İSTANBUL TECHNICAL UNIVERSITY ⋆ INSTITUTE OF SCIENCE AND TECHNOLOGY

DISCONNECTEDNESS OF SELF-AFFINE SETS AND A METHOD FOR FINDING THE CONVEX HULL OF SELF-AFFINE SETS

M.Sc. Thesis by
İLKER KOÇYİĞİT

Department : Mathematics

Programme : Mathematical Engineering

MAY 2007

İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

DISCONNECTEDNESS OF SELF-AFFINE SETS AND A METHOD FOR FINDING THE CONVEX HULL OF SELF-AFFINE SETS

M.Sc. Thesis by

İLKER KOÇYİĞİT

509041003

Date of submission : 7 May 2007

Date of defense examination : 12 June 2007

Supervisor (Chairman) : Assoc. Prof. İbrahim KIRAT

Members of the Examining Committee Prof. Avadis HACINLIYAN (Y.Ü.)

Prof. Ayşe H. BİLGE (İ.T.Ü.)

MAY 2007

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

KENDİNE İLGİN KÜMELERİN BAĞLANTISIZLIĞI VE KENDİNE İLGİN KÜMELERİN DIŞBÜKEY DÜRÜMÜNÜ BULAN BİR YÖNTEM

YÜKSEK LİSANS TEZİ

İLKER KOÇYİĞİT

509041003

Tezin Enstitüye Verildiği Tarih : 7 Mayıs 2007
Tezin Savunulduğu Tarih : 12 Haziran 2007

Tez Danışmanı: Doç. Dr. İbrahim KIRAT
Diğer Jüri Üyeleri Prof. Dr. Avadis HACINLIYAN (Y.Ü.)
Prof. Dr. Ayşe H. BİLGE (İ.T.Ü.)

MAYIS 2007

ACKNOWLEDGEMENTS

I would like to thank my thesis supervisor İbrahim Kırat for all of the advice and encouragement he provided during the course of this thesis, and also for introducing me to the field of fractal geometry.

May 2007 İlker Koçyiğit

TABLE OF CONTENTS

LIST OF FIGURES

Page No

KENDİNE İLGİN KÜMELERİN BAĞLANTISIZLIĞI VE KENDİNE İLGİN KÜMELERİN DIŞBÜKEY DÜRÜMÜNÜ BULAN BİR YÖNTEM

ÖZET

$\mathbb{R}^n$ de $AT = \bigcup_{i=1}^{q}(T + d_i)$ özelliğine sahip T kümesine *kendine-ilgin küme* ve $D = \{d_1, d_2, ..., d_q\} \subseteq \mathbb{R}^n$ kümesine de *sayak kümesi* denir. Burada A, genişleyen (yani tüm özdeğerleri mutlak değerce 1'den büyük) gerçel değerli bir $n \times n$ matristir. Bu çalışmada, $\mathbb{R}^n$ deki kendine-ilgin bir küme için, şu farklı fakat ilintili iki soruyu ele aldık: 1) Verilen kendine-ilgin bir kümenin bağlantısızlığına nasıl ve hangi şartlar altında karar verebiliriz? 2) Verilen kendine-ilgin bir kümenin dışbükey dürümüne nasıl karar verebiliriz? Bu sorulardan ilkiyle ilgili olarak yapılan çalışmada, $\mathbb{R}^n$ deki her bağlantısız kompakt kümenin, $d(c, T\backslash c) > 0$ koşulunu sağlayan bir c kompakt altkümesinin olması gerektiği gerçeğinden yola çıkılmıştır. Kendine-ilgin bir T kümesinin verilen bir $\varepsilon > 0$ için $d(c, T\backslash c) > \varepsilon$ koşulunu sağlayan bir c kompakt altkümesi olması durumunda, T kümesinin bağlantısızlığına karar verecek bir yöntem sunulmuştur. Ayrıca verilen bu kümenin bağlantısızlığını ya da yukarıda bahsedilen koşulu sağlamadığını belirleyen bir bilgisayar uygulaması da sunulmuştur. İkinci soru için ise, T'nin dışbükey dürümünün politop olabilmesi için bir yeter koşul sunulmuştur. Ayrıca, bu gibi politopların köşe noktalarını bulabilmek amacıyla, inşa yöntemi kullanılarak yeter koşul ispatlanmıştır. Eğer dışbükey dürüm bir politop ise, verilen yeter koşulun aynı zamanda gerek koşul olduğunu tahmin ediyor ve sanı olarak belirtiyoruz.

