], [2, -1], [2, 0]\} \quad (4)$$

```
> ds2:=makeanylist(ds1):
> a2 intersect ds2;
```

$$\{[-2, 0], [-1, 0], [0, 0], [1, 0], [2, 0]\} \quad (5)$$

Figure 5.16: $\lambda = 4$, $k = 13$.

Example 15:

```
> DigitSet:={<0,0>,<1,0>,<-1,0>,<2,0>,<-2,0>,<2,1>,<3,1>,<4,1>,<5,
  1>,<6,1>,<-6,-1>,<-5,-1>,<-4,-1>,<-3,-1>,<-2,-1>,<6,2>,<7,2>,<8,
  2>,<6,2>,<7,2>,<8,2>,<9,2>,<10,2>,<-6,-2>,<-7,-2>,<-8,-2>,<-9,
  -2>,<-10,-2>};
```

$$DigitSet := \left\{ \begin{bmatrix} -10 \\ -2 \end{bmatrix}, \begin{bmatrix} -9 \\ -2 \end{bmatrix}, \begin{bmatrix} -8 \\ -2 \end{bmatrix}, \begin{bmatrix} -7 \\ -2 \end{bmatrix}, \begin{bmatrix} -6 \\ -2 \end{bmatrix}, \begin{bmatrix} -6 \\ -1 \end{bmatrix}, \begin{bmatrix} -5 \\ -1 \end{bmatrix}, \begin{bmatrix} -4 \\ -1 \end{bmatrix}, \begin{bmatrix} -3 \\ -1 \end{bmatrix}, \begin{bmatrix} -2 \\ -1 \end{bmatrix}, \begin{bmatrix} -2 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 3 \\ 1 \end{bmatrix}, \begin{bmatrix} 4 \\ 1 \end{bmatrix}, \begin{bmatrix} 5 \\ 1 \end{bmatrix}, \begin{bmatrix} 6 \\ 1 \end{bmatrix}, \begin{bmatrix} 6 \\ 2 \end{bmatrix}, \begin{bmatrix} 7 \\ 2 \end{bmatrix}, \begin{bmatrix} 8 \\ 2 \end{bmatrix}, \begin{bmatrix} 9 \\ 2 \end{bmatrix}, \begin{bmatrix} 10 \\ 2 \end{bmatrix} \right\} \quad (1)$$

```
> R0:={<0,0>,<1,0>,<-1,0>,<0,1>,<0,-1>}:
> A:=<<5,0>|<20,5>>;
```

$$A := \begin{bmatrix} 5 & 20 \\ 0 & 5 \end{bmatrix} \quad (2)$$

```
> ds1:=deltaDS(DigitSet):
> pf:=pairsfaces(A,DigitSet,R0);
```

$$pf := \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \end{bmatrix} \right\} \quad (3)$$

```
> a2:=Alg36(A,DigitSet,R0);
```

$$a2 := \{[-2, 0], [-2, 1], [-1, 0], [-1, 1], [0, -1], [0, 0], [0, 1], [1, -1], [1, 0], [2, -1], [2, 0]\} \quad (4)$$

```
> ds2:=makeanylist(ds1):
> a2 intersect ds2;
```

$$\{[-2, 0], [-1, 0], [0, -1], [0, 0], [0, 1], [1, 0], [2, 0]\} \quad (5)$$

Figure 5.17: $\lambda = 5$, $k = 20$.

Example 16:

```
> DigitSet:={<0,0>,<1,0>,<-1,0>,<2,0>,<-2,0>,<2,1>,<3,1>,<4,1>,<5,
  1>,<6,1>,<-6,-1>,<-5,-1>,<-4,-1>,<-3,-1>,<-2,-1>,<6,2>,<7,2>,<8,
  2>,<6,2>,<7,2>,<8,2>,<9,2>,<10,2>,<-6,-2>,<-7,-2>,<-8,-2>,<-9,
  -2>,<-10,-2>};
```

$$DigitSet := \left\{ \begin{bmatrix} -10 \\ -2 \end{bmatrix}, \begin{bmatrix} -9 \\ -2 \end{bmatrix}, \begin{bmatrix} -8 \\ -2 \end{bmatrix}, \begin{bmatrix} -7 \\ -2 \end{bmatrix}, \begin{bmatrix} -6 \\ -2 \end{bmatrix}, \begin{bmatrix} -6 \\ -1 \end{bmatrix}, \begin{bmatrix} -5 \\ -1 \end{bmatrix}, \begin{bmatrix} -4 \\ -1 \end{bmatrix}, \begin{bmatrix} -3 \\ -1 \end{bmatrix}, \begin{bmatrix} -2 \\ -1 \end{bmatrix}, \right.$$
$$\left. \begin{bmatrix} -2 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 3 \\ 1 \end{bmatrix}, \begin{bmatrix} 4 \\ 1 \end{bmatrix}, \begin{bmatrix} 5 \\ 1 \end{bmatrix}, \begin{bmatrix} 6 \\ 1 \end{bmatrix}, \begin{bmatrix} 6 \\ 2 \end{bmatrix}, \begin{bmatrix} 7 \\ 2 \end{bmatrix}, \begin{bmatrix} 8 \\ 2 \end{bmatrix}, \right.$$
$$\left. \begin{bmatrix} 9 \\ 2 \end{bmatrix}, \begin{bmatrix} 10 \\ 2 \end{bmatrix} \right\} \quad (1)$$

```
> R0:={<0,0>,<1,0>,<-1,0>,<0,1>,<0,-1>}:
> A:=<<5,0>|<21,5>>;
```

$$A := \begin{bmatrix} 5 & 21 \\ 0 & 5 \end{bmatrix} \quad (2)$$

```
> ds1:=deltaDS(DigitSet):
> pf:=pairsfaces(A,DigitSet,R0);
```

$$pf := \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \end{bmatrix} \right\} \quad (3)$$

```
> a2:=Alg36(A,DigitSet,R0);
```

$$a2 := \{[-2, 0], [-2, 1], [-1, 0], [-1, 1], [0, 0], [1, -1], [1, 0], [2, -1], [2, 0]\} \quad (4)$$

```
> ds2:=makeanylist(ds1):
> a2 intersect ds2;
```

$$\{[-2, 0], [-1, 0], [0, 0], [1, 0], [2, 0]\} \quad (5)$$

Figure 5.18: $\lambda = 5$, $k = 21$.

Example 17:

```
> DigitSet:={<0,0>,<1,0>,<-1,0>,<2,0>,<-2,0>,<3,0>,<7,1>,<3,1>,<4,
  1>,<5,1>,<6,1>,<8,1>,<-6,-1>,<-5,-1>,<-4,-1>,<-3,-1>,<-2,-1>,
  <-7,-1>,<9,2>,<10,2>,<8,2>,<11,2>,<12,2>,<13,2>,<13,3>,<14,3>,
  <15,3>,<16,3>,<17,3>,<18,3>,<-11,-2>,<-7,-2>,<-8,-2>,<-9,-2>,
  <-10,-2>,<-12,-2>};
```

$$DigitSet := \left\{ \begin{bmatrix} -12 \\ -2 \end{bmatrix}, \begin{bmatrix} -11 \\ -2 \end{bmatrix}, \begin{bmatrix} -10 \\ -2 \end{bmatrix}, \begin{bmatrix} -9 \\ -2 \end{bmatrix}, \begin{bmatrix} -8 \\ -2 \end{bmatrix}, \begin{bmatrix} -7 \\ -2 \end{bmatrix}, \begin{bmatrix} -7 \\ -1 \end{bmatrix}, \begin{bmatrix} -6 \\ -1 \end{bmatrix}, \begin{bmatrix} -5 \\ -1 \end{bmatrix}, \right. \tag{1}$$
$$\begin{bmatrix} -4 \\ -1 \end{bmatrix}, \begin{bmatrix} -3 \\ -1 \end{bmatrix}, \begin{bmatrix} -2 \\ -1 \end{bmatrix}, \begin{bmatrix} -2 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \begin{bmatrix} 3 \\ 0 \end{bmatrix}, \begin{bmatrix} 3 \\ 1 \end{bmatrix}, \begin{bmatrix} 4 \\ 1 \end{bmatrix}, \begin{bmatrix} 5 \\ 1 \end{bmatrix}, \begin{bmatrix} 6 \\ 1 \end{bmatrix},$$
$$\begin{bmatrix} 7 \\ 1 \end{bmatrix}, \begin{bmatrix} 8 \\ 1 \end{bmatrix}, \begin{bmatrix} 8 \\ 2 \end{bmatrix}, \begin{bmatrix} 9 \\ 2 \end{bmatrix}, \begin{bmatrix} 10 \\ 2 \end{bmatrix}, \begin{bmatrix} 11 \\ 2 \end{bmatrix}, \begin{bmatrix} 12 \\ 2 \end{bmatrix}, \begin{bmatrix} 13 \\ 2 \end{bmatrix}, \begin{bmatrix} 13 \\ 3 \end{bmatrix}, \begin{bmatrix} 14 \\ 3 \end{bmatrix}, \begin{bmatrix} 15 \\ 3 \end{bmatrix}, \begin{bmatrix} 16 \\ 3 \end{bmatrix},$$
$$\left. \begin{bmatrix} 17 \\ 3 \end{bmatrix}, \begin{bmatrix} 18 \\ 3 \end{bmatrix} \right\}$$

```
> R0:={<0,0>,<1,0>,<-1,0>,<0,1>,<0,-1>}:
> A:=<<6,0>|<30,6>>;
```

$$A := \begin{bmatrix} 6 & 30 \\ 0 & 6 \end{bmatrix} \tag{2}$$

```
> ds1:=deltaDS(DigitSet):
> pf:=pairsfaces(A,DigitSet,R0);
```

$$pf := \left\{ \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \end{bmatrix} \right\} \tag{3}$$

```
> a2:=Alg36(A,DigitSet,R0);
```

$$a2 := \{[-2, 0], [-2, 1], [-1, 0], [-1, 1], [0, -1], [0, 0], [0, 1], [1, -1], [1, 0], [2, -1], [2, 0]\} \tag{4}$$

```
> ds2:=makeanylist(ds1):
> a2 intersect ds2;
```

$$\{[-2, 0], [-1, 0], [0, -1], [0, 0], [0, 1], [1, 0], [2, 0]\} \tag{5}$$

Figure 5.19: $\lambda = 6$, $k = 30$.

Example 18:

```
> DigitSet:={<0,0>,<1,0>,<-1,0>,<2,0>,<-2,0>,<3,0>,<7,1>,<3,1>,<4,
  1>,<5,1>,<6,1>,<8,1>,<-6,-1>,<-5,-1>,<-4,-1>,<-3,-1>,<-2,-1>,
  <-7,-1>,<9,2>,<10,2>,<8,2>,<11,2>,<12,2>,<13,2>,<13,3>,<14,3>,
  <15,3>,<16,3>,<17,3>,<18,3>,<-11,-2>,<-7,-2>,<-8,-2>,<-9,-2>,
  <-10,-2>,<-12,-2>};
```

$$DigitSet := \left\{ \begin{bmatrix} -12 \\ -2 \end{bmatrix}, \begin{bmatrix} -11 \\ -2 \end{bmatrix}, \begin{bmatrix} -10 \\ -2 \end{bmatrix}, \begin{bmatrix} -9 \\ -2 \end{bmatrix}, \begin{bmatrix} -8 \\ -2 \end{bmatrix}, \begin{bmatrix} -7 \\ -2 \end{bmatrix}, \begin{bmatrix} -7 \\ -1 \end{bmatrix}, \begin{bmatrix} -6 \\ -1 \end{bmatrix}, \begin{bmatrix} -5 \\ -1 \end{bmatrix}, \right. \tag{1}$$
$$\begin{bmatrix} -4 \\ -1 \end{bmatrix}, \begin{bmatrix} -3 \\ -1 \end{bmatrix}, \begin{bmatrix} -2 \\ -1 \end{bmatrix}, \begin{bmatrix} -2 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \begin{bmatrix} 3 \\ 0 \end{bmatrix}, \begin{bmatrix} 3 \\ 1 \end{bmatrix}, \begin{bmatrix} 4 \\ 1 \end{bmatrix}, \begin{bmatrix} 5 \\ 1 \end{bmatrix}, \begin{bmatrix} 6 \\ 1 \end{bmatrix},$$
$$\begin{bmatrix} 7 \\ 1 \end{bmatrix}, \begin{bmatrix} 8 \\ 1 \end{bmatrix}, \begin{bmatrix} 8 \\ 2 \end{bmatrix}, \begin{bmatrix} 9 \\ 2 \end{bmatrix}, \begin{bmatrix} 10 \\ 2 \end{bmatrix}, \begin{bmatrix} 11 \\ 2 \end{bmatrix}, \begin{bmatrix} 12 \\ 2 \end{bmatrix}, \begin{bmatrix} 13 \\ 2 \end{bmatrix}, \begin{bmatrix} 13 \\ 3 \end{bmatrix}, \begin{bmatrix} 14 \\ 3 \end{bmatrix}, \begin{bmatrix} 15 \\ 3 \end{bmatrix}, \begin{bmatrix} 16 \\ 3 \end{bmatrix},$$
$$\left. \begin{bmatrix} 17 \\ 3 \end{bmatrix}, \begin{bmatrix} 18 \\ 3 \end{bmatrix} \right\}$$

```
> R0:={<0,0>,<1,0>,<-1,0>,<0,1>,<0,-1>}:
> A:=<<6,0>|<31,6>>;
```

$$A := \begin{bmatrix} 6 & 31 \\ 0 & 6 \end{bmatrix} \tag{2}$$

```
> ds1:=deltaDS(DigitSet):
> pf:=pairsfaces(A,DigitSet,R0);
```

$$pf := \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right\} \tag{3}$$

```
> a2:=Alg36(A,DigitSet,R0);
```

$$a2 := \{[-2, 0], [-2, 1], [-1, 0], [-1, 1], [0, 0], [1, -1], [1, 0], [2, -1], [2, 0]\} \tag{4}$$

```
> ds2:=makeanylist(ds1):
> a2 intersect ds2;
```

$$\{[-2, 0], [-1, 0], [0, 0], [1, 0], [2, 0]\} \tag{5}$$

Figure 5.20: $\lambda = 6$, $k = 31$.

5.2.2 Alternative Digit Sets

At this point we thought to try different digit sets, D, to see if we could find some connected tile no matter what dilation matrix A was. We are including the Maple output for the matrices A from the preceeding set of examples, and also for the value of k that now yields a disconnected tile. We conjectured that if we could 'correct' the skewness of the tile at this point then we would be on the right track. That is, we wanted D to counter the skew of A, or find a digit set that worked independently of A. This turned out not to be the case. We tried the alternative digit sets shown in the next four examples.

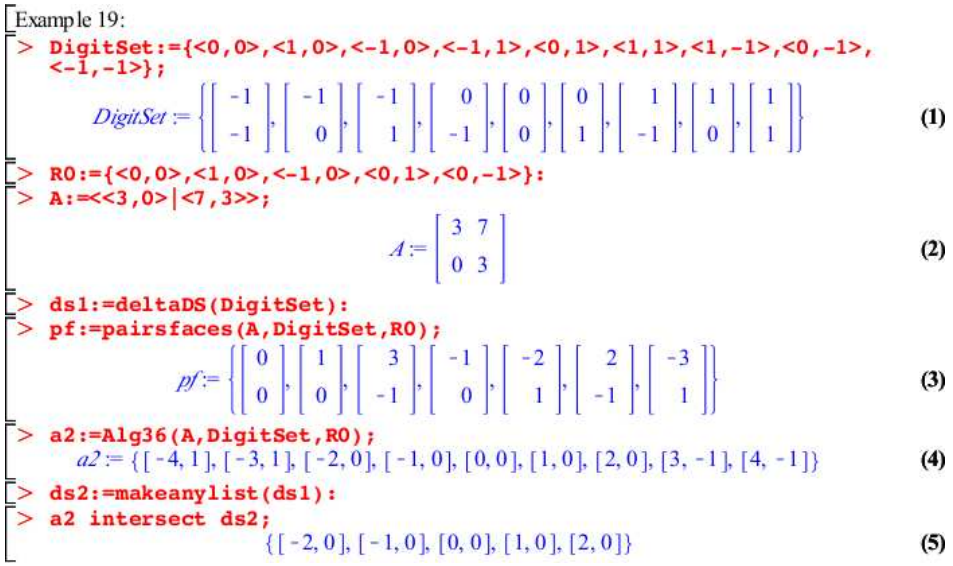


Figure 5.21: The digit set D for when $\lambda = 3$, $k = 0$ for every value of k. $\lambda = 3$, $k = 7$.

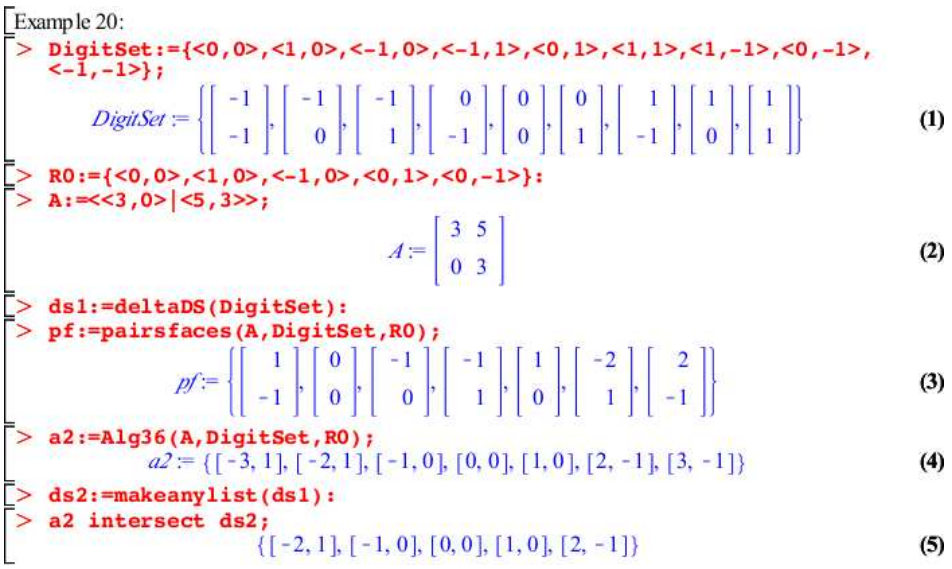


Figure 5.22: The digit set D for when $\lambda = 3$, $k = 0$ for every value of k. $\lambda = 3$, $k = 5$.

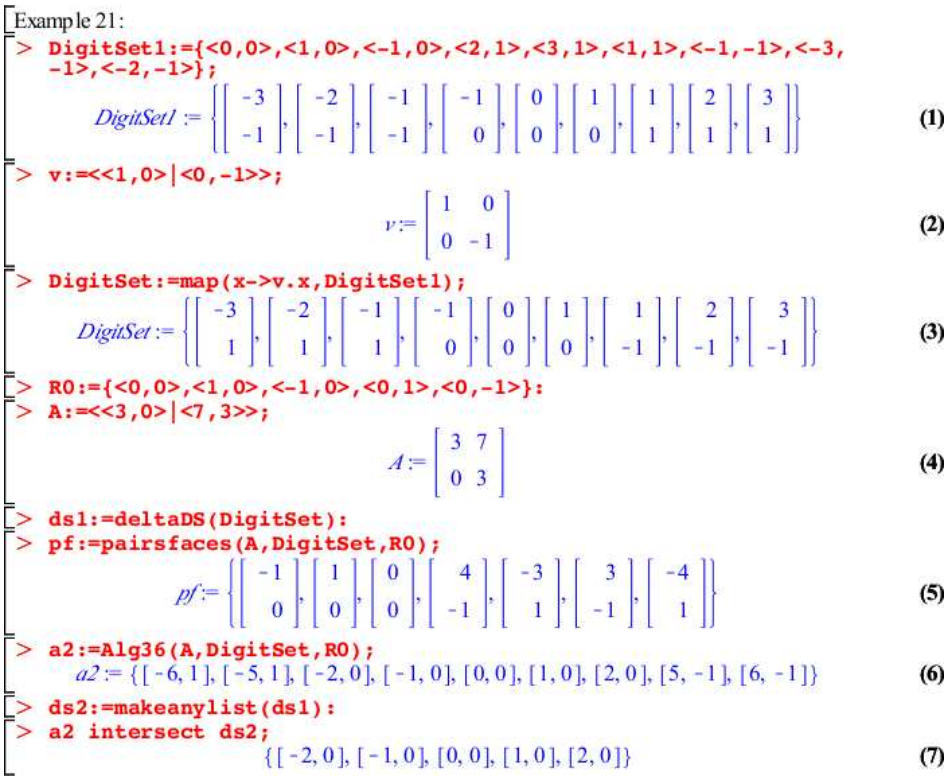


Figure 5.23: The digit set D, where $D = A^{-1}(F) \cap \mathbb{Z}^m$. $\lambda = 3$, $k = 7$.

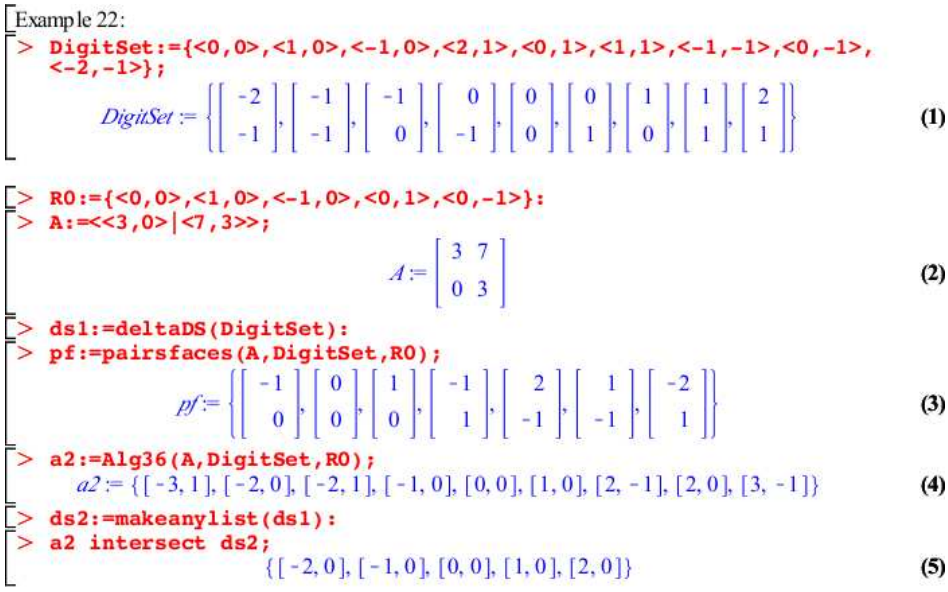


Figure 5.24: The digit set D, where D is the set of digits for the lattice form 1-less skew than what it should be, for an attempt at a more subtle correction of the skew.

$$\lambda = 3,\ k = 7.$$

5.2.3 Digit Sets from the Jordan Form

Recall that the Jordan Form of our matrix A is either

$$J := \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} \tag{5.2}$$

or

$$J := \begin{bmatrix} \lambda & 1 \\ 0 & \lambda \end{bmatrix}. \tag{5.3}$$

Here the matrix J is also a dilation matrix, but will always have the 'block' digit set D as previously described. We used the digit set $D = PD_J$ as the digit set for the dilation matrix A. In Example 5.25, we see that $S \cap \Delta D$ does not contain a basis of $\mathbb{Z}^2$, but contains a basis for a sub-lattice of $\mathbb{Z}^2$.