DISCONNECTEDNESS OF SELF-AFFINE SETS AND A METHOD FOR FINDING THE CONVEX HULL OF SELF-AFFINE SETS

SUMMARY

A *self-affine set* in $\mathbb{R}^n$ is a set T with property $AT = \bigcup_{i=1}^{q}(T + d_i)$, where A is an expanding $n \times n$ real matrix (i.e., all its eigenvalues have modulus > 1), and $D = \{d_1, d_2, ..., d_q\} \subseteq \mathbb{R}^n$ is the *digit set*. In this work, for a self-affine set in $\mathbb{R}^n$, we study the following two different but related questions: 1) How can we decide the disconnectedness of the given self-affine set, and in what conditions? 2) How can we find the convex hull of the given self-affine set? For the first one, we attack the question by using the fact that every disconnected compact set in $\mathbb{R}^n$ has a compact subset c such that the distance $d(c, T\backslash c)$ is positive. We introduce a method to decide the disconnectedness of a given self-affine set T if T has a compact subset c such that $d(c, T\backslash c) > \varepsilon$ for a given $\varepsilon > 0$. We also give the computer implementation to decide the disconnectedness of the given set or that it does not satisfy the condition above. For the second question, we give a sufficient condition for the convex hull of T to be a polytope. Furthermore we can find the vertices of such polytopes. For this purpose, we use the construction method to prove the sufficiency. We also present the computer implementation of the method. We conjecture that if the convex hull is a polytope then the sufficient condition is also a necessary condition.

1. INTRODUCTION

In this work, $M_n(\mathbb{Z})$ denotes the set of $n \times n$ matrices with entries in $\mathbb{Z}$. Let A be an $n \times n$ expanding matrix in $M_n(\mathbb{Z})$, i.e., all eigenvalues of A have moduli greater than 1. Let $D = \{d_1, d_2, ..., d_q\} \subset \mathbb{R}^n$ be a digit set with $q \in \mathbb{N}$ elements. Then it is known ([1] [2] [3]) that there exists a unique non-empty compact set $T = T(A, D)$ satisfying

$$AT = \bigcup_{i=1}^{q} (T + d_i), \tag{1.2}$$

which is explicitly given by

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

T is called a self-affine set. If A is a similarity; i.e., $||Ax|| = \lambda ||x||$ $(\lambda > 0)$ for all $x \in \mathbb{R}^n$, then T is called a self-similar set.

Without loss of generality, we may assume (by Proposition 2.1 in [4]) that $d_1 = 0$.

Topological properties of T, although more fundamental than metric ones, are more difficult to determine computationally [5],[6],[7]. There is a growing literature on the formalization and representations of topological questions for computer applications and on the study of algorithms [8]. One of the very interesting aspects of the self-affine sets is the connectedness.

For the one dimensional case, the following result is the Corollary 4.4 of [4].

Let $q \in \mathbb{Z}$, $|q| \geq 2$. Suppose $A = [q]$ and $D \subseteq \mathbb{R}$ is a $|q|$-digit set. Then $T(A, D)$ is a connected tile if and only if, up to a translation, $D = \{0, a, 2a, ..., (|q| - 1)a\}$ for some $a > 0$.

In higher dimensions, there is no method known to decide whether a self-affine set T is connected or disconnected. There are a number of results for specific cases

with some restriction on digit set D or on matrix A ([9], [10],[4] , [11]). Robins et. al. ([12], [13]) studies connectedness by considering quantities similar to the ones in the definition of box-counting dimension. Those quantities may be computed for self-similar sets. Therefore, the examples in [12] are all self-similar sets. In this work, we will study connectedness in the more general case of self-affine sets and our approach is different from the one in [12], [13].

Finding or estimating the convex hull of a self-affine set is another important topic. Some interesting properties of the convex hull of self-affine sets are studied in [14] and [15]. In [15] a formula for the area of the convex hull of a planar self-affine tile is given. A necessary and sufficient condition for the convex hull to be a polytope is given in [14]. In our work, we give a different sufficient condition for the convex hull of T to be a polytope. Our sufficient condition is different from the one in [14], in the sense that we use a constructive method to prove the sufficiency. Therefore, we are able to find the vertex points of the self-affine sets that satisfy our sufficient condition.