5.2.4 Strictly Skew Matrices and Higher Dimensional Examples

In exploring the connectedness of skew matrices, we looked at other values of λ in the Jordan form to see if $S \cap \Delta D$ still contained a basis. In Example 5.26, $\lambda = 4$, and in Example 5.27, $\lambda = 5$. Finally, to further demonstrate the capabilities of the Maple program, we are including two three-dimensional examples of the Jordan form skew matrices. We are also including a 3D graphic generated in Sage for these examples. Although running time is much longer, we wanted to indicate

Example 23:

```
> DigitSet1:=cube2ds([-1,0,1]);
```

$$DigitSet1 := \left\{\begin{bmatrix}-1\\-1\end{bmatrix}, \begin{bmatrix}-1\\0\end{bmatrix}, \begin{bmatrix}0\\-1\end{bmatrix}, \begin{bmatrix}-1\\1\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix}, \begin{bmatrix}1\\-1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}\right\} \quad (1)$$

```
> R0:={<0,0>,<1,0>,<-1,0>,<0,1>,<0,-1>};
```

$$R0 := \left\{\begin{bmatrix}-1\\0\end{bmatrix}, \begin{bmatrix}0\\-1\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}\right\} \quad (2)$$

```
> A:=<<3,0>|<7,3>>;
```

$$A := \begin{bmatrix}3 & 7\\0 & 3\end{bmatrix} \quad (3)$$

```
> Q := JordanForm(A, output = 'Q');
```

$$Q := \begin{bmatrix}7 & 0\\0 & 1\end{bmatrix} \quad (4)$$

```
> DigitSet:=map(x->Q.x,DigitSet1);
```

$$DigitSet := \left\{\begin{bmatrix}-7\\1\end{bmatrix}, \begin{bmatrix}7\\-1\end{bmatrix}, \begin{bmatrix}7\\0\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}-7\\-1\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}-7\\0\end{bmatrix}, \begin{bmatrix}7\\1\end{bmatrix}, \begin{bmatrix}0\\-1\end{bmatrix}\right\} \quad (5)$$

```
> ds1:=deltaDS(DigitSet);
```

$$ds1 := \left\{\begin{bmatrix}7\\-1\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}-14\\-1\end{bmatrix}, \begin{bmatrix}7\\1\end{bmatrix}, \begin{bmatrix}-7\\-1\end{bmatrix}, \begin{bmatrix}0\\-2\end{bmatrix}, \begin{bmatrix}-14\\0\end{bmatrix}, \begin{bmatrix}-7\\0\end{bmatrix}, \begin{bmatrix}-7\\-1\end{bmatrix}, \begin{bmatrix}7\\0\end{bmatrix}, \begin{bmatrix}-7\\0\end{bmatrix}, \right. \quad (6)$$

$$\begin{bmatrix}-7\\1\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\-1\end{bmatrix}, \begin{bmatrix}-7\\2\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}-7\\2\end{bmatrix}, \begin{bmatrix}-7\\0\end{bmatrix}, \begin{bmatrix}0\\2\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}0\\-1\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}7\\0\end{bmatrix},$$

$$\begin{bmatrix}-14\\1\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}7\\1\end{bmatrix}, \begin{bmatrix}-7\\0\end{bmatrix}, \begin{bmatrix}0\\-1\end{bmatrix}, \begin{bmatrix}0\\-2\end{bmatrix}, \begin{bmatrix}7\\-1\end{bmatrix}, \begin{bmatrix}14\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}-7\\-1\end{bmatrix}, \begin{bmatrix}7\\-2\end{bmatrix},$$

$$\begin{bmatrix}14\\-1\end{bmatrix}, \begin{bmatrix}-7\\0\end{bmatrix}, \begin{bmatrix}-7\\1\end{bmatrix}, \begin{bmatrix}0\\-2\end{bmatrix}, \begin{bmatrix}14\\2\end{bmatrix}, \begin{bmatrix}7\\0\end{bmatrix}, \begin{bmatrix}14\\-1\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}7\\-1\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}-14\\0\end{bmatrix},$$

$$\begin{bmatrix}-14\\0\end{bmatrix}, \begin{bmatrix}7\\0\end{bmatrix}, \begin{bmatrix}-7\\0\end{bmatrix}, \begin{bmatrix}-14\\2\end{bmatrix}, \begin{bmatrix}-14\\-1\end{bmatrix}, \begin{bmatrix}7\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}-7\\1\end{bmatrix}, \begin{bmatrix}-7\\-1\end{bmatrix}, \begin{bmatrix}0\\-1\end{bmatrix}, \begin{bmatrix}14\\1\end{bmatrix},$$

$$\begin{bmatrix}-7\\-2\end{bmatrix}, \begin{bmatrix}-14\\1\end{bmatrix}, \begin{bmatrix}7\\2\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}7\\-2\end{bmatrix}, \begin{bmatrix}0\\2\end{bmatrix}, \begin{bmatrix}0\\-1\end{bmatrix}, \begin{bmatrix}14\\-2\end{bmatrix}, \begin{bmatrix}7\\1\end{bmatrix}, \begin{bmatrix}0\\2\end{bmatrix}, \begin{bmatrix}7\\-1\end{bmatrix}, \begin{bmatrix}7\\1\end{bmatrix},$$

$$\left.\begin{bmatrix}14\\0\end{bmatrix}, \begin{bmatrix}7\\2\end{bmatrix}, \begin{bmatrix}-7\\-2\end{bmatrix}, \begin{bmatrix}-7\\1\end{bmatrix}, \begin{bmatrix}14\\0\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}-14\\-2\end{bmatrix}, \begin{bmatrix}7\\0\end{bmatrix}, \begin{bmatrix}0\\-1\end{bmatrix}, \begin{bmatrix}14\\1\end{bmatrix}\right\}$$

```
> pf:=pairsfaces(A,DigitSet,R0):
> a2:=Alg36(A,DigitSet,R0);
```

$$a2 := \{[-10, 1], [-9, 1], [-8, 0], [-8, 1], [-7, 0], [-7, 1], [-6, 0], [-6, 1], [-5, 0], [-5, 1], [-4, 0], [-4, 1], [-3, -1], [-3, 0], [-3, 1], [-2, -1], [-2, 0], [-2, 1], [-1, -1], [-1, 0], [-1, 1], [0, -1], [0, 0], [0, 1], [1, -1], [1, 0], [1, 1], [2, -1], [2, 0], [2, 1], [3, -1], [3, 0], [3, 1], [4, -1], [4, 0], [5, -1], [5, 0], [6, -1], [6, 0], [7, -1], [7, 0], [8, -1], [8, 0], [9, -1], [10, -1]\} \quad (7)$$

```
> ds2:=makeanylist(ds1):
> a2 intersect ds2;
```

$$\{[-7, 0], [-7, 1], [0, -1], [0, 0], [0, 1], [7, -1], [7, 0]\} \quad (8)$$

Figure 5.25: $\lambda = 3$, $k = 7$.

that work in higher dimensions is possible. These matrices are of the form

$$\begin{bmatrix} \lambda & k & 0 \\ 0 & \lambda & k \\ 0 & 0 & \lambda \end{bmatrix}.$$

Example 24:

```
> DigitSet:=cube2ds([1,-1,0,2]);
```

$$DigitSet := \left\{ \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 2 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \begin{bmatrix} 2 \\ -1 \end{bmatrix}, \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 2 \end{bmatrix} \right\} \quad (1)$$

```
> R0:={<0,0>,<1,0>,<-1,0>,<0,1>,<0,-1>};
```

$$R0 := \left\{ \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right\} \quad (2)$$

```
> A:=<<4,0>|<1,4>>;
```

$$A := \begin{bmatrix} 4 & 1 \\ 0 & 4 \end{bmatrix} \quad (3)$$

```
> ds1:=deltaDS(DigitSet):
> pf:=pairsfaces(A,DigitSet,R0):
> a2:=Alg36(A,DigitSet,R0);
```

$$a2 := \{[-1, 0], [-1, 1], [0, -1], [0, 0], [0, 1], [1, -1], [1, 0]\} \quad (4)$$

```
> ds2:=makeanylist(ds1):
> a2 intersect ds2;
```

$$\{[-1, 0], [-1, 1], [0, -1], [0, 0], [0, 1], [1, -1], [1, 0]\} \quad (5)$$

Figure 5.26: $\lambda = 4$, $k = 1$.

Example 25:

```
> DigitSet:=cube2ds([-2,1,-1,0,2]);
```

$$DigitSet := \left\{ \begin{bmatrix} -2 \\ -2 \end{bmatrix}, \begin{bmatrix} -2 \\ -1 \end{bmatrix}, \begin{bmatrix} -2 \\ 0 \end{bmatrix}, \begin{bmatrix} -2 \\ 1 \end{bmatrix}, \begin{bmatrix} -2 \\ 2 \end{bmatrix}, \begin{bmatrix} -1 \\ -2 \end{bmatrix}, \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 2 \end{bmatrix}, \begin{bmatrix} 0 \\ -2 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 \\ -2 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \begin{bmatrix} 2 \\ -2 \end{bmatrix}, \begin{bmatrix} 2 \\ -1 \end{bmatrix}, \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 2 \end{bmatrix} \right\} \quad (1)$$

```
> R0:={<0,0>,<1,0>,<-1,0>,<0,1>,<0,-1>};
```

$$R0 := \left\{ \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right\} \quad (2)$$

```
> A:=<<5,0>|<1,5>>;
```

$$A := \begin{bmatrix} 5 & 1 \\ 0 & 5 \end{bmatrix} \quad (3)$$

```
> ds1:=deltaDS(DigitSet):
> pf:=pairsfaces(A,DigitSet,R0):
> a2:=Alg36(A,DigitSet,R0);
```

$$a2 := \{[-1, 0], [-1, 1], [0, -1], [0, 0], [0, 1], [1, -1], [1, 0]\} \quad (4)$$

```
> ds2:=makeanylist(ds1):
> a2 intersect ds2;
```

$$\{[-1, 0], [-1, 1], [0, -1], [0, 0], [0, 1], [1, -1], [1, 0]\} \quad (5)$$

Figure 5.27: $\lambda = 5$, $k = 1$.

Example 26:

```
> DigitSet:=cube3ds([-1,0,1]);
```

$$DigitSet := \left\{ \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ -1 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \\ -1 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right\} \quad (1)$$

```
> R0:={<0,0,0>,<1,0,0>,<-1,0,0>,<0,1,0>,<0,-1,0>,<0,0,1>,<0,0,-1>}
  :
> A:=<<3,0,0>|<1,3,0>|<0,1,3>>;
```

$$A := \begin{bmatrix} 3 & 1 & 0 \\ 0 & 3 & 1 \\ 0 & 0 & 3 \end{bmatrix} \quad (2)$$

```
> ds1:=deltaDS(DigitSet):
> pf:=pairsfaces(A,DigitSet,R0):
> a2:=Alg36(A,DigitSet,R0);
```

$$a2 := \{[-1, 0, -1], [-1, 0, 0], [-1, 0, 1], [-1, 1, -1], [-1, 1, 0], [0, -1, 0], [0, -1, 1], [0, 0, -1], [0, 0, 0], [0, 0, 1], [0, 1, -1], [0, 1, 0], [1, -1, 0], [1, -1, 1], [1, 0, -1], [1, 0, 0], [1, 0, 1]\} \quad (3)$$

```
> ds2:=makeanylist(ds1):
> a2 intersect ds2;
```

$$\{[-1, 0, -1], [-1, 0, 0], [-1, 0, 1], [-1, 1, -1], [-1, 1, 0], [0, -1, 0], [0, -1, 1], [0, 0, -1], [0, 0, 0], [0, 0, 1], [0, 1, -1], [0, 1, 0], [1, -1, 0], [1, -1, 1], [1, 0, -1], [1, 0, 0], [1, 0, 1]\} \quad (4)$$

Figure 5.28: $\lambda = 3$, $k = 1$.

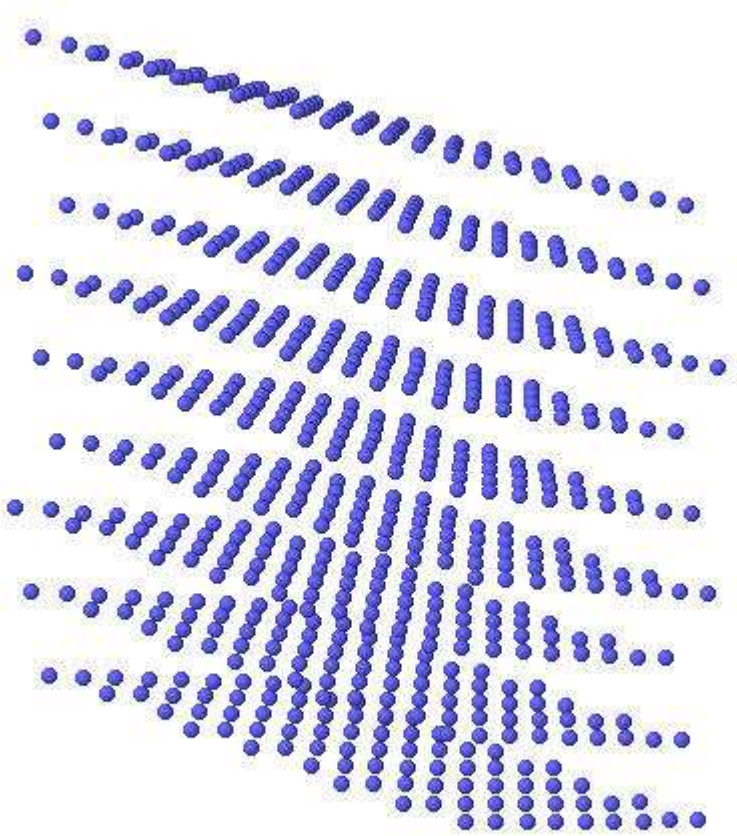


Figure 5.29: T_4 for $\lambda = 3$, $k = 1$.

Example 27:

```
> DigitSet:=cube3ds([-1,0,1,2]);
```

$$DigitSet := \left\{ \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ -1 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ -1 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ -1 \\ 2 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \\ 2 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix}, \right. \quad (1)$$

$$\begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \\ 2 \end{bmatrix}, \begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ 2 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 2 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 2 \\ 2 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \\ -1 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \\ 2 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix},$$

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix}, \begin{bmatrix} 0 \\ 2 \\ -1 \end{bmatrix}, \begin{bmatrix} 0 \\ 2 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 2 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix},$$

$$\begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix},$$

$$\begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix}, \begin{bmatrix} 2 \\ -1 \\ -1 \end{bmatrix}, \begin{bmatrix} 2 \\ -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ -1 \\ 2 \end{bmatrix}, \begin{bmatrix} 2 \\ 0 \\ -1 \end{bmatrix}, \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 0 \\ 2 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \\ -1 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix},$$

$$\left. \begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix}, \begin{bmatrix} 2 \\ 2 \\ -1 \end{bmatrix}, \begin{bmatrix} 2 \\ 2 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 2 \\ 2 \end{bmatrix} \right\}$$

```
> R0:={<0,0,0>,<1,0,0>,<-1,0,0>,<0,1,0>,<0,-1,0>,<0,0,1>,<0,0,-1>}
  :
> A:=<<4,0,0>|<1,4,0>|<0,1,4>>;
```

$$A := \begin{bmatrix} 4 & 1 & 0 \\ 0 & 4 & 1 \\ 0 & 0 & 4 \end{bmatrix} \quad (2)$$

```
> ds1:=deltaDS(DigitSet):
> a2:=Alg36(A,DigitSet,R0);
```

$$a2 := \{[-1, 0, -1], [-1, 0, 0], [-1, 0, 1], [-1, 1, -1], [-1, 1, 0], [0, -1, 0], [0, -1, 1], [0, 0, -1], [0, 0, 0], [0, 0, 1], [0, 1, -1], [0, 1, 0], [1, -1, 0], [1, -1, 1], [1, 0, -1], [1, 0, 0], [1, 0, 1]\} \quad (3)$$

```
> ds2:=makeanylist(ds1):
> a2 intersect ds2;
```

$$\{[-1, 0, -1], [-1, 0, 0], [-1, 0, 1], [-1, 1, -1], [-1, 1, 0], [0, -1, 0], [0, -1, 1], [0, 0, -1], [0, 0, 0], [0, 0, 1], [0, 1, -1], [0, 1, 0], [1, -1, 0], [1, -1, 1], [1, 0, -1], [1, 0, 0], [1, 0, 1]\} \quad (4)$$

Figure 5.30: $\lambda = 4$, $k = 1$.

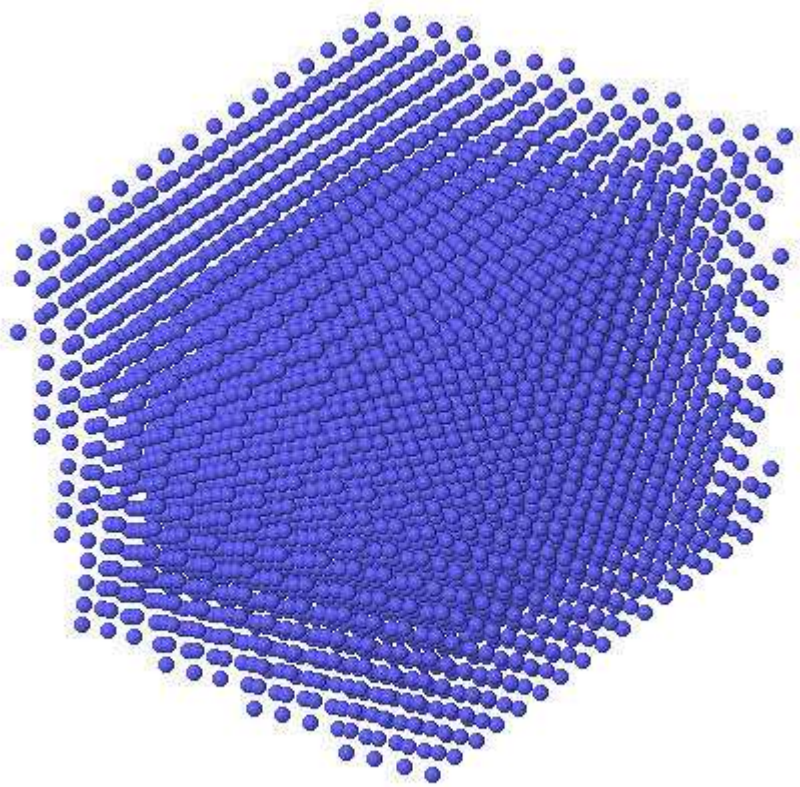


Figure 5.31: T_3 for $\lambda = 4$, $k = 1$.

6 Observations and Future Work

6.1 Observations

In order to get a better idea how the digits and neighbours of our tile $T = T(A, D)$ corresponded, we drew from a large set of examples, which were generated by the Maple program. Although we tried to generalize our results to higher dimensions, we primarily looked at dilation matrices $A \in M_m(\mathbb{Z})$ of the following form:

$$A := \begin{bmatrix} k & \lambda \\ 0 & k \end{bmatrix}. \tag{6.1}$$

From the program, we were able to determine the set of vectors that pair faces with D, the set of neighbours S, the difference set of the digit set ΔD, and the intersection of the difference set and the set of neighbours $S \cap \Delta D$. This gave us an indication that if $S \cap \Delta D$ contained a basis of $\mathbb{Z}^m$, then our tile $T(A, D)$ would be connected. As shown in Proposition 4.2, this turned out to true, in any dimension and for any dilation matrix A.

We also guessed that this would imply that for our type of digit sets, $D = A(F) \cap \mathbb{Z}^m$ [7], and for any basis B for $\mathbb{Z}^m$, that $B \subset \Delta D$ if and only if D

is B-connected. Showing that if D is B-connected, then $B \subset \Delta D$ is clear, but it turned out that the opposite direction is not true. A counterexample to this result is the following matrix and its digit set:

$$A = \begin{bmatrix} 3 & 7 \\ 0 & 3 \end{bmatrix}$$

$$D = \begin{bmatrix} -3 \\ -1 \end{bmatrix}, \begin{bmatrix} -2 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 3 \\ 1 \end{bmatrix}.$$

In this example ΔD is B-connected by the standard basis elements e_1 and e_2 for $\mathbb{Z}^m$, but $S \cap \Delta D$ contains no basis.

This result in itself is interesting, and our examples indicate that in the two dimensional case when $k = \lambda^2 - \lambda$ then we have a connected tile T, yet when $k = \lambda^2 - \lambda + 1$, we no longer have a connected tile. That is, for $k = \lambda^2 - \lambda + 1$, $S \cap \Delta D$ contains no basis for $\mathbb{Z}^m$, whereas when $k = \lambda^2 - \lambda$, $S \cap \Delta D$ does contain a basis. We explored this property experimentally, but have not yet been able to prove this conjecture.

With the aid of the program, we also looked for alternative digit sets D that would yield a connected tile T for any dilation matrix A. We looked at the 'block' digit set for matrices of the form (6.1), where λ is fixed and $k = 0$, and used those digits for any value of k. In general, this did not give rise to connected tiles T. We also looked at digit sets of the form $D' = A^{-1}(F) \cap \mathbb{Z}^m$. We guessed that this might compensate for the skew, and possibly correct the large spread in the digits. However, this digit set also did not give us a connected tile in general.

We tried several other alternative digit sets as the examples indicate, and although there may be other digit sets under which T is connected, the Jordan form was the only successful approach thus far.

6.2 Future Work

Over the course of this project, we encountered several interesting and promising avenues of research that remain to be pursued.

Of primary concern is completion of the proof of Conjecture 4.7.

The most direct extension to our work would involve investigating the connectedness of the digit level sets, D_n, for Jordan form matrices in higher dimensions. The first step would be to show that a similar inequality holds for the $c_{1,n}$'s and the $e_{1,n}$'s as holds in the two-dimensional case. In three dimensions we will have that

$$c_0 = \begin{bmatrix} c_{1,0} \\ c_{2,0} \\ c_{3,0} \end{bmatrix} \quad \text{and} \quad e_0 = \begin{bmatrix} e_{1,0} \\ e_{2,0} \\ e_{3,0} \end{bmatrix}.$$

Inequalities on $c_{2,n}$ and $e_{2,n}$ will likely be an essential part in showing connectedness of T_n, and thus of T, in $\mathbb{R}^3$.

For tiles in $\mathbb{R}^4$, we will have to look at appropriate inequalities between $c_{1,n}$ and $e_{1,n}$, $c_{2,n}$ and $e_{2,n}$, and $c_{3,n}$ and $e_{3,n}$. In $\mathbb{R}^5$, we will have to include an inequality on $c_{4,n}$ and $e_{4,n}$, and so on. Thus, in the m-dimensional case, we will need to consider inequalities between $c_{i,n}$ and $e_{i,n}$ for $0 \leq i \leq m-1$. In this way, we

expect that a pattern will emerge that will allow us to show connectedness in the general m-dimensional case.

For a dilation matrix $A = \begin{bmatrix} \lambda & k \\ 0 & \lambda \end{bmatrix}$, further investigation into the experimental bounds on k such that $T(A, D)$ is connected also promises to be fruitful. It was experimentally observed that when $k = \lambda^2 - \lambda$, experimental results indicate that we have connected tile T, yet when $k = \lambda^2 - \lambda + 1$, we no longer have a connected tile. That is, for $k = \lambda^2 - \lambda + 1$, $S \cap \Delta D$ contains no basis for $\mathbb{Z}^m$, whereas when $k = \lambda^2 - \lambda$, $S \cap \Delta D$ does contain a basis.

Finally, investigating disklikeness properties related to Jordan form matrices may prove more tractable than previous methods.

I hope that this work will be useful to others in pursuing further developments, such as those indicated above.

Bibliography

[1] Akiyama, S. and Gjini, N. On the connectedness of self-affine attractors, *Arch. Math.* **82** (2004), 153-163.

[2] Akiyama, S. and Thuswaldner, J.M. A survey on topological properties of tiles related to number systems, *Geometriae Dedicata* **109**, (2004), 89-105.

[3] Bandt, C. Self-similar sets. *Proc. AMS* **112**, (1991), 549-562.

[4] Bandt, C. and Gelbrich, G. Classification of self-affine lattice tilings. *J. London Math. Soc.* **50**, (1994), 581-593.

[5] Bandt, C. and Wang, Y. Disk-like self-affine tiles in $\mathbb{R}^2$. *Disc. Comput. Geom.* **26** (2001), 591-601.

[6] Barnsley, M.F. and Rising, H. Fractals Everywhere. *Academic Press Professional, Boston* (1993).

[7] Curry, E. Radix and pseudodigit representations in $\mathbb{Z}^n$. preprint.

[8] Falconer, K.J. Techniques in Fractal Geometry. *John Wiley and Sons, Chichester* (1997).

[9] Gröchenig, K. and Haas, A. Self-similar lattice tilings. *J. Fourier Anal. Appl.* **1** (1994), 131-170.

[10] Gröchenig, K. and Madych, W. Multiresolution analysis, Haar bases, and self-similar tilings, *IEEE Trans. Inform. Thy.* **38**, (1992), 556-568.

[11] Kirat, I. and Lau, K.-S.J. On the connectedness of self-affine tiles. *J. London Math. Soc.* (2) **62** (2000), 291-304.

[12] Kirat, I., Lau, K.-S.J., and Rao, H. Expanding polynomials and connectedness of self-affine tiles. *Disc. Comput. Geom.* **31** (2004), 275-286.

[13] Lagarias, J. and Wang, Y. Integral self-affine tiles in $\mathbb{R}^n$. I, Standard and Nonstandard Digit Sets *J. London Math. Soc.* **54**, (1996), 161-179.

[14] Lagarias, J. and Wang, Y. Self-affine tiles in $\mathbb{R}^n$ *Adv. Math.* **121**, (1996), 21-49.

[15] Leung, K.-S. and Lau, K.-S.J. Disklikeness of planar self-affine tiles. *Trans. AMS* **359** (2007), 3337-3355.

[16] Luo, J., Akiyama, S., and Thuswaldner, J.M. On the boundary connectedness of connected tiles. *Math. Proc. Cambridge Phil. Soc.* **137** (2004), 397-410.

[17] Luo, J., Rao, H., and Tan, B. Topological Structure of Self-Similar Sets. *Fractals* (2) **10** (2002), 223-227.

[18] Matula, D.W. Basic digit sets for radix representation. *J. ACM* (4) **29** (1982), 1131-1143.

[19] Ngai, S.-M. and Tang, T.-M. A technique in the topology of connected self-similar tiles. *Fractals* (4) **12** (2004), 389-403.

[20] Scheicher, K. and Thuswaldner, J.M. Neighbours of self-affine tiles in lattice tilings. *P. Grabner and W. Woess (eds), Proceedings of the Conference Fractals in Graz.* (2002), 241-262.

[21] Wang, Y. Self-affine tiles. *K. S. Lau, ed., Advances in Wavelet Analysis. Springer, New York*, (1998), 261-285.

[S+09] W. A. Stein et al., Sage Mathematics Software (Version 3.3), *The Sage Development Team (specific package authors include: Robert Bradshaw, Emily Kirkman, Robert Miller, Networkx, William Stein)*, (2009), `http://www.sagemath.org`.

A Maple Procedures

This appendix contains the Maple code for the implementation of the neighbour finding algorithm as proposed by Scheicher and Thuswalnder [20]. It also contains four Maple files of extended output illustrating how many of the procedures in the package `neighbours` work and what the graphs $G(R)$ and $G(S)$ look like.

When the dimension being considered is greater than 1, we have to use the `verify` command to be able to compare the equality of two vectors. Also, we needed to use the `type(v,'Vector'(datatype =integer))` command to see if the entries in our vector v were integers. This can also be used whe the dimension is equal to 1. `makelist`(`makeanylist`) and `makevector`(`makeanyvector`) allow us to go between sets and vectors, to utilize Maple's built in set manipulation properties, as well as use basic properties of the `linalg` package to optimize speed of `alg36` and `Alg36`. In generating R_n and the initial graph $G(R)$, it is more practical to use matrix multiplication, and matrix inverses. When reducing the graph, considering products, and ultimately computing $G(S)$, it is much more efficient (time and memory) to convert that information into list format. Also redundancy of set elements is automatically avoided in this way.

```
> restart;
> neighbours:=table():
> with(LinearAlgebra):
> with(combinat):
```

```
> with(ListTools):
```

`cart` allows us to take Cartesian products of lists and then further manipulate them as a single list, or, in other words, outputs the Cartesian product as a single list. Maple is not capable of doing exactly this in any built-in function.

```
> neighbours[cart]:=proc();
>     if type([args],list(set)) then
>       if nargs>2 then
>         procname(procname(args[1],args[2]), args[3..-1]);
>       else
>         [ seq(seq( [ `if`(type(i,list),op(i),i),
>         `if`(type(j,list),op(j),j)], j=args[2]), i=args[1])];
>       end if;
>     else
>       error "sequence of lists expected";
>     end if;
> end proc;
```

`deltaDS` is the difference set of our digit set. This is used in almost all relevant literature to our research project, just was not needed in the construction of Algorithm 3.6.

```
> neighbours[deltaDS]:=proc(DS::set)::set;
>     local i,j,diff,DiffS;
>     DiffS:={};
>     for i from 1 to nops(DS) do
>       for j from 1 to nops(DS) do
>         diff:={DS[i]-DS[j]};
>         DiffS:=diff union DiffS;
>       od;
>     od;
>     return DiffS;
> end proc;
```

`sumDS` is a procedure that will take two sets and add all possible combinations of two elements (one from each set) and give a new set with any new elements from adding.

```
> neighbours[sumDS]:=proc(S0::set,S1::set)::set;
>     local i,j,diff1,DiffS1;
>     DiffS1:={};
>     for i from 1 to nops(S0) do
>       for j from 1 to nops(S1) do
>         diff1:={S0[i]+S1[j]};
>         DiffS1:=diff1 union DiffS1;
>       od;
>     od;
>     return DiffS1;
> end proc;
```

`CheckIn` takes a set of vectors and compares it to a single vector. `CheckIn` compares each element in the set to the vector and will return `true` if they are equal and `false` if they are not.

```
> neighbours[CheckIn]:=proc(vectset::Vector,bigList::set)
    ::boolean;
>     local j,yesno,count;
>     yesno:=false;
>     while yesno=false do
>       for j from 1 to nops(bigList) do
>         if verify(vectset,bigList[j],'Vector')=true
          then yesno:=true;
>         fi;
>       od;
>       if yesno=false then break;
>       fi;
>     od;
>     return yesno;
> end proc;
```

`parelist` is intended to remove any redundencies from a set of vectors. `parelist` implements `CheckIn` which checks if the i^{th} element in `biglist`, is already included in the `shortlist` we are generating. If the element under inspection is already in the `shortlist` we move on to the next element in `biglist`, if not we add it, and move on to the next element in `biglist`.

```
> neighbours[parelist]:=proc(biglist::set)::set;
>     local shortlist,i;
>     shortlist:={};
>     for i from 1 to nops(biglist) do
>        if CheckIn(biglist[i],shortlist)=false
>        then shortlist:=shortlist union {biglist[i]};
>        fi;
>     od;
>     return shortlist;
> end proc;
```

`genRn` takes our dilation factor, our digit set and our initial basis and will generate our final basis. That is

$$R_n := \{k \in \mathbb{Z}^m | (Ak + D) \cap (l + D) \neq \emptyset \ for \ l \in R_{n-1}\}.$$

`genRn` will iterate through this set and terminate when $R_n = R_{n-1}$. It applies, `sumDS, subDS, checkRnext`, and `genRnext`. We then take the inverse of each element in `potsol`, uses the `type` command to see whether the solution is an integer or not, and if so adds it to our new set `newelem`. We then use `parelist` to remove any redundancy.

```
> neighbours[genRn]:=proc(DM::Matrix,DS::set,r0::set)::set;
>     local Rntemp,potsol,newel,Rnext,deltDS,i,newelem;
>     Rntemp:=r0;
```

```
>      Rnext:={};
>      newelem:={};
>      deltDS:={};
>      deltDS:=deltaDS(DS);
>      while(true) do
>        potsol:=sumDS(deltDS,Rntemp);
>        newel:=map(x->MatrixInverse(DM).x,potsol);
>        for i from 1 to nops(newel) do
>          if type(newel[i],'Vector'(integer))=true
           then newelem:=newelem union {newel[i]};
>          fi;
>        od;
>        Rnext:=parelist(newelem);
>        if Rnext=Rntemp then break;
>        else Rntemp:=Rnext;
>        fi;
>      od;
>      return Rntemp;
>  end proc;
```

`pairsfaces` performs the same computations as `genRn`, yet stops after the first iteration. This command was useful when comparing *pairs faces* from Gröchenig and Haas, to R_0 in Scheicher and Thuswalnder.

```
>  neighbours[pairsfaces]:=proc(DM::Matrix,DS::set,r0::set)::set;
>      local Rntemp,potsol,newel,Rnext,deltDS,i,newelem;
>      Rntemp:=r0;
>      Rnext:={};
>      newelem:={};
>      deltDS:={};
>      deltDS:=deltaDS(DS);
>      while(true) do
>        potsol:=sumDS(deltDS,Rntemp);
>        newel:=map(x->MatrixInverse(DM).x,potsol);
>        for i from 1 to nops(newel) do
>          if type(newel[i],'Vector'(integer))=true
           then newelem:=newelem union {newel[i]};
```

```
>          fi;
>        od;
>        Rnext:=parelist(newelem);
>        break;
>     od;
>     return Rnext;
>  end proc;
```

`makeall` takes the set R_n, our dilation factor, and our digit set and generates the graph $G(R)$ with the appropriate edge labels. We define the graph $G(R) = (V, E)$ with set of states R in the following way. There exists an edge r to r' labelled by the digit d if $Ar + d' = r' + d$ holds for some $d' \in D$. We can also write this as $Ar + r' = d - d'$ for a $d' \in D$ to keep the form of subDS.

```
>  neighbours[makeall]:=proc(rn::set,DM::Matrix,DS::set)::set;
>     local grgen,i,j,q,r,g;
>     grgen:={};
>     for i from 1 to nops(rn) do
>        for j from 1 to nops(rn) do
>           for q from 1 to nops(DS) do
>              for r from 1 to nops(DS) do
>              g:=verify(DM.rn[i]-rn[j],DS[q]-DS[r],'Vector');
>              if g=true then grgen:=grgen union
              {[rn[i],DS[q],DS[r],rn[j]]};
>              fi;
>              od;
>           od;
>        od;
>     od;
>     return grgen;
>  end proc;
```

`condfunc` is another boolean procedure that will look at the fourth element in the list of four elements generated in `makeall`, regard it as a set, and compare it with some other set element. In our case we will have a set of leaves, where the leaves will be the set of destination elements that are not also source elements. `condfun` will return `true` or `false`.

```
> neighbours[condfunc]:=proc(element::list,leaves::set)
    ::boolean;
>     local h,i;
>     for i from 1 to nops(leaves) do
>       if element[4]=leaves[i] then return false;
>       else return true;
>       fi;
>     od;
> end proc;
```

`RedG` takes all the first elements of our four-part lists, which are the elements of our set $G(R)$ and calls them `Sources`. It takes all of the fourth elements of our four-part lists, which are the elements of our set $G(R)$ and calls them `Destinations`. It makes the set `Leaves`, which is the set of all `Destinations` that are not also `Sources`. It then scans through the fourth elements of every four- element list (the components of our set), and removes any four-element list that has a leaf as its fourth element. This is now the reduced graph, $Red(G)$.

```
> neighbours[RedG]:=proc(GRaph::set)::set;
>     local Sources,Dest,Leaves,Res,Res1,Res2;
>     Res:=GRaph;
>     Leaves:={};
>     while(true) do
>       Res1:=Res;
>       Sources:=Res1[1..nops(Res1),1];
>       Dest:=Res1[1..nops(Res1),4];
>       Leaves:=Dest minus Sources;
>       if Leaves={} then break;
```

```
>        fi;
>        Res:=select(condfunc,Res1,Leaves);
>        if Res1=Res then break;
>        fi;
>      od;
>      return Res;
>  end proc;
```

`Rprime` picks off the first and fourth elements of each of the 4-tuples in $Red(G(R))$. This is the set of `Sources` and `Destinations` that equal one another, and thus does not include any of the destinations that are leaves. These form the set R' , where $G(R') := Red(G(R))$.

```
>  neighbours[Rprime]:=proc(GR8::set)::set;
>      local i,j,Sources,Dest,keep,Res;
>      Res:=GR8;
>      keep:={};
>      Sources:=Res[1..nops(Res),1];
>      Dest:=Res[1..nops(Res),4];
>      for i from 1 to nops(Sources) do
>         for j from 1 to nops(Dest) do
>            if Dest[j]=Sources[i] then
             keep:=keep union {Sources[i]};
>            fi;
>         od;
>      od;
>      return keep;
>  end proc;
```

`prod2GR` takes the product or two graphs G_1 and G'_1, that is, $G_2 = G_1 \otimes G'_1$. Let r_1 and s_1 be states of G_1, and r'_1, and s'_1 be states of G'_1. Furthermore, let d_1, d'_l, d_2, and $d'_2 \in D$. We have denoted edges in G_1 and G'_1 as 4-tuples $[r_1, d_1, d_2, s_1]$ and $[r'_1, d'_1, d'_2, s'_1]$, that is, $A * r_1 - s_1 = d_1 - d_2$ and $A * r'_1 - s'_1 = d'_1 - d'_2$, respectively.

Let r_2 be a state of G_2 if $r_2 = r_1 + r_1'$ for some $r_1 \in G_1$ and $r_1' \in G_1'$.

There exists an edge from r_2 to s_2 in G_2, that is $[r_2, D_1, D_2, s_2]$, if for $[r_1, d_1, d_2, s_1]$ and $[r_1', d_1', d_2', s_1']$ either:

$d_2 = d_1'$ with $r_1 + r_1' = r_2$ and $s_1 + s_1' = s_2$, and $d_1 = D_1$ and $d_2' = D_2$, or,

$d_1 = d_2'$ with $r_1 + r_1' = r_2$ and $s_1 + s_1' = s_2$, and $d_1' = D_1$ and $d_2 = D_2$.

```
> neighbours[prod2GR]:=proc(GR5::set, GR6::set)::set;
>     local cgr,prodgr1,i,C,B,s,t;
>     cgr:=cart(GR5,GR6);
>     C:={};
>     B:={};
>     for i from 1 to nops(cgr) do
>        if cgr[i,2]=cgr[i,7] then C:=C union{[cgr[i,1]+
         cgr[i,5], cgr[i,6],cgr[i,3], cgr[i,4]+cgr[i,8]]};
>           if cgr[i,3]=cgr[i,6] then B:=B union{[cgr[i,1]+
            cgr[i,5], cgr[i,2],cgr[i,7], cgr[i,4]+cgr[i,8]]};
>           fi;
>        fi;
>     od;
>     prodgr1:=C union B;
>     return prodgr1;
> end proc;
```

`makelist` converts $G(R)$, a list of lists of vectors or anything in a similar format, to a list of lists of lists. This makes for quick computation. Maple deals with lists better than with vectors. We can use operations such as a set minus as opposed to having to compare vectors.

```
> neighbours[makelist]:=proc(GR9::set)::set;
>     local newgraph,i,element;
>     newgraph:={};
>     for i from 1 to nops(GR9) do
>        element:=[convert(GR9[i,1],list), convert(GR9[i,2],
          list),convert(GR9[i,3],list),convert(GR9[i,4],list)];
```

```
>        newgraph:=newgraph union{element};
>     od;
>     return newgraph;
>  end proc;
```

`makevector` converts $G(S)$, a list of lists of lists, back to a list of lists of vectors. This is much easier to read than purely list format.

```
>  neighbours[makevector]:=proc(GR9::set)::set;
>     local newgraph,i,element;
>     newgraph:={};
>     for i from 1 to nops(GR9) do
>        element:=[convert(GR9[i,1],Vector),convert(GR9[i,2],
          Vector),convert(GR9[i,3],Vector),convert(GR9[i,4],
          Vector)];
>        newgraph:=newgraph union{element};
>     od;
>     return newgraph;
>  end proc;
```

`makeanyvector` takes any list of lists, and converts it to a list of vectors. This is useful for looking at output of `Rprime`.

```
>  neighbours[makeanyvector]:=proc(GR9::set)::set;
>     local newgraph,i,element;
>     newgraph:={};
>     for i from 1 to nops(GR9) do
>        element:=convert(GR9[i],Vector);
>        newgraph:=newgraph union{element};
>     od;
>     return newgraph;
>  end proc;
```

`makeanylist` takes any list of vectors, and converts it to a list of lists. This is useful for looking at the intersection of the set of neighbours and the difference

set of the digit set.

```
> neighbours[makeanylist]:=proc(GR9::set)::set;
>     local newgraph,i,element;
>     newgraph:={};
>     for i from 1 to nops(GR9) do
>       element:=convert(GR9[i],list);
>       newgraph:=newgraph union{element};
>     od;
>     return newgraph;
> end proc;
```

`alg36` is finally Algorithm 3.6 as described in Scheicher and Thuswaldner's paper. It takes the product of the original graph and its reduced graph (prod2GR($G(R)$, RedG($G(R)$)), applies the reduction to that product, then takes the product of the new graph with the reduction of the original and again reduces. This process continues until the the graph being produced is the same as the one produced in the previous iteration. One then removes the 0 element and this gives us the graph $G(S)$.

```
> neighbours[alg36]:=proc(GR7::set)::set;
>     local p,gengr,grgent,RG,temp;
>     p:=2;
>     RG:=RedG(GR7);
>     temp:= prod2GR(GR7,RG);
>     gengr:=RedG(temp);
>     while (true) do
>       grgent:=gengr;
>       temp := prod2GR(gengr,RG);
>       gengr:=RedG(temp);
>       p:=p+1;
>       if gengr=grgent then break;
>       fi;
>     od;
>     return makevector(gengr);
```

```
> end proc;
```

`Alg36` is a reduced form of `alg36` needing only the dilation matrix `A`, the digit set `DigitSet`, and the set of vectors `R0`, which one must define prior to useing this command. It implements the commands `genRn, makeall, makelist, alg36`, and `Rprime` to find the neighbours in a single command.

```
> neighbours[Alg36]:=proc(DM::Matrix,DS::set,r0::set)::list;
>     local Rn,GR,GR1,a,a1,a2;
>     Rn:=genRn(A,DigitSet,R0);
>     GR:=makeall(Rn,A,DigitSet);
>     GR1:=makelist(GR):
>     a:=alg36(GR1);
>     a1:=makelist(a);
>     a2:=Rprime(a1);
>     return a2;
> end proc;
```

Several other procedures which are useful but not needed procedures for the implementation of Algorithm 3.6:

`subDS` is a procedure that will take two sets and subtract all possible combinations of two elements (one from each set) and give a new set with any new elements found from this 'subtraction'.

```
> neighbours[subDS] := proc (S1::set, S2::set)::set;
>     local i, j, diff2, DiffS2;
>     DiffS2 := {};
>     for i to nops(S2) do
>        for j to nops(S1) do
>           diff2 := S2[i]-S1[j];
>           DiffS2 := union(diff2, DiffS2)
>        end do;
```

```
>      end do;
>      return DiffS2;
>  end proc;
```

`prodnGR` allows one to take the product of the graph $G(R)$ some specified number of times. One may take the product say four times, or 57 times, as opposed to stopping at some limiting step as in `prodGR`.

```
>  neighbours[prodnGR]:=proc(GR3::set,t::integer)::set;
>      local m,ngr;
>      m:=2;
>      ngr:=prod2GR(GR3,GR3);
>      while (m<=t) do
>         ngr:=prod2GR(ngr,GR3);
>         m:=m+1;
>      od;
>      return ngr;
>  end proc;
```

The following commands `cube2ds,cube3ds,` and `cube4ds` generated the digit sets D for the Jordan form of the dilation matrices A in dimensions 2,3, and 4. These digits are the lattice forming a cube about the origin for all values of λ. One only needs to know $\lambda(F) \cap \mathbb{Z}$, as the extension to higher dimension will be copies of the intersection in $\mathbb{R}$. Note that the '`cube`' command can obviously be extended to higher dimensions, but Maple could no longer compute the examples when the dimension was greater than or equal to 5.