We divide the work into three chapters. In Chapter 2, we give a method which will decide in finite steps whether a self-affine set is disconnected with a gap more than a given positive ϵ. We make use of the fact that if T is disconnected then it has two compact subsets S_1 and S_2 such that $d(S_1, S_2) > 0$ and $T = S_1 \cup S_2$, where d is the Euclidean distance between sets. In order to implement this method on a computer, we need a bounding set for T. In Chapter 3, we give a sufficient condition to find the convex hull of T. In Chapter 4, we give the computer implementation of the methods and some examples.

2. DISCONNECTEDNESS OF CERTAIN SELF-AFFINE SETS

A necessary and sufficient condition for the connectedness of T is given in [4]. In this section, using a similar condition, we give a method that decides the disconnectedness of T if it has two subsets apart from each other more than a distance of a given $\varepsilon > 0$. By using the fact that a disconnected compact T has two closed subsets S_1 and $T\backslash S_1$ such that $d(S_1, T\backslash S_1)$ is positive, we will define the notion of disconnectedness with a gap more than a given number ε, and we will give a method which decides whether the set T is disconnected in finite steps. We start with definitions.

Definition 2.1. We call a subset of T clopen if it is both closed and open in T. For a non-empty compact set $T \subset \mathbb{R}^n$ with subspace topology, let C be the set of clopen sets $c \subset T$ such that $c \neq \emptyset, T$, and let $\varepsilon > 0$. Then we say that T is ε-connected if $\varepsilon \geq \sup_{c \in C}\{d(c, T\backslash c)\}$. Otherwise, T is said to be ε-disconnected.

Remark: After the completion of this work, we noticed that a similar definition is given in [12] and [13].

If T is ε-disconnected then there is a gap more than ε between some two clopen sets of T. If $\gamma \geq \varepsilon$ then it is clear that if T is ε -connected then it is also γ-connected, and similarly if a T is γ-disconnected then it is also ε-disconnected.

Definition 2.2. Let $\varepsilon > 0$. An ε-chain between points x and y is a finite sequence of points $\{x_1, x_2, \ldots, x_n\}$ such that $x_1 = x$, $x_n = y$ and $d(x_i, x_{i+1}) \leq \varepsilon$ for $1 \leq i \leq n-1$. A metric space X is ε-chain connected if there is an ε-chain between any two points of it.

It was also mentioned in [12] that Cantor defined an equivalent notion ([16]) by introducing the term ε-chain. X is called Cantor-connected if for all $\varepsilon > 0$, it is ε-chain connected.

Remark: A Cantor-connected set may not need to be connected. For example the set of rational numbers is Cantor-connected but not connected.

Let $T_i = A^{-1}(T + d_i)$,$i = 1, \ldots, q$ and $q = |D|$, in the rest of this work. It can be seen that the following holds.

Proposition 2.3. A non-empty compact set $T \subset \mathbb{R}^n$ is disconnected if and only if it is ε-disconnected for some $\varepsilon > 0$. Equivalently, T is connected if and only if it is ε-connected for all $\varepsilon > 0$.

Proof. Assume T is ε-disconnected for some $\varepsilon > 0$. By the definition of ε-disconnectedness, there exists a clopen set $c \neq \emptyset, T$ in T such that $d(c, T\backslash c)\} > \varepsilon$. Then T is disconnected. To prove the necessity, we assume that T is disconnected. Then there exists a clopen subset $c \neq \emptyset, T$ in T. Because T is compact, c and $T\backslash c$ are also compact in $\mathbb{R}^n$. The distance between two non-empty disjoint compact sets is positive, therefore $\sup_{c \in C} d(c, T\backslash c)\} > \varepsilon$ for some $\varepsilon > 0$; i.e., T is ε-disconnected for some $\varepsilon > 0$.

In the following, we give a scheme that decides a self affine set T is disconnected or it is ε-connected for a given $\varepsilon > 0$.

Let $I = \{(i, j) : 1 \leq i \leq q, \text{ and } 1 \leq j \leq q\}$, and

$$\Sigma = \{(i, j) \in I : \ T_i \cap T_j \neq \emptyset\}$$

and

$$\Sigma_\varepsilon = \{(i, j) \in I : d(T_i, T_j) \leq \varepsilon\}$$

where q is the number of elements in D.