`cube2ds` takes a list of n- elements and will generate all two-tuples of combinations of these elements.

```
>  neighbours[cube2ds]:=proc(p::list)::list;
>      local i,j,n,addp,p1,p2,p3,p4;
```

```
>     p1:={};
>     p4:={};
>     for i from 1 to nops(p) do
>       for j from 1 to nops(p) do
>         p2:=[op(i,p),op(j,p)];
>         p1:=p1 union p2;
>       od;
>     od;
>     p3:=convert(p1,list);
>     for n from 1 to nops(p3) do
>       addp:=convert(op(n,p3),Vector);
>       p4:= p4 union addp;
>     od;
>     return p4;
> end proc;
```

`cube3ds` takes a list of n- elements and will generate all three-tuples of combinations of these elements.

```
> neighbours[cube3ds]:=proc(p::list)::list;
>     local i,j,k,n,addp,p1,p2,p3,p4;
>     p1:={};
>     p4:={};
>     for i from 1 to nops(p) do
>       for j from 1 to nops(p) do
>         for k from 1 to nops(p) do
>           p2:=[op(i,p),op(j,p),op(k,p)];
>           p1:=p1 union p2;
>         od;
>       od;
>     od;
>     p3:=convert(p1,list);
>     for n from 1 to nops(p3) do
>       addp:=convert(op(n,p3),Vector);
>       p4:= p4 union addp;
>     od;
>     return p4;
```

```
> end proc;
```

`cube4ds` takes a list of n- elements and will generate all four-tuples of combinations of these elements.

```
> neighbours[cube4ds]:=proc(p::list)::list;
>     local i,j,k,n,addp,p1,p2,p3,p4;
>     p1:={};
>     p4:={};
>     for i from 1 to nops(p) do
>       for j from 1 to nops(p) do
>         for k from 1 to nops(p) do
>           for l from 1 to nops(p) do
>             p2:=[op(i,p),op(j,p),op(k,p),op(l,p)];
>             p1:=p1 union p2;
>           od;
>         od;
>       od;
>     od;
>     p3:=convert(p1,list);
>     for n from 1 to nops(p3) do
>       addp:=convert(op(n,p3),Vector);
>       p4:= p4 union addp;
>     od;
>     return p4;
> end proc;
```

These final lines of input save all of these procedures as a package in the Maple Library.

```
> save(neighbours,"/Library/Frameworks//
  Maple.framework/Versions/12/lib/neighbours.m");
> save(neighbours,"/Library/Frameworks//
  Maple.framework/Versions/12/lib/neighbours.mw");
```

One can now call upon the package `neighbours`, as one does with the package, say, `Student[Calculus1]`. One must type the following two commands to begin to use the package. The `libname` command accesses custom packages, and now one can type simply `with(neighbours)`, as one does with, say `with(Student[Calculus1])`.

```
> libname:="/Library/Frameworks//
  Maple.framework/Versions/12",libname;
> with(neighbours);
```

Appendix-Example 1:

```
> DigitSet:={<0,0>,<1,0>,<-1,0>,<-1,1>,<0,1>,<1,1>,<1,-1>,<0,-1>,
  <-1,-1>};
```

$$DigitSet := \left\{\begin{bmatrix}-1\\-1\end{bmatrix}, \begin{bmatrix}-1\\0\end{bmatrix}, \begin{bmatrix}-1\\1\end{bmatrix}, \begin{bmatrix}0\\-1\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}1\\-1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix}\right\} \quad (1)$$

```
> R0:={<0,0>,<1,0>,<-1,0>,<0,1>,<0,-1>};
```

$$R0 := \left\{\begin{bmatrix}-1\\0\end{bmatrix}, \begin{bmatrix}0\\-1\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}\right\} \quad (2)$$

```
> A:=<<3,0>|<0,3>>;
```

$$A := \begin{bmatrix}3 & 0\\0 & 3\end{bmatrix} \quad (3)$$

```
> ds1:=deltaDS(DigitSet);
```

$$ds1 := \left\{\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\-1\end{bmatrix}, \begin{bmatrix}0\\-2\end{bmatrix}, \begin{bmatrix}-1\\0\end{bmatrix}, \begin{bmatrix}-1\\-1\end{bmatrix}, \begin{bmatrix}-1\\-2\end{bmatrix}, \begin{bmatrix}-2\\0\end{bmatrix}, \begin{bmatrix}-2\\-1\end{bmatrix}, \begin{bmatrix}-2\\-2\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}, \right. \quad (4)$$

$$\begin{bmatrix}0\\-1\end{bmatrix}, \begin{bmatrix}-1\\1\end{bmatrix}, \begin{bmatrix}-1\\0\end{bmatrix}, \begin{bmatrix}-1\\-1\end{bmatrix}, \begin{bmatrix}-2\\1\end{bmatrix}, \begin{bmatrix}-2\\0\end{bmatrix}, \begin{bmatrix}-2\\-1\end{bmatrix}, \begin{bmatrix}0\\2\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}-1\\2\end{bmatrix}, \begin{bmatrix}-1\\1\end{bmatrix},$$

$$\begin{bmatrix}-1\\0\end{bmatrix}, \begin{bmatrix}-2\\2\end{bmatrix}, \begin{bmatrix}-2\\1\end{bmatrix}, \begin{bmatrix}-2\\0\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}1\\-1\end{bmatrix}, \begin{bmatrix}1\\-2\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\-1\end{bmatrix}, \begin{bmatrix}0\\-2\end{bmatrix}, \begin{bmatrix}-1\\0\end{bmatrix},$$

$$\begin{bmatrix}-1\\-1\end{bmatrix}, \begin{bmatrix}-1\\-2\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}1\\-1\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\-1\end{bmatrix}, \begin{bmatrix}-1\\1\end{bmatrix}, \begin{bmatrix}-1\\0\end{bmatrix}, \begin{bmatrix}-1\\-1\end{bmatrix}, \begin{bmatrix}1\\2\end{bmatrix},$$

$$\begin{bmatrix}1\\1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}0\\2\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}-1\\2\end{bmatrix}, \begin{bmatrix}-1\\1\end{bmatrix}, \begin{bmatrix}-1\\0\end{bmatrix}, \begin{bmatrix}2\\0\end{bmatrix}, \begin{bmatrix}2\\-1\end{bmatrix}, \begin{bmatrix}2\\-2\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix},$$

$$\begin{bmatrix}1\\-1\end{bmatrix}, \begin{bmatrix}1\\-2\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\-1\end{bmatrix}, \begin{bmatrix}0\\-2\end{bmatrix}, \begin{bmatrix}2\\1\end{bmatrix}, \begin{bmatrix}2\\0\end{bmatrix}, \begin{bmatrix}2\\-1\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}1\\-1\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix},$$

$$\left.\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\-1\end{bmatrix}, \begin{bmatrix}2\\2\end{bmatrix}, \begin{bmatrix}2\\1\end{bmatrix}, \begin{bmatrix}2\\0\end{bmatrix}, \begin{bmatrix}1\\2\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}0\\2\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}\right\}$$

```
> pf:=pairsfaces(A,DigitSet,R0);
```

$$pf := \left\{\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}0\\-1\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}-1\\0\end{bmatrix}\right\} \quad (5)$$

```
> Rn:=genRn(A,DigitSet,R0);
```

$$Rn := \left\{\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}-1\\0\end{bmatrix}, \begin{bmatrix}0\\-1\end{bmatrix}\right\} \quad (6)$$

```
> GR:=makeall(Rn,A,DigitSet):
> GR1:=makelist(GR);
```

$$GR1 := \{[[-1,0],[-1,-1],[1,-1],[-1,0]],[[-1,0],[-1,0],[1,0],[-1,0]],[[-1,0], \quad (7)$$

```
[-1, 1], [1, 1], [-1, 0]], [[0, -1], [-1, -1], [-1, 1], [0, -1]], [[0, -1], [0, -1], [0, 1],
[0, -1]], [[0, -1], [1, -1], [1, 1], [0, -1]], [[0, 0], [-1, -1], [-1, -1], [0, 0]], [[0,
0], [-1, -1], [-1, 0], [0, 1]], [[0, 0], [-1, -1], [0, -1], [1, 0]], [[0, 0], [-1, 0], [-1,
-1], [0, -1]], [[0, 0], [-1, 0], [-1, 0], [0, 0]], [[0, 0], [-1, 0], [-1, 1], [0, 1]], [[0,
0], [-1, 0], [0, 0], [1, 0]], [[0, 0], [-1, 1], [-1, 0], [0, -1]], [[0, 0], [-1, 1], [-1, 1],
[0, 0]], [[0, 0], [-1, 1], [0, 1], [1, 0]], [[0, 0], [0, -1], [-1, -1], [-1, 0]], [[0, 0], [0,
-1], [0, -1], [0, 0]], [[0, 0], [0, -1], [0, 0], [0, 1]], [[0, 0], [0, -1], [1, -1], [1, 0]],
[[0, 0], [0, 0], [-1, 0], [-1, 0]], [[0, 0], [0, 0], [0, -1], [0, -1]], [[0, 0], [0, 0], [0,
0], [0, 0]], [[0, 0], [0, 0], [0, 1], [0, 1]], [[0, 0], [0, 0], [1, 0], [1, 0]], [[0, 0], [0, 1], [
-1, 1], [-1, 0]], [[0, 0], [0, 1], [0, 0], [0, -1]], [[0, 0], [0, 1], [0, 1], [0, 0]], [[0, 0],
[0, 1], [1, 1], [1, 0]], [[0, 0], [1, -1], [0, -1], [-1, 0]], [[0, 0], [1, -1], [1, -1], [0,
0]], [[0, 0], [1, -1], [1, 0], [0, 1]], [[0, 0], [1, 0], [0, 0], [-1, 0]], [[0, 0], [1, 0], [1,
-1], [0, -1]], [[0, 0], [1, 0], [1, 0], [0, 0]], [[0, 0], [1, 0], [1, 1], [0, 1]], [[0, 0], [1,
1], [0, 1], [-1, 0]], [[0, 0], [1, 1], [1, 0], [0, -1]], [[0, 0], [1, 1], [1, 1], [0, 0]], [[0,
1], [-1, 1], [-1, -1], [0, 1]], [[0, 1], [0, 1], [0, -1], [0, 1]], [[0, 1], [1, 1], [1, -1],
[0, 1]], [[1, 0], [1, -1], [-1, -1], [1, 0]], [[1, 0], [1, 0], [-1, 0], [1, 0]], [[1, 0], [1,
1], [-1, 1], [1, 0]]}
> RGR:=makelist(RedG(GR));
RGR := {[[-1, 0], [-1, -1], [1, -1], [-1, 0]], [[-1, 0], [-1, 0], [1, 0], [-1, 0]], [[-1, 0],   (8)
[-1, 1], [1, 1], [-1, 0]], [[0, -1], [-1, -1], [-1, 1], [0, -1]], [[0, -1], [0, -1], [0, 1],
[0, -1]], [[0, -1], [1, -1], [1, 1], [0, -1]], [[0, 0], [-1, -1], [-1, -1], [0, 0]], [[0,
0], [-1, -1], [-1, 0], [0, 1]], [[0, 0], [-1, -1], [0, -1], [1, 0]], [[0, 0], [-1, 0], [-1,
-1], [0, -1]], [[0, 0], [-1, 0], [-1, 0], [0, 0]], [[0, 0], [-1, 0], [-1, 1], [0, 1]], [[0,
0], [-1, 0], [0, 0], [1, 0]], [[0, 0], [-1, 1], [-1, 0], [0, -1]], [[0, 0], [-1, 1], [-1, 1],
[0, 0]], [[0, 0], [-1, 1], [0, 1], [1, 0]], [[0, 0], [0, -1], [-1, -1], [-1, 0]], [[0, 0], [0,
-1], [0, -1], [0, 0]], [[0, 0], [0, -1], [0, 0], [0, 1]], [[0, 0], [0, -1], [1, -1], [1, 0]],
[[0, 0], [0, 0], [-1, 0], [-1, 0]], [[0, 0], [0, 0], [0, -1], [0, -1]], [[0, 0], [0, 0], [0,
0], [0, 0]], [[0, 0], [0, 0], [0, 1], [0, 1]], [[0, 0], [0, 0], [1, 0], [1, 0]], [[0, 0], [0, 1], [
-1, 1], [-1, 0]], [[0, 0], [0, 1], [0, 0], [0, -1]], [[0, 0], [0, 1], [0, 1], [0, 0]], [[0, 0],
[0, 1], [1, 1], [1, 0]], [[0, 0], [1, -1], [0, -1], [-1, 0]], [[0, 0], [1, -1], [1, -1], [0,
0]], [[0, 0], [1, -1], [1, 0], [0, 1]], [[0, 0], [1, 0], [0, 0], [-1, 0]], [[0, 0], [1, 0], [1,
-1], [0, -1]], [[0, 0], [1, 0], [1, 0], [0, 0]], [[0, 0], [1, 0], [1, 1], [0, 1]], [[0, 0], [1,
1], [0, 1], [-1, 0]], [[0, 0], [1, 1], [1, 0], [0, -1]], [[0, 0], [1, 1], [1, 1], [0, 0]], [[0,
1], [-1, 1], [-1, -1], [0, 1]], [[0, 1], [0, 1], [0, -1], [0, 1]], [[0, 1], [1, 1], [1, -1],
[0, 1]], [[1, 0], [1, -1], [-1, -1], [1, 0]], [[1, 0], [1, 0], [-1, 0], [1, 0]], [[1, 0], [1,
1], [-1, 1], [1, 0]]}
> a:=alg36(GR1):
> a1:=makelist(a);
a1 := {[[-1, -1], [-1, -1], [1, 1], [-1, -1]], [[-1, 0], [-1, -1], [1, -1], [-1, 0]], [[-1,   (9)
```

```
0], [-1, -1], [1, 0], [-1, 1]], [[-1, 0], [-1, 0], [1, -1], [-1, -1]], [[-1, 0], [-1, 0],
[1, 0], [-1, 0]], [[-1, 0], [-1, 0], [1, 1], [-1, 1]], [[-1, 0], [-1, 1], [1, 0], [-1, -1]],
[[-1, 0], [-1, 1], [1, 1], [-1, 0]], [[-1, 1], [-1, 1], [1, -1], [-1, 1]], [[0, -1], [-1,
-1], [-1, 1], [0, -1]], [[0, -1], [-1, -1], [0, 1], [1, -1]], [[0, -1], [0, -1], [-1, 1], [
-1, -1]], [[0, -1], [0, -1], [0, 1], [0, -1]], [[0, -1], [0, -1], [1, 1], [1, -1]], [[0,
-1], [1, -1], [0, 1], [-1, -1]], [[0, -1], [1, -1], [1, 1], [0, -1]], [[0, 0], [-1, -1], [
-1, -1], [0, 0]], [[0, 0], [-1, -1], [-1, 0], [0, 1]], [[0, 0], [-1, -1], [0, -1], [1, 0]],
[[0, 0], [-1, -1], [0, 0], [1, 1]], [[0, 0], [-1, 0], [-1, -1], [0, -1]], [[0, 0], [-1, 0], [
-1, 0], [0, 0]], [[0, 0], [-1, 0], [-1, 1], [0, 1]], [[0, 0], [-1, 0], [0, -1], [1, -1]], [[0,
0], [-1, 0], [0, 0], [1, 0]], [[0, 0], [-1, 0], [0, 1], [1, 1]], [[0, 0], [-1, 1], [-1, 0], [0,
-1]], [[0, 0], [-1, 1], [-1, 1], [0, 0]], [[0, 0], [-1, 1], [0, 0], [1, -1]], [[0, 0], [-1,
1], [0, 1], [1, 0]], [[0, 0], [0, -1], [-1, -1], [-1, 0]], [[0, 0], [0, -1], [-1, 0], [-1,
1]], [[0, 0], [0, -1], [0, -1], [0, 0]], [[0, 0], [0, -1], [0, 0], [0, 1]], [[0, 0], [0, -1],
[1, -1], [1, 0]], [[0, 0], [0, -1], [1, 0], [1, 1]], [[0, 0], [0, 0], [-1, -1], [-1, -1]],
[[0, 0], [0, 0], [-1, 0], [-1, 0]], [[0, 0], [0, 0], [-1, 1], [-1, 1]], [[0, 0], [0, 0], [0,
-1], [0, -1]], [[0, 0], [0, 0], [0, 0], [0, 0]], [[0, 0], [0, 0], [0, 1], [0, 1]], [[0, 0], [0,
0], [1, -1], [1, -1]], [[0, 0], [0, 0], [1, 0], [1, 0]], [[0, 0], [0, 0], [1, 1], [1, 1]], [[0,
0], [0, 1], [-1, 0], [-1, -1]], [[0, 0], [0, 1], [-1, 1], [-1, 0]], [[0, 0], [0, 1], [0, 0],
[0, -1]], [[0, 0], [0, 1], [0, 1], [0, 0]], [[0, 0], [0, 1], [1, 0], [1, -1]], [[0, 0], [0, 1],
[1, 1], [1, 0]], [[0, 0], [1, -1], [0, -1], [-1, 0]], [[0, 0], [1, -1], [0, 0], [-1, 1]], [[0,
0], [1, -1], [1, -1], [0, 0]], [[0, 0], [1, -1], [1, 0], [0, 1]], [[0, 0], [1, 0], [0, -1], [
-1, -1]], [[0, 0], [1, 0], [0, 0], [-1, 0]], [[0, 0], [1, 0], [0, 1], [-1, 1]], [[0, 0], [1, 0],
[1, -1], [0, -1]], [[0, 0], [1, 0], [1, 0], [0, 0]], [[0, 0], [1, 0], [1, 1], [0, 1]], [[0, 0],
[1, 1], [0, 0], [-1, -1]], [[0, 0], [1, 1], [0, 1], [-1, 0]], [[0, 0], [1, 1], [1, 0], [0, -1]],
[[0, 0], [1, 1], [1, 1], [0, 0]], [[0, 1], [-1, 1], [-1, -1], [0, 1]], [[0, 1], [-1, 1], [0,
-1], [1, 1]], [[0, 1], [0, 1], [-1, -1], [-1, 1]], [[0, 1], [0, 1], [0, -1], [0, 1]], [[0, 1],
[0, 1], [1, -1], [1, 1]], [[0, 1], [1, 1], [0, -1], [-1, 1]], [[0, 1], [1, 1], [1, -1], [0, 1]],
[[1, -1], [1, -1], [-1, 1], [1, -1]], [[1, 0], [1, -1], [-1, -1], [1, 0]], [[1, 0], [1, -1],
[-1, 0], [1, 1]], [[1, 0], [1, 0], [-1, -1], [1, -1]], [[1, 0], [1, 0], [-1, 0], [1, 0]], [[1,
0], [1, 0], [-1, 1], [1, 1]], [[1, 0], [1, 1], [-1, 0], [1, -1]], [[1, 0], [1, 1], [-1, 1], [1,
0]], [[1, 1], [1, 1], [-1, -1], [1, 1]]}
> a2:=Rprime(a1);
   a2 := {[-1, -1], [-1, 0], [-1, 1], [0, -1], [0, 0], [0, 1], [1, -1], [1, 0], [1, 1]}        (10)
> Alg36(A,DigitSet,R0);
       {[-1, -1], [-1, 0], [-1, 1], [0, -1], [0, 0], [0, 1], [1, -1], [1, 0], [1, 1]}        (11)
> ds2:=makeanylist(ds1):
> a2 intersect ds2;
       {[-1, -1], [-1, 0], [-1, 1], [0, -1], [0, 0], [0, 1], [1, -1], [1, 0], [1, 1]}        (12)
```

Figure A.1: Extended output for 3003

Appendix-Example 2:

```
> DigitSet:={<0,0>,<1,0>,<-1,0>,<1,1>,<0,1>,<-1,1>,<1,-1>,<0,-1>,
  <-1,-1>};
```

$$DigitSet := \left\{\begin{bmatrix}-1\\-1\end{bmatrix}, \begin{bmatrix}-1\\0\end{bmatrix}, \begin{bmatrix}-1\\1\end{bmatrix}, \begin{bmatrix}0\\-1\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}1\\-1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix}\right\} \quad (1)$$

```
> R0:={<0,0>,<1,0>,<-1,0>,<0,1>,<0,-1>};
```

$$R0 := \left\{\begin{bmatrix}-1\\0\end{bmatrix}, \begin{bmatrix}0\\-1\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}\right\} \quad (2)$$

```
> A:=<<3,0>|<1,3>>;
```

$$A := \begin{bmatrix}3 & 1\\0 & 3\end{bmatrix} \quad (3)$$

```
> ds1:=deltaDS(DigitSet);
```

$$ds1 := \left\{\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\-1\end{bmatrix}, \begin{bmatrix}0\\-2\end{bmatrix}, \begin{bmatrix}-1\\0\end{bmatrix}, \begin{bmatrix}-1\\-1\end{bmatrix}, \begin{bmatrix}-1\\-2\end{bmatrix}, \begin{bmatrix}-2\\0\end{bmatrix}, \begin{bmatrix}-2\\-1\end{bmatrix}, \begin{bmatrix}-2\\-2\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}, \right. \quad (4)$$

$$\begin{bmatrix}0\\-1\end{bmatrix}, \begin{bmatrix}-1\\1\end{bmatrix}, \begin{bmatrix}-1\\0\end{bmatrix}, \begin{bmatrix}-1\\-1\end{bmatrix}, \begin{bmatrix}-2\\1\end{bmatrix}, \begin{bmatrix}-2\\0\end{bmatrix}, \begin{bmatrix}-2\\-1\end{bmatrix}, \begin{bmatrix}0\\2\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}-1\\2\end{bmatrix}, \begin{bmatrix}-1\\1\end{bmatrix},$$

$$\begin{bmatrix}-1\\0\end{bmatrix}, \begin{bmatrix}-2\\2\end{bmatrix}, \begin{bmatrix}-2\\1\end{bmatrix}, \begin{bmatrix}-2\\0\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}1\\-1\end{bmatrix}, \begin{bmatrix}1\\-2\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\-1\end{bmatrix}, \begin{bmatrix}0\\-2\end{bmatrix}, \begin{bmatrix}-1\\0\end{bmatrix},$$

$$\begin{bmatrix}-1\\-1\end{bmatrix}, \begin{bmatrix}-1\\-2\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}1\\-1\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\-1\end{bmatrix}, \begin{bmatrix}-1\\1\end{bmatrix}, \begin{bmatrix}-1\\0\end{bmatrix}, \begin{bmatrix}-1\\-1\end{bmatrix}, \begin{bmatrix}1\\2\end{bmatrix},$$

$$\begin{bmatrix}1\\1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}0\\2\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}-1\\2\end{bmatrix}, \begin{bmatrix}-1\\1\end{bmatrix}, \begin{bmatrix}-1\\0\end{bmatrix}, \begin{bmatrix}2\\0\end{bmatrix}, \begin{bmatrix}2\\-1\end{bmatrix}, \begin{bmatrix}2\\-2\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix},$$

$$\begin{bmatrix}1\\-1\end{bmatrix}, \begin{bmatrix}1\\-2\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\-1\end{bmatrix}, \begin{bmatrix}0\\-2\end{bmatrix}, \begin{bmatrix}2\\1\end{bmatrix}, \begin{bmatrix}2\\0\end{bmatrix}, \begin{bmatrix}2\\-1\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}1\\-1\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix},$$

$$\left.\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}0\\-1\end{bmatrix}, \begin{bmatrix}2\\2\end{bmatrix}, \begin{bmatrix}2\\1\end{bmatrix}, \begin{bmatrix}2\\0\end{bmatrix}, \begin{bmatrix}1\\2\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}0\\2\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}\right\}$$

```
> pf:=pairsfaces(A,DigitSet,R0);
```

$$pf := \left\{\begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}-1\\0\end{bmatrix}, \begin{bmatrix}0\\-1\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}-1\\1\end{bmatrix}, \begin{bmatrix}1\\-1\end{bmatrix}\right\} \quad (5)$$

```
> Rn:=genRn(A,DigitSet,R0);
```

$$Rn := \left\{\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}-1\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}1\\-1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}0\\-1\end{bmatrix}, \begin{bmatrix}-1\\1\end{bmatrix}\right\} \quad (6)$$

```
> GR:=makeall(Rn,A,DigitSet):
> GR1:=makelist(GR);
```

$$GR1 := \{[[-1, 0], [-1, -1], [1, -1], [-1, 0]], [[-1, 0], [-1, -1], [1, 0], [-1, 1]], [[-1, \quad (7)$$

```
0], [-1, 0], [1, 0], [-1, 0]], [[-1, 0], [-1, 0], [1, 1], [-1, 1]], [[-1, 0], [-1, 1], [1,
1], [-1, 0]], [[-1, 1], [-1, 1], [0, -1], [-1, 1]], [[-1, 1], [-1, 1], [1, -1], [0, 1]], [[
-1, 1], [0, 1], [1, -1], [-1, 1]], [[0, -1], [-1, -1], [0, 1], [0, -1]], [[0, -1], [-1,
-1], [1, 1], [1, -1]], [[0, -1], [0, -1], [1, 1], [0, -1]], [[0, 0], [-1, -1], [-1, -1], [0,
0]], [[0, 0], [-1, -1], [-1, 0], [0, 1]], [[0, 0], [-1, -1], [0, -1], [1, 0]], [[0, 0], [-1,
0], [-1, -1], [0, -1]], [[0, 0], [-1, 0], [-1, 0], [0, 0]], [[0, 0], [-1, 0], [-1, 1], [0,
1]], [[0, 0], [-1, 0], [0, -1], [1, -1]], [[0, 0], [-1, 0], [0, 0], [1, 0]], [[0, 0], [-1, 1],
[-1, 0], [0, -1]], [[0, 0], [-1, 1], [-1, 1], [0, 0]], [[0, 0], [-1, 1], [0, 0], [1, -1]],
[[0, 0], [-1, 1], [0, 1], [1, 0]], [[0, 0], [0, -1], [-1, -1], [-1, 0]], [[0, 0], [0, -1], [
-1, 0], [-1, 1]], [[0, 0], [0, -1], [0, -1], [0, 0]], [[0, 0], [0, -1], [0, 0], [0, 1]], [[0,
0], [0, -1], [1, -1], [1, 0]], [[0, 0], [0, 0], [-1, 0], [-1, 0]], [[0, 0], [0, 0], [-1, 1], [
-1, 1]], [[0, 0], [0, 0], [0, -1], [0, -1]], [[0, 0], [0, 0], [0, 0], [0, 0]], [[0, 0], [0, 0],
[0, 1], [0, 1]], [[0, 0], [0, 0], [1, -1], [1, -1]], [[0, 0], [0, 0], [1, 0], [1, 0]], [[0, 0],
[0, 1], [-1, 1], [-1, 0]], [[0, 0], [0, 1], [0, 0], [0, -1]], [[0, 0], [0, 1], [0, 1], [0, 0]],
[[0, 0], [0, 1], [1, 0], [1, -1]], [[0, 0], [0, 1], [1, 1], [1, 0]], [[0, 0], [1, -1], [0, -1], [
-1, 0]], [[0, 0], [1, -1], [0, 0], [-1, 1]], [[0, 0], [1, -1], [1, -1], [0, 0]], [[0, 0], [1,
-1], [1, 0], [0, 1]], [[0, 0], [1, 0], [0, 0], [-1, 0]], [[0, 0], [1, 0], [0, 1], [-1, 1]], [[0,
0], [1, 0], [1, -1], [0, -1]], [[0, 0], [1, 0], [1, 0], [0, 0]], [[0, 0], [1, 0], [1, 1], [0,
1]], [[0, 0], [1, 1], [0, 1], [-1, 0]], [[0, 0], [1, 1], [1, 0], [0, -1]], [[0, 0], [1, 1], [1,
1], [0, 0]], [[0, 1], [0, 1], [-1, -1], [0, 1]], [[0, 1], [1, 1], [-1, -1], [-1, 1]], [[0, 1],
[1, 1], [0, -1], [0, 1]], [[1, -1], [0, -1], [-1, 1], [1, -1]], [[1, -1], [1, -1], [-1, 1],
[0, -1]], [[1, -1], [1, -1], [0, 1], [1, -1]], [[1, 0], [1, -1], [-1, -1], [1, 0]], [[1, 0],
[1, 0], [-1, -1], [1, -1]], [[1, 0], [1, 0], [-1, 0], [1, 0]], [[1, 0], [1, 1], [-1, 0], [1,
-1]], [[1, 0], [1, 1], [-1, 1], [1, 0]]}
> RGR:=makelist(RedG(GR));
RGR := {[[-1, 0], [-1, -1], [1, -1], [-1, 0]], [[-1, 0], [-1, -1], [1, 0], [-1, 1]], [[-1,                (8)
0], [-1, 0], [1, 0], [-1, 0]], [[-1, 0], [-1, 0], [1, 1], [-1, 1]], [[-1, 0], [-1, 1], [1,
1], [-1, 0]], [[-1, 1], [-1, 1], [0, -1], [-1, 1]], [[-1, 1], [-1, 1], [1, -1], [0, 1]], [[
-1, 1], [0, 1], [1, -1], [-1, 1]], [[0, -1], [-1, -1], [0, 1], [0, -1]], [[0, -1], [-1,
-1], [1, 1], [1, -1]], [[0, -1], [0, -1], [1, 1], [0, -1]], [[0, 0], [-1, -1], [-1, -1], [0,
0]], [[0, 0], [-1, -1], [-1, 0], [0, 1]], [[0, 0], [-1, -1], [0, -1], [1, 0]], [[0, 0], [-1,
0], [-1, -1], [0, -1]], [[0, 0], [-1, 0], [-1, 0], [0, 0]], [[0, 0], [-1, 0], [-1, 1], [0,
1]], [[0, 0], [-1, 0], [0, -1], [1, -1]], [[0, 0], [-1, 0], [0, 0], [1, 0]], [[0, 0], [-1, 1],
[-1, 0], [0, -1]], [[0, 0], [-1, 1], [-1, 1], [0, 0]], [[0, 0], [-1, 1], [0, 0], [1, -1]],
[[0, 0], [-1, 1], [0, 1], [1, 0]], [[0, 0], [0, -1], [-1, -1], [-1, 0]], [[0, 0], [0, -1], [
-1, 0], [-1, 1]], [[0, 0], [0, -1], [0, -1], [0, 0]], [[0, 0], [0, -1], [0, 0], [0, 1]], [[0,
0], [0, -1], [1, -1], [1, 0]], [[0, 0], [0, 0], [-1, 0], [-1, 0]], [[0, 0], [0, 0], [-1, 1], [
-1, 1]], [[0, 0], [0, 0], [0, -1], [0, -1]], [[0, 0], [0, 0], [0, 0], [0, 0]], [[0, 0], [0, 0],
```

```
    [0, 1], [0, 1]], [[0, 0], [0, 0], [1, -1], [1, -1]], [[0, 0], [0, 0], [1, 0], [1, 0]], [[0, 0],
    [0, 1], [-1, 1], [-1, 0]], [[0, 0], [0, 1], [0, 0], [0, -1]], [[0, 0], [0, 1], [0, 1], [0, 0]],
    [[0, 0], [0, 1], [1, 0], [1, -1]], [[0, 0], [0, 1], [1, 1], [1, 0]], [[0, 0], [1, -1], [0, -1], [
    -1, 0]], [[0, 0], [1, -1], [0, 0], [-1, 1]], [[0, 0], [1, -1], [1, -1], [0, 0]], [[0, 0], [1,
    -1], [1, 0], [0, 1]], [[0, 0], [1, 0], [0, 0], [-1, 0]], [[0, 0], [1, 0], [0, 1], [-1, 1]], [[0,
    0], [1, 0], [1, -1], [0, -1]], [[0, 0], [1, 0], [1, 0], [0, 0]], [[0, 0], [1, 0], [1, 1], [0,
    1]], [[0, 0], [1, 1], [0, 1], [-1, 0]], [[0, 0], [1, 1], [1, 0], [0, -1]], [[0, 0], [1, 1], [1,
    1], [0, 0]], [[0, 1], [0, 1], [-1, -1], [0, 1]], [[0, 1], [1, 1], [-1, -1], [-1, 1]], [[0, 1],
    [1, 1], [0, -1], [0, 1]], [[1, -1], [0, -1], [-1, 1], [1, -1]], [[1, -1], [1, -1], [-1, 1],
    [0, -1]], [[1, -1], [1, -1], [0, 1], [1, -1]], [[1, 0], [1, -1], [-1, -1], [1, 0]], [[1, 0],
    [1, 0], [-1, -1], [1, -1]], [[1, 0], [1, 0], [-1, 0], [1, 0]], [[1, 0], [1, 1], [-1, 0], [1,
    -1]], [[1, 0], [1, 1], [-1, 1], [1, 0]]}
> a:=alg36(GR1):
> a1:=makelist(a);
```

a1 := {[[-1, 0], [-1, -1], [1, -1], [-1, 0]], [[-1, 0], [-1, -1], [1, 0], [-1, 1]], [[-1, 0], [**(9)**
-1, 0], [1, 0], [-1, 0]], [[-1, 0], [-1, 0], [1, 1], [-1, 1]], [[-1, 0], [-1, 1], [1, 1], [
-1, 0]], [[-1, 1], [-1, 1], [0, -1], [-1, 1]], [[-1, 1], [-1, 1], [1, -1], [0, 1]], [[-1,
1], [0, 1], [1, -1], [-1, 1]], [[0, -1], [-1, -1], [0, 1], [0, -1]], [[0, -1], [-1, -1], [1,
1], [1, -1]], [[0, -1], [0, -1], [1, 1], [0, -1]], [[0, 0], [-1, -1], [-1, -1], [0, 0]],
[[0, 0], [-1, -1], [-1, 0], [0, 1]], [[0, 0], [-1, -1], [0, -1], [1, 0]], [[0, 0], [-1, 0], [
-1, -1], [0, -1]], [[0, 0], [-1, 0], [-1, 0], [0, 0]], [[0, 0], [-1, 0], [-1, 1], [0, 1]],
[[0, 0], [-1, 0], [0, -1], [1, -1]], [[0, 0], [-1, 0], [0, 0], [1, 0]], [[0, 0], [-1, 1], [-1,
0], [0, -1]], [[0, 0], [-1, 1], [-1, 1], [0, 0]], [[0, 0], [-1, 1], [0, 0], [1, -1]], [[0, 0],
[-1, 1], [0, 1], [1, 0]], [[0, 0], [0, -1], [-1, -1], [-1, 0]], [[0, 0], [0, -1], [-1, 0], [
-1, 1]], [[0, 0], [0, -1], [0, -1], [0, 0]], [[0, 0], [0, -1], [0, 0], [0, 1]], [[0, 0], [0,
-1], [1, -1], [1, 0]], [[0, 0], [0, 0], [-1, 0], [-1, 0]], [[0, 0], [0, 0], [-1, 1], [-1, 1]],
[[0, 0], [0, 0], [0, -1], [0, -1]], [[0, 0], [0, 0], [0, 0], [0, 0]], [[0, 0], [0, 0], [0, 1],
[0, 1]], [[0, 0], [0, 0], [1, -1], [1, -1]], [[0, 0], [0, 0], [1, 0], [1, 0]], [[0, 0], [0, 1], [
-1, 1], [-1, 0]], [[0, 0], [0, 1], [0, 0], [0, -1]], [[0, 0], [0, 1], [0, 1], [0, 0]], [[0, 0],
[0, 1], [1, 0], [1, -1]], [[0, 0], [0, 1], [1, 1], [1, 0]], [[0, 0], [1, -1], [0, -1], [-1, 0]],
[[0, 0], [1, -1], [0, 0], [-1, 1]], [[0, 0], [1, -1], [1, -1], [0, 0]], [[0, 0], [1, -1], [1,
0], [0, 1]], [[0, 0], [1, 0], [0, 0], [-1, 0]], [[0, 0], [1, 0], [0, 1], [-1, 1]], [[0, 0], [1,
0], [1, -1], [0, -1]], [[0, 0], [1, 0], [1, 0], [0, 0]], [[0, 0], [1, 0], [1, 1], [0, 1]], [[0,
0], [1, 1], [0, 1], [-1, 0]], [[0, 0], [1, 1], [1, 0], [0, -1]], [[0, 0], [1, 1], [1, 1], [0,
0]], [[0, 1], [0, 1], [-1, -1], [0, 1]], [[0, 1], [1, 1], [-1, -1], [-1, 1]], [[0, 1], [1, 1],
[0, -1], [0, 1]], [[1, -1], [0, -1], [-1, 1], [1, -1]], [[1, -1], [1, -1], [-1, 1], [0,
-1]], [[1, -1], [1, -1], [0, 1], [1, -1]], [[1, 0], [1, -1], [-1, -1], [1, 0]], [[1, 0], [1,
0], [-1, -1], [1, -1]], [[1, 0], [1, 0], [-1, 0], [1, 0]], [[1, 0], [1, 1], [-1, 0], [1, -1]],

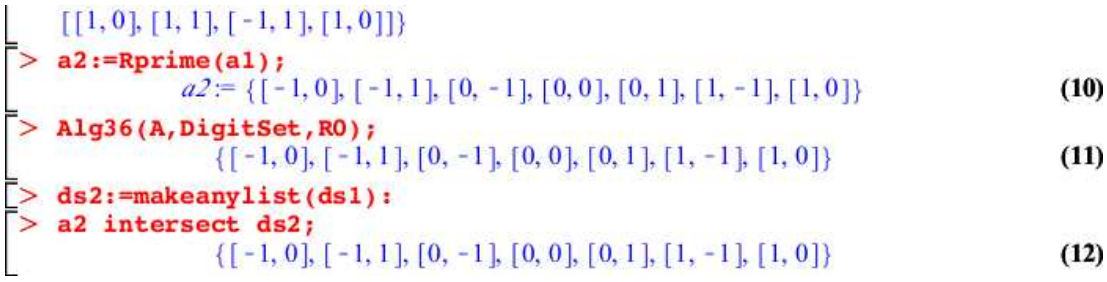


Figure A.2: Extended output for 3103

Appendix-Example 3:

```
> DigitSet:={<0,0>,<1,0>,<-1,0>,<2,1>,<3,1>,<1,1>,<-1,-1>,<-3,-1>,
  <-2,-1>};
```

$$DigitSet := \left\{\begin{bmatrix} -3 \\ -1 \end{bmatrix}, \begin{bmatrix} -2 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 3 \\ 1 \end{bmatrix}\right\} \quad (1)$$

```
> R0:={<0,0>,<1,0>,<-1,0>,<0,1>,<0,-1>};
```

$$R0 := \left\{\begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}\right\} \quad (2)$$

```
> A:=<<3,0>|<6,3>>;
```

$$A := \begin{bmatrix} 3 & 6 \\ 0 & 3 \end{bmatrix} \quad (3)$$

```
> ds1:=deltaDS(DigitSet):
> pf:=pairsfaces(A,DigitSet,R0);
```

$$pf := \left\{\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \end{bmatrix}\right\} \quad (4)$$

```
> Rn:=genRn(A,DigitSet,R0);
```

$$Rn := \left\{\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \end{bmatrix}\right\} \quad (5)$$

```
> GR:=makeall(Rn,A,DigitSet):
> GR1:=makelist(GR):
> RGR:=makelist(RedG(GR));
```

$$RGR := \{[[-1,0],[-3,-1],[-1,-1],[-1,0]],[[-1,0],[-3,-1],[-1,0],[-1,1]],[[-1,0],[-3,-1],[0,0],[0,1]],[[-1,0],[-2,-1],[0,0],[-1,1]],[[-1,0],[-2,-1],[1,0],[0,1]],[[-1,0],[-1,-1],[1,0],[-1,1]],[[-1,0],[-1,0],[1,0],[-1,0]],[[-1,0],[-1,0],[1,1],[-1,1]],[[-1,0],[-1,0],[2,1],[0,1]],[[-1,0],[0,0],[2,1],[-1,1]],[[-1,0],[0,0],[3,1],[0,1]],[[-1,0],[1,0],[3,1],[-1,1]],[[-1,0],[1,1],[3,1],[-1,0]],[[-1,1],[1,1],[-3,-1],[-1,1]],[[-1,1],[1,1],[-2,-1],[0,1]],[[-1,1],[2,1],[-2,-1],[-1,1]],[[-1,1],[2,1],[-1,-1],[0,1]],[[-1,1],[3,1],[-1,-1],[-1,1]],[[0,-1],[-3,-1],[3,1],[0,-1]],[[0,0],[-3,-1],[-3,-1],[0,0]],[[0,0],[-3,-1],[-2,-1],[1,0]],[[0,0],[-2,-1],[-3,-1],[-1,0]],[[0,0],[-2,-1],[-2,-1],[0,0]],[[0,0],[-2,-1],[-1,-1],[1,0]],[[0,0],[-1,-1],[-2,-1],[-1,0]],[[0,0],[-1,-1],[-1,-1],[0,0]],[[0,0],[-1,-1],[-1,0],[0,1]],[[0,0],[-1,0],[-1,-1],[0,-1]],[[0,0],[-1,0],[-1,0],[0,0]],[[0,0],[-1,0],[0,0],[1,0]],[[0,0],[0,0],[-1,0],[-1,0]],[[0,0],[0,0],[0,0],[0,0]],[[0,0],[0,0],[1,0],[1,0]],[[0,0],[1,0],[0,0],[-1,0]],[[0,0],[1,0],[1,0],[0,0]],[[0,0],[1,0],[1,1],[0,1]],[[0,0],[1,1],[1,0],[0,-1]],[[0,0],[1,1],[1,1],[0,0]],[[0,0],[1,1],[2,1],[1,0]],[[0,0],[2,1],[1,1],[-1,0]],[[0,0],[2,1],[2,1],[0,0]],[[0,0],[2,1],[3,1],[1,0]],[[0,0],[3,1],[2,1],[-1,0]],[[0,0], \quad (6)$$

```
[3, 1], [3, 1], [0, 0]], [[0, 1], [3, 1], [-3, -1], [0, 1]], [[1, -1], [-3, -1], [1, 1], [1,
-1]], [[1, -1], [-2, -1], [1, 1], [0, -1]], [[1, -1], [-2, -1], [2, 1], [1, -1]], [[1,
-1], [-1, -1], [2, 1], [0, -1]], [[1, -1], [-1, -1], [3, 1], [1, -1]], [[1, 0], [-1, -1], [
-3, -1], [1, 0]], [[1, 0], [-1, 0], [-3, -1], [1, -1]], [[1, 0], [0, 0], [-3, -1], [0, -1]],
[[1, 0], [0, 0], [-2, -1], [1, -1]], [[1, 0], [1, 0], [-2, -1], [0, -1]], [[1, 0], [1, 0], [
-1, -1], [1, -1]], [[1, 0], [1, 0], [-1, 0], [1, 0]], [[1, 0], [1, 1], [-1, 0], [1, -1]], [[1,
0], [2, 1], [-1, 0], [0, -1]], [[1, 0], [2, 1], [0, 0], [1, -1]], [[1, 0], [3, 1], [0, 0], [0,
-1]], [[1, 0], [3, 1], [1, 0], [1, -1]], [[1, 0], [3, 1], [1, 1], [1, 0]]}
> a:=alg36(GR1):
> a1:=makelist(a);
```

$$a1 := \{[[-2,0],[-3,-1],[1,0],[-2,1]],[[-2,0],[-1,0],[3,1],[-2,1]],[[-2,1],[1,1],[-1,-1],[-2,1]],[[-1,0],[-3,-1],[-2,-1],[-2,0]],[[-1,0],[-3,-1],[-1,-1],[-1,0]],[[-1,0],[-3,-1],[-1,0],[-1,1]],[[-1,0],[-3,-1],[0,0],[0,1]],[[-1,0],[-2,-1],[-1,-1],[-2,0]],[[-1,0],[-2,-1],[-1,0],[-2,1]],[[-1,0],[-2,-1],[0,0],[-1,1]],[[-1,0],[-2,-1],[1,0],[0,1]],[[-1,0],[-1,-1],[0,0],[-2,1]],[[-1,0],[-1,-1],[1,0],[-1,1]],[[-1,0],[-1,0],[0,0],[-2,0]],[[-1,0],[-1,0],[1,0],[-1,0]],[[-1,0],[-1,0],[1,1],[-1,1]],[[-1,0],[-1,0],[2,1],[0,1]],[[-1,0],[0,0],[1,0],[-2,0]],[[-1,0],[0,0],[1,1],[-2,1]],[[-1,0],[0,0],[2,1],[-1,1]],[[-1,0],[0,0],[3,1],[0,1]],[[-1,0],[1,0],[2,1],[-2,1]],[[-1,0],[1,0],[3,1],[-1,1]],[[-1,0],[1,1],[2,1],[-2,0]],[[-1,0],[1,1],[3,1],[-1,0]],[[-1,0],[2,1],[3,1],[-2,0]],[[-1,1],[1,1],[-3,-1],[-1,1]],[[-1,1],[1,1],[-2,-1],[0,1]],[[-1,1],[2,1],[-3,-1],[-2,1]],[[-1,1],[2,1],[-2,-1],[-1,1]],[[-1,1],[2,1],[-1,-1],[0,1]],[[-1,1],[3,1],[-2,-1],[-2,1]],[[-1,1],[3,1],[-1,-1],[-1,1]],[[0,-1],[-3,-1],[3,1],[0,-1]],[[0,0],[-3,-1],[-3,-1],[0,0]],[[0,0],[-3,-1],[-2,-1],[1,0]],[[0,0],[-3,-1],[-1,-1],[2,0]],[[0,0],[-2,-1],[-3,-1],[-1,0]],[[0,0],[-2,-1],[-2,-1],[0,0]],[[0,0],[-2,-1],[-1,-1],[1,0]],[[0,0],[-1,-1],[-3,-1],[-2,0]],[[0,0],[-1,-1],[-2,-1],[-1,0]],[[0,0],[-1,-1],[-1,-1],[0,0]],[[0,0],[-1,-1],[-1,0],[0,1]],[[0,0],[-1,0],[-1,-1],[0,-1]],[[0,0],[-1,0],[-1,0],[0,0]],[[0,0],[-1,0],[0,0],[1,0]],[[0,0],[-1,0],[1,0],[2,0]],[[0,0],[0,0],[-1,0],[-1,0]],[[0,0],[0,0],[0,0],[0,0]],[[0,0],[0,0],[1,0],[1,0]],[[0,0],[1,0],[-1,0],[-2,0]],[[0,0],[1,0],[0,0],[-1,0]],[[0,0],[1,0],[1,0],[0,0]],[[0,0],[1,0],[1,1],[0,1]],[[0,0],[1,1],[1,0],[0,-1]],[[0,0],[1,1],[1,1],[0,0]],[[0,0],[1,1],[2,1],[1,0]],[[0,0],[1,1],[3,1],[2,0]],[[0,0],[2,1],[1,1],[-1,0]],[[0,0],[2,1],[2,1],[0,0]],[[0,0],[2,1],[3,1],[1,0]],[[0,0],[3,1],[1,1],[-2,0]],[[0,0],[3,1],[2,1],[-1,0]],[[0,0],[3,1],[3,1],[0,0]],[[0,1],[3,1],[-3,-1],[0,1]],[[1,-1],[-3,-1],[1,1],[1,-1]],[[1,-1],[-3,-1],[2,1],[2,-1]],[[1,-1],[-2,-1],[1,1],[0,-1]],[[1,-1],[-2,-1],[2,1],[1,-1]],[[1,-1],[-2,-1],[3,\quad (7)$$

```
    1], [2, -1]], [[1, -1], [-1, -1], [2, 1], [0, -1]], [[1, -1], [-1, -1], [3, 1], [1, -1]],
    [[1, 0], [-2, -1], [-3, -1], [2, 0]], [[1, 0], [-1, -1], [-3, -1], [1, 0]], [[1, 0], [-1,
    -1], [-2, -1], [2, 0]], [[1, 0], [-1, 0], [-3, -1], [1, -1]], [[1, 0], [-1, 0], [-2, -1],
    [2, -1]], [[1, 0], [0, 0], [-3, -1], [0, -1]], [[1, 0], [0, 0], [-2, -1], [1, -1]], [[1, 0],
    [0, 0], [-1, -1], [2, -1]], [[1, 0], [0, 0], [-1, 0], [2, 0]], [[1, 0], [1, 0], [-2, -1], [0,
    -1]], [[1, 0], [1, 0], [-1, -1], [1, -1]], [[1, 0], [1, 0], [-1, 0], [1, 0]], [[1, 0], [1, 0],
    [0, 0], [2, 0]], [[1, 0], [1, 1], [-1, 0], [1, -1]], [[1, 0], [1, 1], [0, 0], [2, -1]], [[1, 0],
    [2, 1], [-1, 0], [0, -1]], [[1, 0], [2, 1], [0, 0], [1, -1]], [[1, 0], [2, 1], [1, 0], [2, -1]],
    [[1, 0], [2, 1], [1, 1], [2, 0]], [[1, 0], [3, 1], [0, 0], [0, -1]], [[1, 0], [3, 1], [1, 0], [1,
    -1]], [[1, 0], [3, 1], [1, 1], [1, 0]], [[1, 0], [3, 1], [2, 1], [2, 0]], [[2, -1], [-1, -1],
    [1, 1], [2, -1]], [[2, 0], [1, 0], [-3, -1], [2, -1]], [[2, 0], [3, 1], [-1, 0], [2, -1]]}
> a2:=Rprime(a1);
a2 := {[-2, 0], [-2, 1], [-1, 0], [-1, 1], [0, -1], [0, 0], [0, 1], [1, -1], [1, 0], [2, -1], [2,      (8)
    0]}
> Alg36(A,DigitSet,R0);
  {[-2, 0], [-2, 1], [-1, 0], [-1, 1], [0, -1], [0, 0], [0, 1], [1, -1], [1, 0], [2, -1], [2, 0]}      (9)
> ds2:=makeanylist(ds1):
> a2 intersect ds2;
                {[-2, 0], [-1, 0], [0, -1], [0, 0], [0, 1], [1, 0], [2, 0]}                           (10)
```

Figure A.3: Extended output for 3603

Appendix-Example 4:

```
> DigitSet:={<0,0>,<1,0>,<-1,0>,<2,1>,<3,1>,<1,1>,<-1,-1>,<-3,-1>,
  <-2,-1>};
```

$$DigitSet := \left\{\begin{bmatrix} -3 \\ -1 \end{bmatrix}, \begin{bmatrix} -2 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 3 \\ 1 \end{bmatrix}\right\} \quad (1)$$

```
> R0:={<0,0>,<1,0>,<-1,0>,<0,1>,<0,-1>};
```

$$R0 := \left\{\begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}\right\} \quad (2)$$

```
> A:=<<3,0>|<7,3>>;
```

$$A := \begin{bmatrix} 3 & 7 \\ 0 & 3 \end{bmatrix} \quad (3)$$

```
> ds1:=deltaDS(DigitSet):
> pf:=pairsfaces(A,DigitSet,R0);
```

$$pf := \left\{\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 1 \end{bmatrix}\right\} \quad (4)$$

```
> Rn:=genRn(A,DigitSet,R0);
```

$$Rn := \left\{\begin{bmatrix} -1 \\ 1 \end{bmatrix}, \begin{bmatrix} -2 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} -2 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ -1 \end{bmatrix}, \begin{bmatrix} 2 \\ 0 \end{bmatrix}\right\} \quad (5)$$

```
> GR:=makeall(Rn,A,DigitSet):
> GR1:=makelist(GR):
> RGR:=makelist(RedG(GR));
```

$$RGR := \{[[-2,0],[-3,-1],[1,0],[-2,1]],[[-2,0],[-1,0],[3,1],[-2,1]],[[-2,1],$$
$$[1,1],[-2,-1],[-2,1]],[[-2,1],[1,1],[-1,-1],[-1,1]],[[-2,1],[2,1],[-1,$$
$$-1],[-2,1]],[[-1,0],[-3,-1],[-2,-1],[-2,0]],[[-1,0],[-3,-1],[-1,-1],[$$
$$-1,0]],[[-1,0],[-3,-1],[-1,0],[-1,1]],[[-1,0],[-2,-1],[-1,-1],[-2,0]],[[$$
$$-1,0],[-2,-1],[-1,0],[-2,1]],[[-1,0],[-2,-1],[0,0],[-1,1]],[[-1,0],[-1,$$
$$-1],[0,0],[-2,1]],[[-1,0],[-1,-1],[1,0],[-1,1]],[[-1,0],[-1,0],[0,0],[-2,$$
$$0]],[[-1,0],[-1,0],[1,0],[-1,0]],[[-1,0],[-1,0],[1,1],[-1,1]],[[-1,0],[0,$$
$$0],[1,0],[-2,0]],[[-1,0],[0,0],[1,1],[-2,1]],[[-1,0],[0,0],[2,1],[-1,1]],$$
$$[[-1,0],[1,0],[2,1],[-2,1]],[[-1,0],[1,0],[3,1],[-1,1]],[[-1,0],[1,1],[2,$$
$$1],[-2,0]],[[-1,0],[1,1],[3,1],[-1,0]],[[-1,0],[2,1],[3,1],[-2,0]],[[-1,$$
$$1],[2,1],[-3,-1],[-1,1]],[[-1,1],[3,1],[-3,-1],[-2,1]],[[-1,1],[3,1],[-2,$$
$$-1],[-1,1]],[[0,0],[-3,-1],[-3,-1],[0,0]],[[0,0],[-3,-1],[-2,-1],[1,0]],$$
$$[[0,0],[-3,-1],[-1,-1],[2,0]],[[0,0],[-2,-1],[-3,-1],[-1,0]],[[0,0],[-2,$$
$$-1],[-2,-1],[0,0]],[[0,0],[-2,-1],[-1,-1],[1,0]],[[0,0],[-1,-1],[-3,-1],$$
$$[-2,0]],[[0,0],[-1,-1],[-2,-1],[-1,0]],[[0,0],[-1,-1],[-1,-1],[0,0]],$$
$$[[0,0],[-1,0],[-1,0],[0,0]],[[0,0],[-1,0],[0,0],[1,0]],[[0,0],[-1,0],[1,$$
$$0],[2,0]],[[0,0],[0,0],[-1,0],[-1,0]],[[0,0],[0,0],[0,0],[0,0]],[[0,0],[0,$$

(6)

```
0], [1, 0], [1, 0]], [[0, 0], [1, 0], [-1, 0], [-2, 0]], [[0, 0], [1, 0], [0, 0], [-1, 0]], [[0,
0], [1, 0], [1, 0], [0, 0]], [[0, 0], [1, 1], [1, 1], [0, 0]], [[0, 0], [1, 1], [2, 1], [1, 0]],
[[0, 0], [1, 1], [3, 1], [2, 0]], [[0, 0], [2, 1], [1, 1], [-1, 0]], [[0, 0], [2, 1], [2, 1], [0,
0]], [[0, 0], [2, 1], [3, 1], [1, 0]], [[0, 0], [3, 1], [1, 1], [-2, 0]], [[0, 0], [3, 1], [2, 1],
[-1, 0]], [[0, 0], [3, 1], [3, 1], [0, 0]], [[1, -1], [-3, -1], [2, 1], [1, -1]], [[1, -1], [
-3, -1], [3, 1], [2, -1]], [[1, -1], [-2, -1], [3, 1], [1, -1]], [[1, 0], [-2, -1], [-3,
-1], [2, 0]], [[1, 0], [-1, -1], [-3, -1], [1, 0]], [[1, 0], [-1, -1], [-2, -1], [2, 0]],
[[1, 0], [-1, 0], [-3, -1], [1, -1]], [[1, 0], [-1, 0], [-2, -1], [2, -1]], [[1, 0], [0, 0],
[-2, -1], [1, -1]], [[1, 0], [0, 0], [-1, -1], [2, -1]], [[1, 0], [0, 0], [-1, 0], [2, 0]],
[[1, 0], [1, 0], [-1, -1], [1, -1]], [[1, 0], [1, 0], [-1, 0], [1, 0]], [[1, 0], [1, 0], [0,
0], [2, 0]], [[1, 0], [1, 1], [-1, 0], [1, -1]], [[1, 0], [1, 1], [0, 0], [2, -1]], [[1, 0], [2,
1], [0, 0], [1, -1]], [[1, 0], [2, 1], [1, 0], [2, -1]], [[1, 0], [2, 1], [1, 1], [2, 0]], [[1,
0], [3, 1], [1, 0], [1, -1]], [[1, 0], [3, 1], [1, 1], [1, 0]], [[1, 0], [3, 1], [2, 1], [2, 0]],
[[2, -1], [-2, -1], [1, 1], [2, -1]], [[2, -1], [-1, -1], [1, 1], [1, -1]], [[2, -1], [-1,
-1], [2, 1], [2, -1]], [[2, 0], [1, 0], [-3, -1], [2, -1]], [[2, 0], [3, 1], [-1, 0], [2,
-1]]}
> a:=alg36(GR1):
> a1:=makelist(a);
```

$$a1 := \{[[-2,0],[-3,-1],[1,0],[-2,1]],[[-2,0],[-1,0],[3,1],[-2,1]],[[-2,1],[1,1],[-2,-1],[-2,1]],[[-2,1],[1,1],[-1,-1],[-1,1]],[[-2,1],[2,1],[-1,-1],[-2,1]],[[-1,0],[-3,-1],[-2,-1],[-2,0]],[[-1,0],[-3,-1],[-1,-1],[-1,0]],[[-1,0],[-3,-1],[-1,0],[-1,1]],[[-1,0],[-2,-1],[-1,-1],[-2,0]],[[-1,0],[-2,-1],[-1,0],[-2,1]],[[-1,0],[-2,-1],[0,0],[-1,1]],[[-1,0],[-1,-1],[0,0],[-2,1]],[[-1,0],[-1,-1],[1,0],[-1,1]],[[-1,0],[-1,0],[0,0],[-2,0]],[[-1,0],[-1,0],[1,0],[-1,0]],[[-1,0],[-1,0],[1,1],[-1,1]],[[-1,0],[0,0],[1,0],[-2,0]],[[-1,0],[0,0],[1,1],[-2,1]],[[-1,0],[0,0],[2,1],[-1,1]],[[-1,0],[1,0],[2,1],[-2,1]],[[-1,0],[1,0],[3,1],[-1,1]],[[-1,0],[1,1],[2,1],[-2,0]],[[-1,0],[1,1],[3,1],[-1,0]],[[-1,0],[2,1],[3,1],[-2,0]],[[-1,1],[2,1],[-3,-1],[-1,1]],[[-1,1],[3,1],[-3,-1],[-2,1]],[[-1,1],[3,1],[-2,-1],[-1,1]],[[0,0],[-3,-1],[-3,-1],[0,0]],[[0,0],[-3,-1],[-2,-1],[1,0]],[[0,0],[-3,-1],[-1,-1],[2,0]],[[0,0],[-2,-1],[-3,-1],[-1,0]],[[0,0],[-2,-1],[-2,-1],[0,0]],[[0,0],[-2,-1],[-1,-1],[1,0]],[[0,0],[-1,-1],[-3,-1],[-2,0]],[[0,0],[-1,-1],[-2,-1],[-1,0]],[[0,0],[-1,-1],[-1,-1],[0,0]],[[0,0],[-1,0],[-1,0],[0,0]],[[0,0],[-1,0],[0,0],[1,0]],[[0,0],[-1,0],[1,0],[2,0]],[[0,0],[0,0],[-1,0],[-1,0]],[[0,0],[0,0],[0,0],[0,0]],[[0,0],[0,0],[1,0],[1,0]],[[0,0],[1,0],[-1,0],[-2,0]],[[0,0],[1,0],[0,0],[-1,0]],[[0,0],[1,0],[1,0],[0,0]],[[0,0],[1,1],[1,1],[0,0]],[[0,0],[1,1],[2,1],[1,0]],[[0,0],[1,1],[3,1],[2,0]],[[0,0],[2,1],[1,1],[-1,0]],[[0,0],[2,1],[2,1],[0,0]],[[0,0], \quad (7)$$

```
[2, 1], [3, 1], [1, 0]], [[0, 0], [3, 1], [1, 1], [-2, 0]], [[0, 0], [3, 1], [2, 1], [-1, 0]],
[[0, 0], [3, 1], [3, 1], [0, 0]], [[1, -1], [-3, -1], [2, 1], [1, -1]], [[1, -1], [-3, -1],
[3, 1], [2, -1]], [[1, -1], [-2, -1], [3, 1], [1, -1]], [[1, 0], [-2, -1], [-3, -1], [2,
0]], [[1, 0], [-1, -1], [-3, -1], [1, 0]], [[1, 0], [-1, -1], [-2, -1], [2, 0]], [[1, 0], [
-1, 0], [-3, -1], [1, -1]], [[1, 0], [-1, 0], [-2, -1], [2, -1]], [[1, 0], [0, 0], [-2,
-1], [1, -1]], [[1, 0], [0, 0], [-1, -1], [2, -1]], [[1, 0], [0, 0], [-1, 0], [2, 0]], [[1,
0], [1, 0], [-1, -1], [1, -1]], [[1, 0], [1, 0], [-1, 0], [1, 0]], [[1, 0], [1, 0], [0, 0], [2,
0]], [[1, 0], [1, 1], [-1, 0], [1, -1]], [[1, 0], [1, 1], [0, 0], [2, -1]], [[1, 0], [2, 1], [0,
0], [1, -1]], [[1, 0], [2, 1], [1, 0], [2, -1]], [[1, 0], [2, 1], [1, 1], [2, 0]], [[1, 0], [3,
1], [1, 0], [1, -1]], [[1, 0], [3, 1], [1, 1], [1, 0]], [[1, 0], [3, 1], [2, 1], [2, 0]], [[2,
-1], [-2, -1], [1, 1], [2, -1]], [[2, -1], [-1, -1], [1, 1], [1, -1]], [[2, -1], [-1, -1],
[2, 1], [2, -1]], [[2, 0], [1, 0], [-3, -1], [2, -1]], [[2, 0], [3, 1], [-1, 0], [2, -1]]}
> a2:=Rprime(a1);
   a2 := {[-2, 0], [-2, 1], [-1, 0], [-1, 1], [0, 0], [1, -1], [1, 0], [2, -1], [2, 0]}      (8)
> Alg36(A,DigitSet,R0);
      {[-2, 0], [-2, 1], [-1, 0], [-1, 1], [0, 0], [1, -1], [1, 0], [2, -1], [2, 0]}        (9)
> ds2:=makeanylist(ds1):
> a2 intersect ds2;
                    {[-2, 0], [-1, 0], [0, 0], [1, 0], [2, 0]}                              (10)
```

Figure A.4: Extended output for 3703

B Sage Procedures

In all of our examples from Sage, to generate the level sets D_n, we used the following iteratively defined expanding function.

```
def Dn(k):
    ans = []
    for d in Tuples(D, k):
        s = sum(A^n*d[n] for n in range(k))
        ans.append(s)
    return ans
```

To generate the sets T_n, we used the corresponding contraction.

```
def Dn(k):
    ans = []
    for d in Tuples(D, k):
        s = sum(A^(-(n+1)*d[n] for n in range(k))
        ans.append(s)
    return ans
```

Using the built-in function `max` in Sage, we are able to find the largest (x, y) value in D_n, and add one to the y value to examine the overlap of the horizontal rows of D_{n+1} to D_n.

```
max(D(2))
```

```
v=Dn(3)
v.sort(lambda x,y: cmp(reversed(x),reversed(y)))
```

```
w=[x for x in v if x[1]==11]
w
```

Description of Examples in Sage	
Name	Description
3603slider	shows the 'slider' application available using the Sage interact function
3703slider	shows the 'slider' application available using the Sage interact function
1D Sage Procedure	displays a basic digit set as both a tree and a nodal graph.
Twin Dragon	a fundamental two dimensional example exhibiting a fractal boundary, and an unrepresentable point.
3003	displays D_n and T_n for $\lambda = 3$, $k = 0$
3103	displays D_n and T_n for $\lambda = 3$, $k = 1$
4104	displays D_n and T_n for $\lambda = 4$, $k = 1$
Fractal Cloud	a three-dimensional tile with a fractal boundary.
Jordan Form-31-3D	a visualization of T_4, a three dimensional Jordan Form matrix, where $\lambda = 3$, $k = 1$.
Jordan Form-41-3D	a visualization of T_3, a three dimensional Jordan Form matrix, where $\lambda = 4$, $k = 1$.

Table B.1: Sage Examples

The following table outlines the examples in this Appendix.

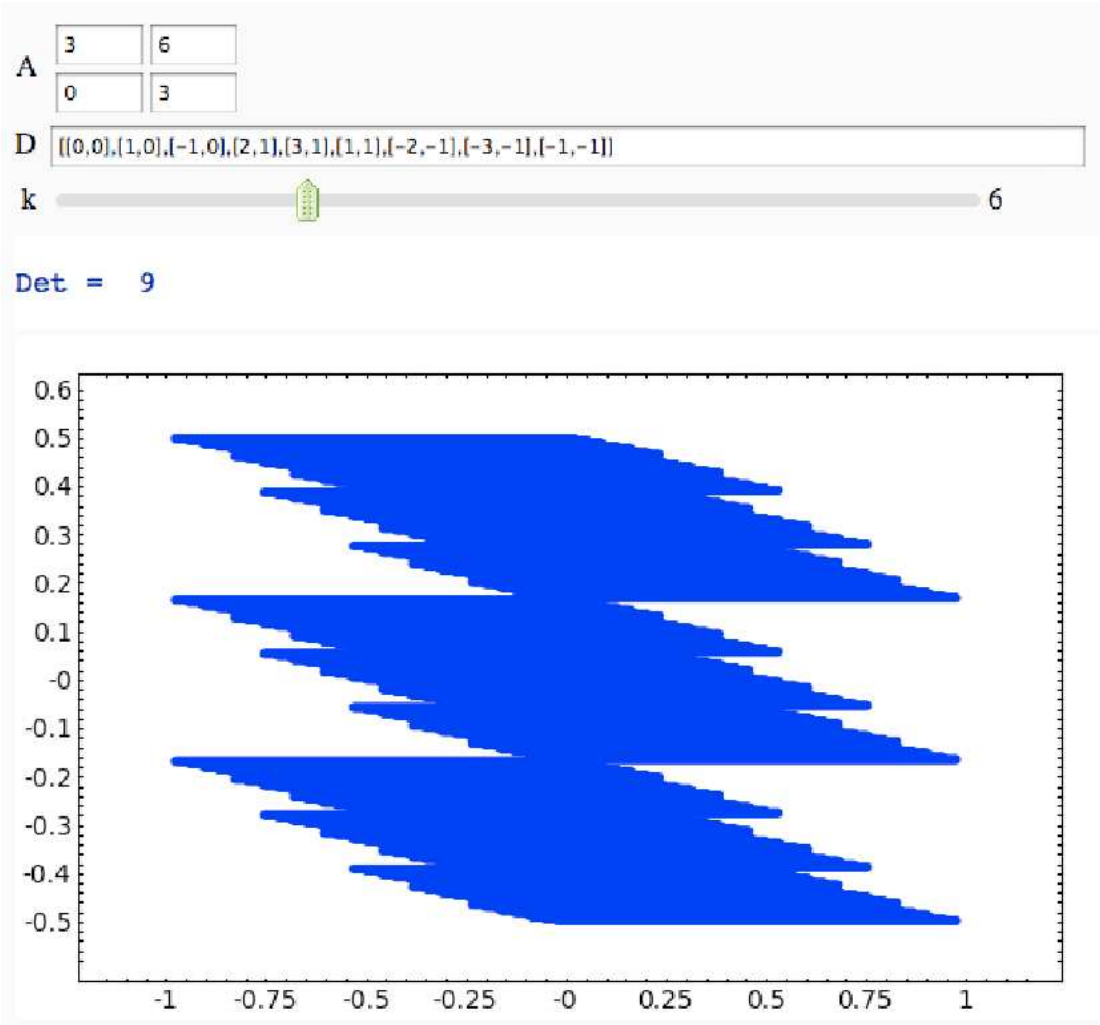


Figure B.1: 3603slider

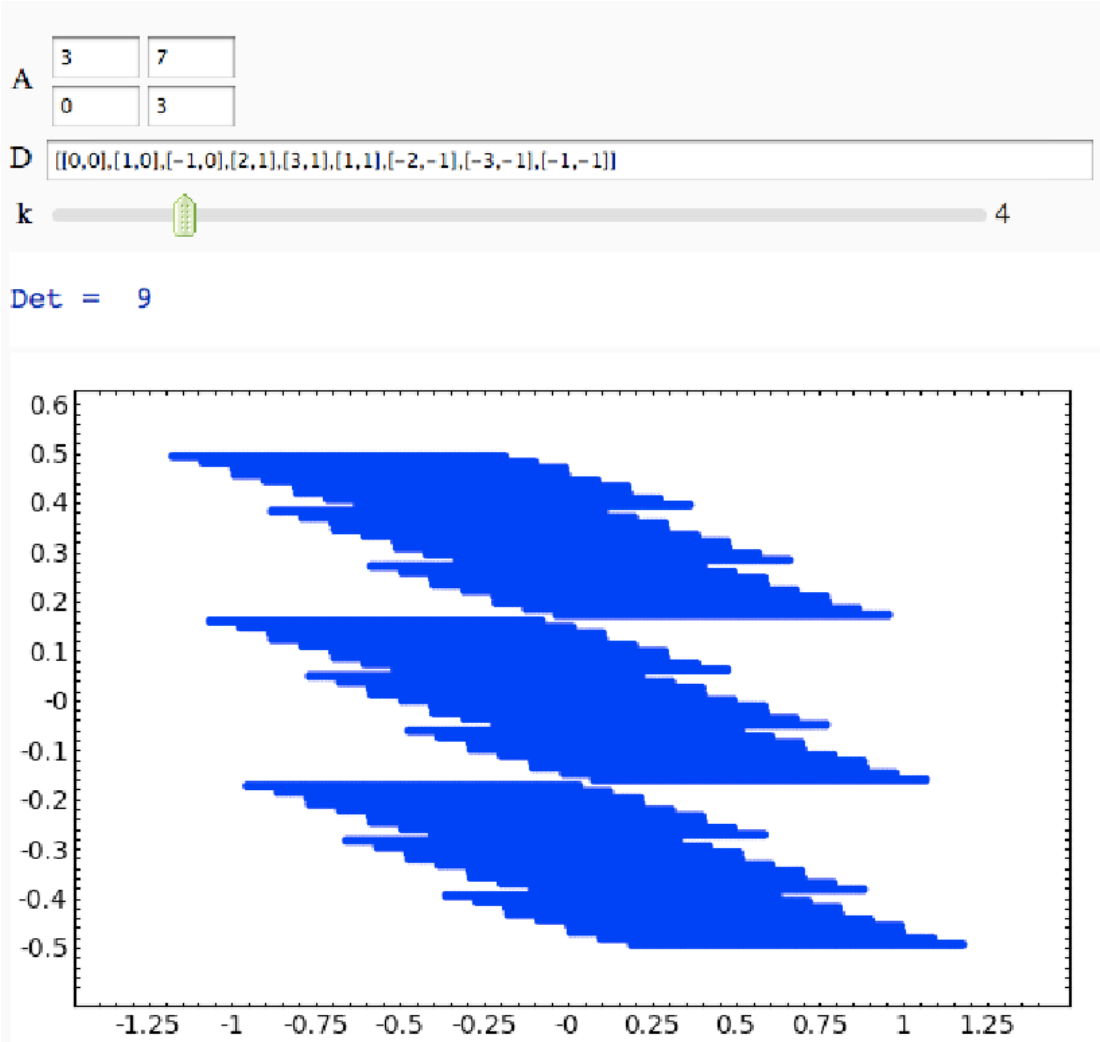
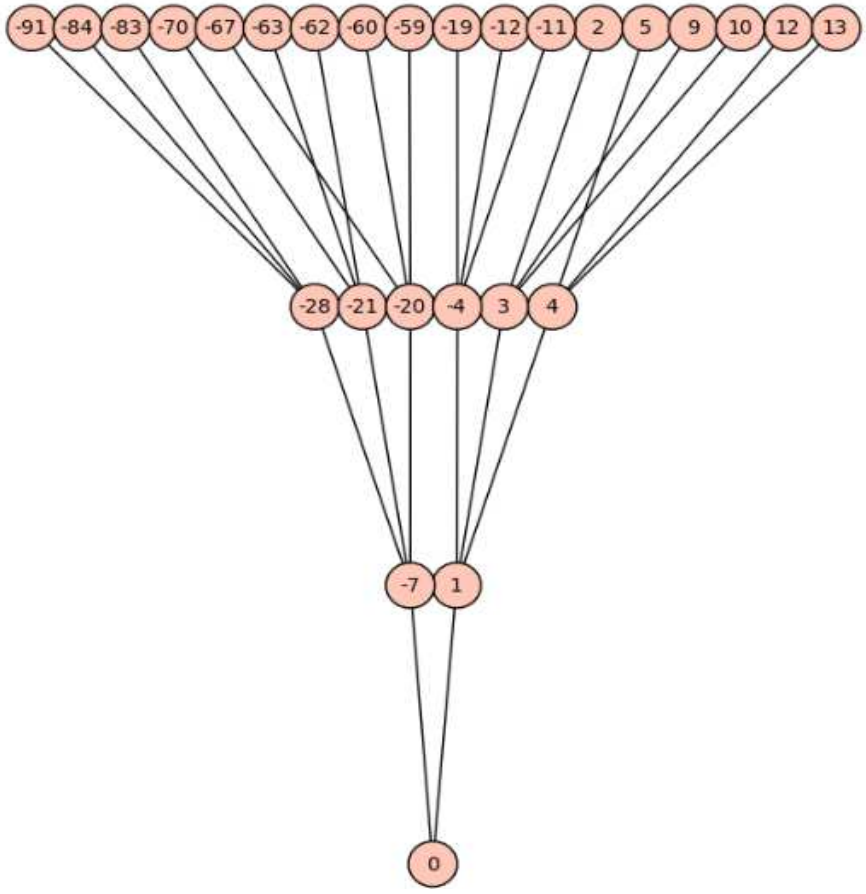


Figure B.2: 3703slider

1D Sage Procedure

```
D = Graph()
D.add_vertex(0)
visited = []
digits = [0, 1, -7]
```

```
D = Graph()
D.add_vertex(0)
visited = []
digits = [0, 1, -7]
beta = 3

def recursive_tree_generate(D,digits,beta,d1,depth=0,max_depth=3):
   if depth >= max_depth:
       return
   for d2 in digits:
       D.add_edge(d1, beta*d1 + d2, d2)
       recursive_tree_generate(D,digits,beta,beta*d1 + d2,depth+1,max_depth)

recursive_tree_generate(D,digits,beta,0,0)

D.plot(save_pos=True,layout='tree',edge_labels=False,vertex_size=450,
tree_root=0,tree_orientation='up').show(figsize=[8])
```



```
D.plot3d(edge_size=0.001, vertex_size=0.01).show(figsize=10)
```

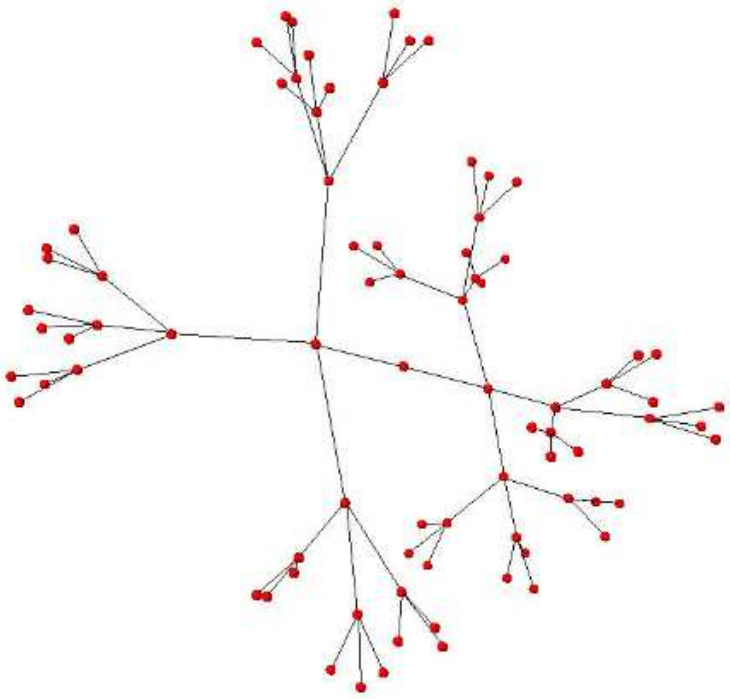


Figure B.3: 1D Sage Procedure

Twin Dragon

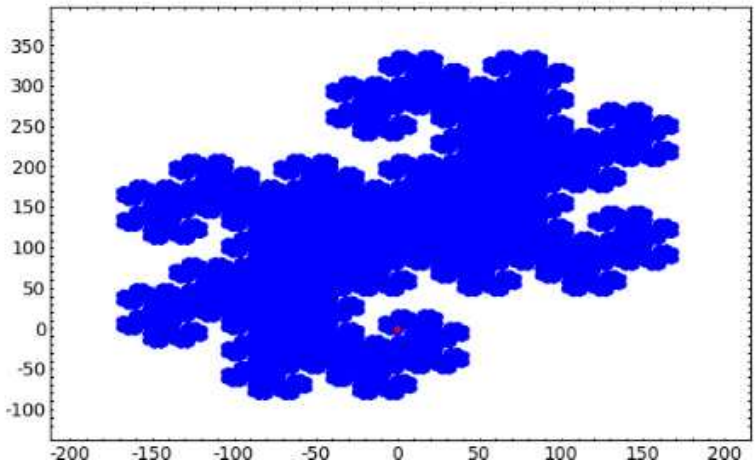
```
A = matrix([[1,1],[-1,1]])
D = [vector([0,0]), vector([1,0])]

@interact
def f(A = matrix([[1,1],[-1,1]]), D = '[[0,0],[1,0]]', k=(3..17)):
    print "Det = ", A.det()
    D = matrix(eval(D)).rows()
    def Dn(k):
        ans = []
        for d in Tuples(D, k):
            s = sum(A^(n)*d[n] for n in range(k))
            ans.append(s)
        return ans

    G = points([v.list() for v in Dn(k)])
    G+= point((0,-1), rgbcolor= (1,0,0))
    show(G, frame=True, axes=False)
```

A	1	1
	-1	1

D [[0,0],[1,0]]

k

Det = 2



```
A = matrix([[1,1],[-1,1]])
D = [vector([0,0]), vector([1,0])]

@interact
def f(A = matrix([[1,1],[-1,1]]), D = '[[0,0],[1,0]]', k=(3..17)):
    print "Det = ", A.det()
    D = matrix(eval(D)).rows()
    def Dn(k):
        ans = []
        for d in Tuples(D, k):
            s = sum(A^(-(n+1))*d[n] for n in range(k))
            ans.append(s)
        return ans

    G = points([v.list() for v in Dn(k)])
    show(G, frame=True, axes=False)
```

A: 1 1 / -1 1

D: [[0,0],[1,0]]

k

Det = 2

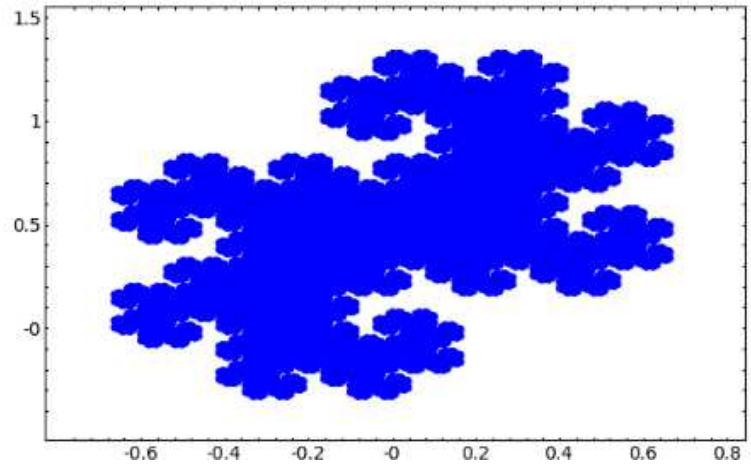


Figure B.4: Twin Dragon

3003

```
A = matrix([[3,0],[0,3]])
D = [vector([0,0]),vector([1,0]),vector([-1,0]),vector([1,1]),vector([-1,1]),
vector([0,1]),vector([1,-1]),vector([-1,-1]),vector([0,-1])]

@interact
def f(A = matrix([[3,0],[0,3]]), D = '[[0,0],[1,0],[-1,0],[1,1],[-1,1],[0,1],[0,-1],
[-1,-1],[1,-1]]', k=(3..17)):
    print "Det = ", A.det()
    D = matrix(eval(D)).rows()
    def Dn(k):
        ans = []
        for d in Tuples(D, k):
            s = sum(A^(-(n+1))*d[n] for n in range(k))
            ans.append(s)
        return ans

    G = points([v.list() for v in Dn(k)])

    show(G, frame=True, axes=False)
```

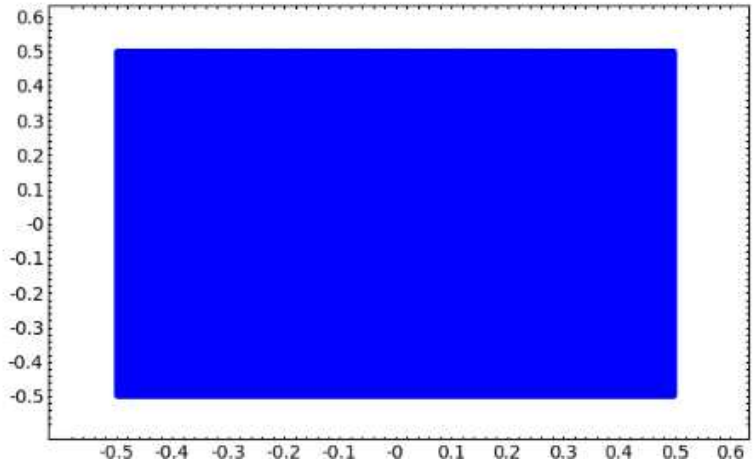
A

3	0
0	3

D [[0,0],[1,0],[-1,0],[1,1],[-1,1],[0,1],[0,-1],[-1,-1],[1,-1]]

k

```
Det =  9
```



```
A = matrix([[3,0],[0,3]])
D = [vector([0,0]),vector([1,0]),vector([-1,0]),vector([1,1]),vector([-1,1]),
vector([0,1]),vector([1,-1]),vector([-1,-1]),vector([0,-1])]

@interact
def f(A = matrix([[3,0],[0,3]]), D = '[[0,0],[1,0],[-1,0],[1,1],[-1,1],[0,1],[0,-1],
[-1,-1],[1,-1]]', k=(3..17)):
    print "Det = ", A.det()
    D = matrix(eval(D)).rows()
    def Dn(k):
        ans = []
        for d in Tuples(D, k):
            s = sum(A^(n)*d[n] for n in range(k))
            ans.append(s)


    G = points([v.list() for v in Dn(k)])

    show(G, frame=True, axes=False)
```

A

3	0
0	3

D [[0,0],[1,0],[-1,0],[1,1],[-1,1],[0,1],[0,-1],[-1,-1],[1,-1]]

k

```
Det =  9
```

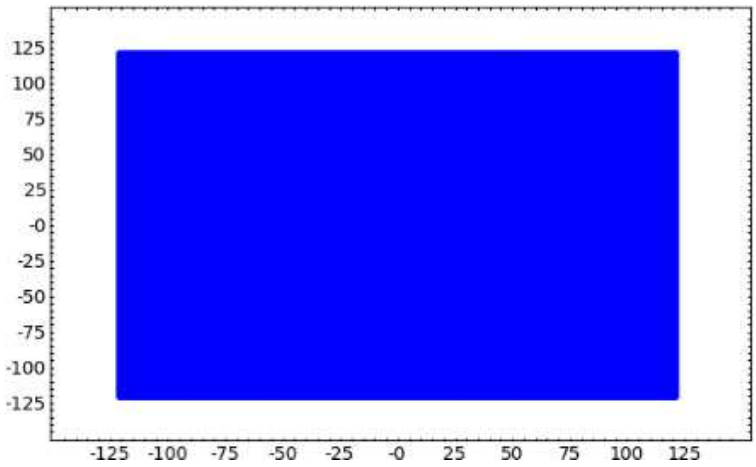


Figure B.5: 3003

3103

```
A = matrix([[3,1],[0,3]])
D = [vector([0,0]),vector([1,0]),vector([-1,0]),vector([1,1]),vector([-1,1]),
vector([0,1]),vector([1,-1]),vector([-1,-1]),vector([0,-1])]

@interact
def f(A = matrix([[3,1],[0,3]]), D = '[[0,0],[1,0],[-1,0],[1,1],[-1,1],[0,1],[0,-1],
[-1,-1],[1,-1]]', k=(3..17)):
    print "Det = ", A.det()
    D = matrix(eval(D)).rows()
    def Dn(k):
        ans = []
        for d in Tuples(D, k):
            s = sum(A^(n)*d[n] for n in range(k))
            ans.append(s)
        return ans

    G = points([v.list() for v in Dn(k)])

    show(G, frame=True, axes=False)
```

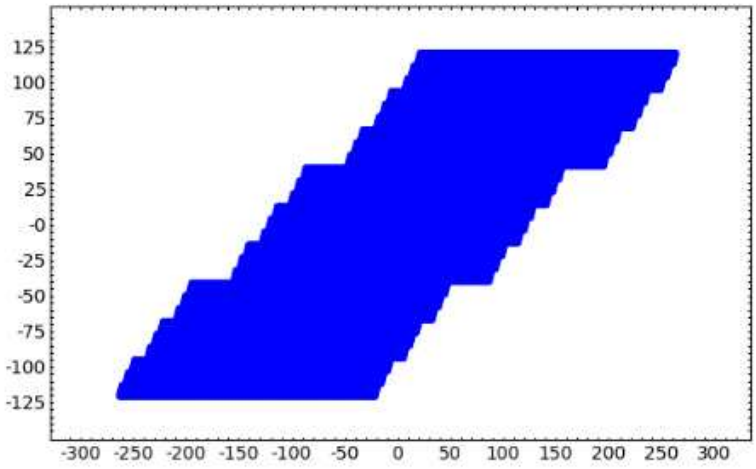
A

3	1
0	3

D [[0,0],[1,0],[-1,0],[1,1],[-1,1],[0,1],[0,-1],[-1,-1],[1,-1]]

k

```
Det =  9
```



```
A = matrix([[3,1],[0,3]])
D = [vector([0,0]),vector([1,0]),vector([-1,0]),vector([1,1]),vector([-1,1]),
vector([0,1]),vector([1,-1]),vector([-1,-1]),vector([0,-1])]

@interact
def f(A = matrix([[3,1],[0,3]]), D = '[[0,0],[1,0],[-1,0],[1,1],[-1,1],[0,1],[0,-1],
[-1,-1],[1,-1]]', k=(3..17)):
    print "Det = ", A.det()
    D = matrix(eval(D)).rows()
    def Dn(k):
        ans = []
        for d in Tuples(D, k):
            s = sum(A^(-(n+1))*d[n] for n in range(k))
            ans.append(s)

        return ans

    G = points([v.list() for v in Dn(k)])

    show(G, frame=True, axes=False)
```

A	3	1
	0	3

D [[0,0],[1,0],[-1,0],[1,1],[-1,1],[0,1],[0,-1],[-1,-1],[1,-1]]

k

```
Det =  9
```

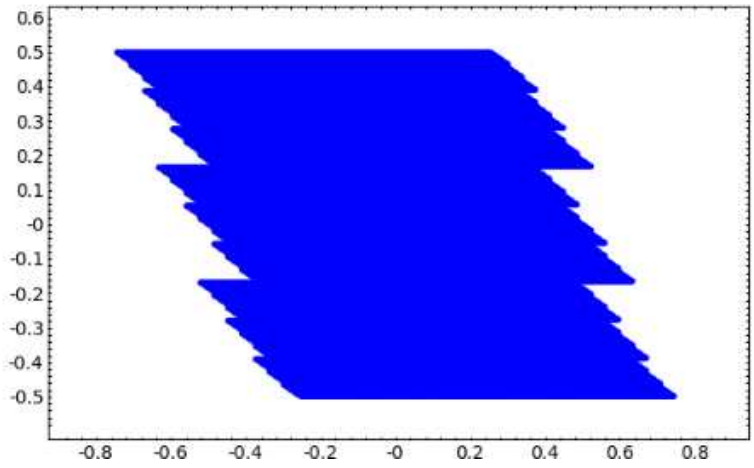


Figure B.6: 3103

4104

```
A = matrix([[4,1],[0,4]])
D = [vector([-1, -1]), vector([-1, 0]), vector([-1, 1]), vector([-1, 2]),
vector([0, -1]), vector([0, 0]), vector([0, 1]), vector([0, 2]),
vector([1, -1]), vector([1, 0]), vector([1, 1]), vector([1, 2]),
vector([2, -1]), vector([2, 0]), vector([2, 1]), vector([2, 2])]
```

```
def Dn(k):
    ans = []
    for d in Tuples(D, k):
        s = sum(A^n*d[n] for n in range(k))
        ans.append(s)
    return ans
```

```
Dn(1)
```

```
[(-1, -1), (-1, 0), (-1, 1), (-1, 2), (0, -1), (0, 0), (0, 1), (0,
2), (1, -1), (1, 0), (1, 1), (1, 2), (2, -1), (2, 0), (2, 1), (2,
2)]
```

```
Dn(2)
```

```
[(-6, -5), (-6, -4), (-6, -3), (-6, -2), (-5, -5), (-5, -4), (-5,
-3), (-5, -2), (-4, -5), (-4, -4), (-4, -3), (-4, -2), (-3, -5),
(-3, -4), (-3, -3), (-3, -2), (-5, -1), (-5, 0), (-5, 1), (-5, 2),
(-4, -1), (-4, 0), (-4, 1), (-4, 2), (-3, -1), (-3, 0), (-3, 1),
(-3, 2), (-2, -1), (-2, 0), (-2, 1), (-2, 2), (-4, 3), (-4, 4), (-4,
5), (-4, 6), (-3, 3), (-3, 4), (-3, 5), (-3, 6), (-2, 3), (-2, 4),
(-2, 5), (-2, 6), (-1, 3), (-1, 4), (-1, 5), (-1, 6), (-3, 7), (-3,
8), (-3, 9), (-3, 10), (-2, 7), (-2, 8), (-2, 9), (-2, 10), (-1, 7),
(-1, 8), (-1, 9), (-1, 10), (0, 7), (0, 8), (0, 9), (0, 10), (-2,
-5), (-2, -4), (-2, -3), (-2, -2), (-1, -5), (-1, -4), (-1, -3),
(-1, -2), (0, -5), (0, -4), (0, -3), (0, -2), (1, -5), (1, -4), (1,
-3), (1, -2), (-1, -1), (-1, 0), (-1, 1), (-1, 2), (0, -1), (0, 0),
(0, 1), (0, 2), (1, -1), (1, 0), (1, 1), (1, 2), (2, -1), (2, 0),
(2, 1), (2, 2), (0, 3), (0, 4), (0, 5), (0, 6), (1, 3), (1, 4), (1,
5), (1, 6), (2, 3), (2, 4), (2, 5), (2, 6), (3, 3), (3, 4), (3, 5),
(3, 6), (1, 7), (1, 8), (1, 9), (1, 10), (2, 7), (2, 8), (2, 9), (2,
10), (3, 7), (3, 8), (3, 9), (3, 10), (4, 7), (4, 8), (4, 9), (4,
10), (2, -5), (2, -4), (2, -3), (2, -2), (3, -5), (3, -4), (3, -3),
(3, -2), (4, -5), (4, -4), (4, -3), (4, -2), (5, -5), (5, -4), (5,
-3), (5, -2), (3, -1), (3, 0), (3, 1), (3, 2), (4, -1), (4, 0), (4,
1), (4, 2), (5, -1), (5, 0), (5, 1), (5, 2), (6, -1), (6, 0), (6,
1), (6, 2), (4, 3), (4, 4), (4, 5), (4, 6), (5, 3), (5, 4), (5, 5),
(5, 6), (6, 3), (6, 4), (6, 5), (6, 6), (7, 3), (7, 4), (7, 5), (7,
6), (5, 7), (5, 8), (5, 9), (5, 10), (6, 7), (6, 8), (6, 9), (6,
10), (7, 7), (7, 8), (7, 9), (7, 10), (8, 7), (8, 8), (8, 9), (8,
10), (6, -5), (6, -4), (6, -3), (6, -2), (7, -5), (7, -4), (7, -3),
(7, -2), (8, -5), (8, -4), (8, -3), (8, -2), (9, -5), (9, -4), (9,
-3), (9, -2), (7, -1), (7, 0), (7, 1), (7, 2), (8, -1), (8, 0), (8,
1), (8, 2), (9, -1), (9, 0), (9, 1), (9, 2), (10, -1), (10, 0), (10,
1), (10, 2), (8, 3), (8, 4), (8, 5), (8, 6), (9, 3), (9, 4), (9, 5),
(9, 6), (10, 3), (10, 4), (10, 5), (10, 6), (11, 3), (11, 4), (11,
5), (11, 6), (9, 7), (9, 8), (9, 9), (9, 10), (10, 7), (10, 8), (10,
9), (10, 10), (11, 7), (11, 8), (11, 9), (11, 10), (12, 7), (12, 8),
(12, 9), (12, 10)]
```

```
max(Dn(2))
```

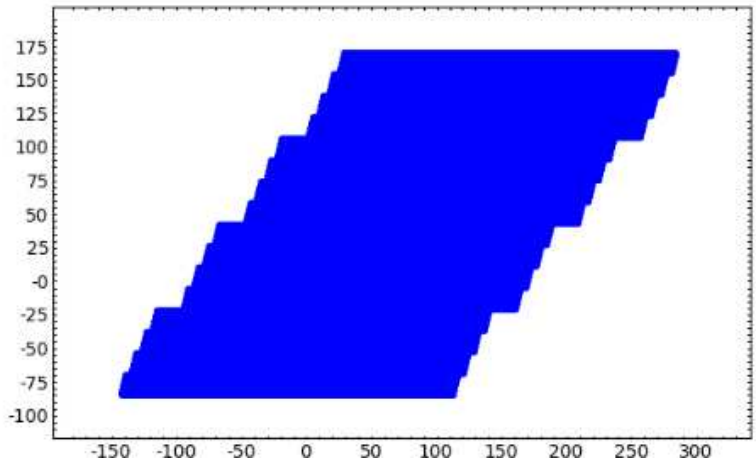
```
(12, 10)
```

```
v=Dn(3)
v.sort(lambda x,y: cmp(reversed(x),reversed(y)))
```

```
w=[x for x in v if x[1]==11]
w
```

```
[(-14, 11), (-13, 11), (-12, 11), (-11, 11), (-10, 11), (-9, 11),
(-8, 11), (-7, 11), (-6, 11), (-5, 11), (-4, 11), (-3, 11), (-2,
11), (-1, 11), (0, 11), (1, 11), (2, 11), (3, 11), (4, 11), (5, 11),
(6, 11), (7, 11), (8, 11), (9, 11), (10, 11), (11, 11), (12, 11),
(13, 11), (14, 11), (15, 11), (16, 11), (17, 11), (18, 11), (19,
11), (20, 11), (21, 11), (22, 11), (23, 11), (24, 11), (25, 11),
(26, 11), (27, 11), (28, 11), (29, 11), (30, 11), (31, 11), (32,
11), (33, 11), (34, 11), (35, 11), (36, 11), (37, 11), (38, 11),
(39, 11), (40, 11), (41, 11), (42, 11), (43, 11), (44, 11), (45,
11), (46, 11), (47, 11), (48, 11), (49, 11)]
```

```
show(points([v.list() for v in Dn(4)]), frame=True, axes=False)
```



```
A = matrix([[4,1],[0,4]])
D = [vector([-1, -1]), vector([-1, 0]), vector([-1, 1]), vector([-1, 2]),
vector([0, -1]), vector([0, 0]), vector([0, 1]), vector([0, 2]),
vector([1, -1]), vector([1, 0]), vector([1, 1]), vector([1, 2]),
vector([2, -1]), vector([2, 0]), vector([2, 1]), vector([2, 2])]

@interact
def f(A = matrix([[4,1],[0,4]]), D = '[[-1, -1], [-1, 0], [-1, 1], [-1, 2], [0, -1],
[0, 0], [0, 1], [0, 2], [1, -1], [1, 0], [1, 1], [1, 2], [2, -1], [2, 0], [2, 1], [2,
2]]', k=(3..17)):
    print "Det = ", A.det()
    D = matrix(eval(D)).rows()
    def Dn(k):
        ans = []
        for d in Tuples(D, k):
            s = sum(A^(-(n+1))*d[n] for n in range(k))
            ans.append(s)
        return ans

    G = points([v.list() for v in Dn(k)])

    show(G, frame=True, axes=False)
```

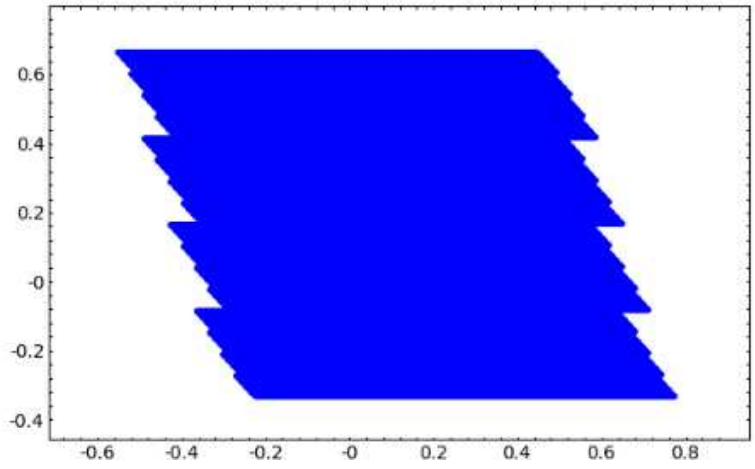
A

4	1
0	4

D [[-1, -1], [-1, 0], [-1, 1], [-1, 2], [0, -1], [0, 0], [0, 1], [0, 2], [1, -1], [1, 0], [1, 1], [1, 2], [2, -1], [2, 0], [2

k

```
Det =  16
```



```
A = matrix([[4,1],[0,4]])
D = [vector([-1, -1]), vector([-1, 0]), vector([-1, 1]), vector([-1, 2]),
vector([0, -1]), vector([0, 0]), vector([0, 1]), vector([0, 2]),
vector([1, -1]), vector([1, 0]), vector([1, 1]), vector([1, 2]),
vector([2, -1]), vector([2, 0]), vector([2, 1]), vector([2, 2])]

@interact
def f(A = matrix([[4,1],[0,4]]), D = '[[-1, -1], [-1, 0], [-1, 1], [-1, 2], [0, -1],
[0, 0], [0, 1], [0, 2], [1, -1], [1, 0], [1, 1], [1, 2], [2, -1], [2, 0], [2, 1], [2,
2]]', k=(3..17)):
    print "Det = ", A.det()
    D = matrix(eval(D)).rows()
    def Dn(k):
        ans = []
        for d in Tuples(D, k):
            s = sum(A^(n)*d[n] for n in range(k))
            ans.append(s)
        return ans

    G = points([v.list() for v in Dn(k)])

    show(G, frame=True, axes=False)
```

A

4	1
0	4

D [[-1, -1], [-1, 0], [-1, 1], [-1, 2], [0, -1], [0, 0], [0, 1], [0, 2], [1, -1], [1, 0], [1, 1], [1, 2], [2, -1], [2, 0], [2

k

Det = 16

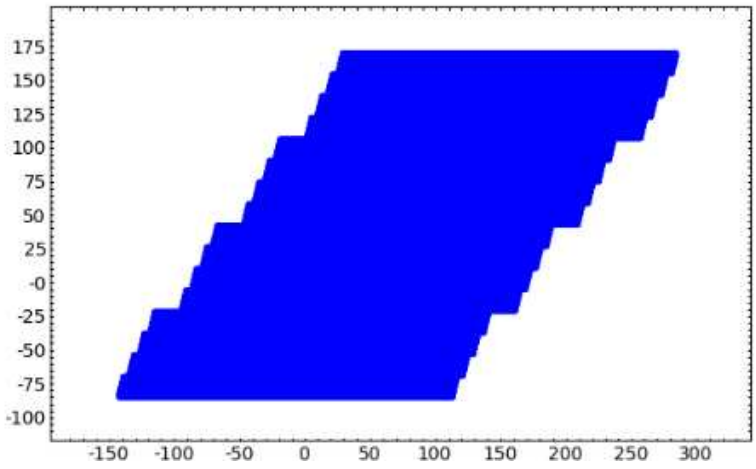


Figure B.7: 4104

Fractal Cloud

```
A = matrix([[0,0,2],[1,0,1],[0,1,-1]])
D = '[[0,0,0],[1,0,0]]'
```

```
def Dn(D,A,k):
    ans = []
    for d in Tuples(D, k):
        s = sum(A^(-n-1)*d[n] for n in range(k))
        ans.append(s)
    return ans
```

```
@interact
def f(A = matrix([[0,0,2],[1,0,1],[0,1,-1]]), D = '[[0,0,0],[1,0,0]]', k=(3..15)):
    print "Det = ", A.det()
    D = matrix(eval(D)).rows()
    print "D:"
    print D
    G = point3d([v.list() for v in Dn(D,A,k)], size=8)#, opacity=.85)


    show(G, axes=False, frame=False)
```

A

0	0	2
1	0	1
0	1	-1

D [[0,0,0],[1,0,0]]

k

```
Det =  2
D:
[(0, 0, 0), (1, 0, 0)]
```

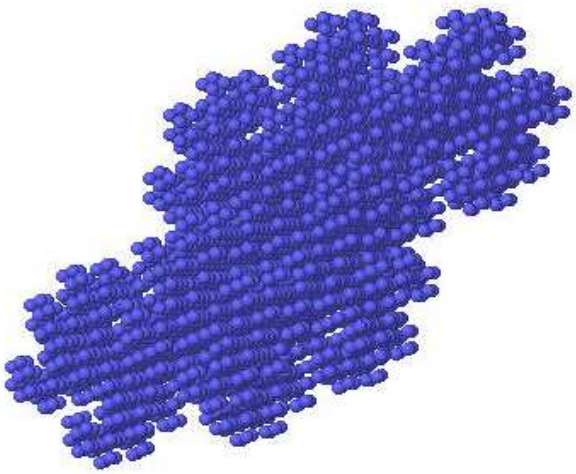


Figure B.8: Fractal Cloud

Jordan Form-31-3D

```
A = matrix([[3,1,0],[0,3,1],[0,0,3]])
D = '[[-1, -1, -1], [-1, -1, 0], [-1, -1, 1], [-1, 0, -1], [-1, 0, 0], [-1, 0, 1],
[-1, 1, -1], [-1, 1, 0], [-1, 1, 1], [0, -1, -1], [0, -1, 0], [0, -1, 1], [0, 0, -1],
[0, 0, 0], [0, 0, 1], [0, 1, -1], [0, 1, 0], [0, 1, 1], [1, -1, -1], [1, -1, 0], [1,
-1, 1], [1, 0, -1], [1, 0, 0], [1, 0, 1], [1, 1, -1], [1, 1, 0], [1, 1, 1]]'
```

```
def Dn(D,A,k):
    ans = []
    for d in Tuples(D, k):
        s = sum(A^(-(n+1))*d[n] for n in range(k))
        ans.append(s)
    return ans
```

```
@interact
def f(A = matrix([[3,1,0],[0,3,1],[0,0,3]]), D = '[[-1, -1, -1], [-1, -1, 0], [-1,
-1, 1], [-1, 0, -1], [-1, 0, 0], [-1, 0, 1], [-1, 1, -1], [-1, 1, 0], [-1, 1, 1], [0,
-1, -1], [0, -1, 0], [0, -1, 1], [0, 0, -1], [0, 0, 0], [0, 0, 1], [0, 1, -1], [0, 1,
0], [0, 1, 1], [1, -1, -1], [1, -1, 0], [1, -1, 1], [1, 0, -1], [1, 0, 0], [1, 0, 1],
[1, 1, -1], [1, 1, 0], [1, 1, 1]]', k=(2..15)):
    print "Det = ", A.det()
    D = matrix(eval(D)).rows()
    print "D:"
    print D
    G = point3d([v.list() for v in Dn(D,A,k)], size=8)#, opacity=.85)

    show(G, axes=False, frame=False)
```

A

3	1	0
0	3	1
0	0	3

D [[-1, -1, -1], [-1, -1, 0], [-1, -1, 1], [-1, 0, -1], [-1, 0, 0], [-1, 0, 1], [-1, 1, -1], [-1, 1, 0], [-1, 1, 1], [0,

k

```
Det =  27
D:
[(-1, -1, -1), (-1, -1, 0), (-1, -1, 1), (-1, 0, -1), (-1, 0, 0),
(-1, 0, 1), (-1, 1, -1), (-1, 1, 0), (-1, 1, 1), (0, -1, -1), (0,
-1, 0), (0, -1, 1), (0, 0, -1), (0, 0, 0), (0, 0, 1), (0, 1, -1),
(0, 1, 0), (0, 1, 1), (1, -1, -1), (1, -1, 0), (1, -1, 1), (1, 0,
-1), (1, 0, 0), (1, 0, 1), (1, 1, -1), (1, 1, 0), (1, 1, 1)]
```

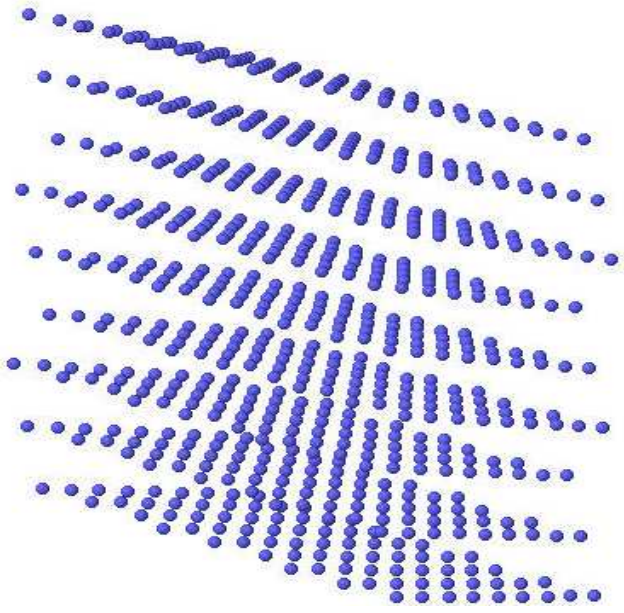


Figure B.9: Jordan Form-31-3D

Jordan Form-41-3D

```
A = matrix([[4,1,0],[0,4,1],[0,0,4]])
D = '[[-1, -1, -1], [-1, -1, 0], [-1, -1, 1], [-1, -1, 2], [-1, 0, -1], [-1, 0, 0],
[-1, 0, 1], [-1, 0, 2], [-1, 1, -1], [-1, 1, 0], [-1, 1, 1], [-1, 1, 2], [-1, 2, -1],
[-1, 2, 0], [-1, 2, 1], [-1, 2, 2], [0, -1, -1], [0, -1, 0], [0, -1, 1], [0, -1, 2],
[0, 0, -1], [0, 0, 0], [0, 0, 1], [0, 0, 2], [0, 1, -1], [0, 1, 0], [0, 1, 1], [0, 1,
2], [0, 2, -1], [0, 2, 0], [0, 2, 1], [0, 2, 2], [1, -1, -1], [1, -1, 0], [1, -1, 1],
[1, -1, 2], [1, 0, -1], [1, 0, 0], [1, 0, 1], [1, 0, 2], [1, 1, -1], [1, 1, 0], [1,
1, 1], [1, 1, 2], [1, 2, -1], [1, 2, 0], [1, 2, 1], [1, 2, 2], [2, -1, -1], [2, -1,
0], [2, -1, 1], [2, -1, 2], [2, 0, -1], [2, 0, 0], [2, 0, 1], [2, 0, 2], [2, 1, -1],
[2, 1, 0], [2, 1, 1], [2, 1, 2], [2, 2, -1], [2, 2, 0], [2, 2, 1], [2, 2, 2]]'
```

```
def Dn(D,A,k):
    ans = []
    for d in Tuples(D, k):
        s = sum(A^(-(n+1))*d[n] for n in range(k))
        ans.append(s)
    return ans
```

```
@interact
def f(A = matrix([[4,1,0],[0,4,1],[0,0,4]]), D = '[[-1, -1, -1], [-1, -1, 0], [-1,
-1, 1], [-1, -1, 2], [-1, 0, -1], [-1, 0, 0], [-1, 0, 1], [-1, 0, 2], [-1, 1, -1],
[-1, 1, 0], [-1, 1, 1], [-1, 1, 2], [-1, 2, -1], [-1, 2, 0], [-1, 2, 1], [-1, 2, 2],
[0, -1, -1], [0, -1, 0], [0, -1, 1], [0, -1, 2], [0, 0, -1], [0, 0, 0], [0, 0, 1],
[0, 0, 2], [0, 1, -1], [0, 1, 0], [0, 1, 1], [0, 1, 2], [0, 2, -1], [0, 2, 0], [0, 2,
1], [0, 2, 2], [1, -1, -1], [1, -1, 0], [1, -1, 1], [1, -1, 2], [1, 0, -1], [1, 0,
0], [1, 0, 1], [1, 0, 2], [1, 1, -1], [1, 1, 0], [1, 1, 1], [1, 1, 2], [1, 2, -1],
[1, 2, 0], [1, 2, 1], [1, 2, 2], [2, -1, -1], [2, -1, 0], [2, -1, 1], [2, -1, 2], [2,
0, -1], [2, 0, 0], [2, 0, 1], [2, 0, 2], [2, 1, -1], [2, 1, 0], [2, 1, 1], [2, 1, 2],
[2, 2, -1], [2, 2, 0], [2, 2, 1], [2, 2, 2]]', k=(2..15)):
    print "Det = ", A.det()
    D = matrix(eval(D)).rows()
    print "D:"
    print D
    G = point3d([v.list() for v in Dn(D,A,k)], size=8)#, opacity=.85)

    show(G, axes=False, frame=False)
```

A

4	1	0
0	4	1
0	0	4

D [[-1, -1, -1], [-1, -1, 0], [-1, -1, 1], [-1, -1, 2], [-1, 0, -1], [-1, 0, 0], [-1, 0, 1], [-1, 0, 2], [-1, 1, -1], [-

k

```
Det =  64
D:
[(-1, -1, -1), (-1, -1, 0), (-1, -1, 1), (-1, -1, 2), (-1, 0, -1),
(-1, 0, 0), (-1, 0, 1), (-1, 0, 2), (-1, 1, -1), (-1, 1, 0), (-1, 1,
1), (-1, 1, 2), (-1, 2, -1), (-1, 2, 0), (-1, 2, 1), (-1, 2, 2), (0,
-1, -1), (0, -1, 0), (0, -1, 1), (0, -1, 2), (0, 0, -1), (0, 0, 0),
(0, 0, 1), (0, 0, 2), (0, 1, -1), (0, 1, 0), (0, 1, 1), (0, 1, 2),
(0, 2, -1), (0, 2, 0), (0, 2, 1), (0, 2, 2), (1, -1, -1), (1, -1,
0), (1, -1, 1), (1, -1, 2), (1, 0, -1), (1, 0, 0), (1, 0, 1), (1, 0,
2), (1, 1, -1), (1, 1, 0), (1, 1, 1), (1, 1, 2), (1, 2, -1), (1, 2,
0), (1, 2, 1), (1, 2, 2), (2, -1, -1), (2, -1, 0), (2, -1, 1), (2,
-1, 2), (2, 0, -1), (2, 0, 0), (2, 0, 1), (2, 0, 2), (2, 1, -1), (2,
1, 0), (2, 1, 1), (2, 1, 2), (2, 2, -1), (2, 2, 0), (2, 2, 1), (2,
2, 2)]
```

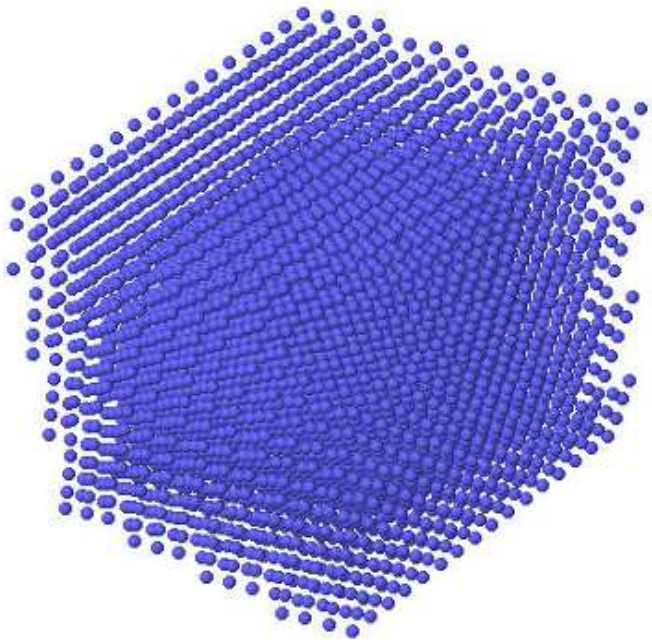


Figure B.10: Jordan Form-41-3D