Definition 2.4. For a non-empty compact set T, we say T is Σ_ε-connected if and only if for every pair $(i, i') \in I$, there exists indices $i_1, \ldots, i_l$ such that $i_1 = i$,

$i_l = i'$ and $(i_k, i_{k+1}) \in \Sigma_\varepsilon$ for $k \in \{1, 2, \ldots, l-1\}$. We say T is Σ-connected if and only if for every pair $(i, i') \in I$, there exists indices $i_1, \ldots, i_l$ such that $i_1 = i$, $i_l = i'$ and $(i_k, i_{k+1}) \in \Sigma$ for $k \in \{1, 2, \ldots, l-1\}$.

The following theorem can be found in [4].

Theorem 2.5. Let $A \in M_n(\mathbb{Z})$ be an expanding matrix and let $D = \{d_1, ..., d_q\} \subseteq \mathbb{R}^n$ be a q-digit set. Then T is connected if and only if T is Σ-connected.

In order to decide the ε-disconnectedness, it is not possible to check all the clopen subsets of T. For this reason, we need a slightly changed version of Theorem 2.5 as follows.

Theorem 2.6. Given any $\varepsilon > 0$, let $A \in M_n(\mathbb{Z})$ be an expanding matrix and let $D = \{d_1, \ldots, d_q\} \subseteq \mathbb{R}^n$ be a q-digit set. Then T is ε-connected if and only if T is Σ_ε-connected.

Proof. The necessity is obvious. To prove the sufficiency, let

$$S_j(x) = A^{-1}(x + d_j), \quad j = 1, \ldots, q$$

and for $J = (j_1, \ldots, j_k)$, we let $S_J = S_{j_1} \circ \ldots \circ S_{j_k}$. There exists a closed ball B such that $T \subseteq B$.

It is clear that $S_J(B)$ is connected. We claim that $V_k = \bigcup_{|J|=k} S_J(B)$ is ε-chain connected. For $k = 1$, we note that $T_j = S_j(T) \subseteq S_j(B)$ and that the Σ_ε-connectedness property on T implies that V_1 is ε-chain connected. For the inductive step $k+1$, we write

$$\bigcup_{|J|=k+1} S_J(B) = S_1(V_k) \cup \ldots \cup S_q(V_k).$$

By the induction hypothesis, V_k is ε-chain connected and because $T \subseteq V_k$, we get $T_j = S_j(T) \subseteq S_j(V_k)$. Again Σ_ε-connectedness property on T implies that V_{k+1} is ε-chain connected. This proves the claim.

To show that T is ε-connected, we assume that there exists a clopen set c such that $d(c, T\backslash c) > \varepsilon + \delta > \varepsilon$. Since $\{V_k\}_{k=1}^{\infty}$ converges to T in the Hausdorff metric (In [3] ,Thm 2.6, pg.30), then for k large enough, we have $d_H(T, V_k) < \delta/4$ and $diam(S_J(B)) < \delta/4$ for all $|J| = k$. Let

$$\Psi_1 = \left\{ J : |J| = k, d(c, S_J(B)) < \frac{\delta}{4} \right\}$$

$$\Psi_2 = \left\{ J : |J| = k, d(T\backslash c, S_J(B)) < \frac{\delta}{4} \right\}$$

Then from $d(c, T\backslash c) > \epsilon + \delta$, we have $d(\bigcup_{J\in\Psi_1} S_J(B)\ ,\ \bigcup_{J\in\Psi_2} S_J(B)) \geq \varepsilon$. This contradicts the assertion that $V_k = \bigcup_{|J|=k} S_J(B)$ is ε-chain connected.

We continue to construct the method. By this theorem, we guaranteed that in order to check the ε-disconnectedness of T it is sufficient to check the distances between each T_i and T_j. Let $D' = D - D$ and enumerate elements of D' as $d_{i,j} = d_i - d_j$. That is, $D' = \{d_{i,j} : d_{i,j} = d_i - d_j, \quad d_i, d_j \in D\}$. It is possible that $d_{i,j} = d_{i',j'}$ for $d_i \neq d_{i'}$ and $d_j \neq d_{j'}$. Now D' defines a new attractor $T' = T(A, D')$. The inequality above becomes $d\left(A^{-1}d_{i,j}, A^{-1}T'\right) \leq \varepsilon$. We give this in the following proposition.

Proposition 2.7. Let T', T, D, D' be as defined above. Let $d_{i,j} \in D'$. Then $d(T_i, T_j) = d(A^{-1}d_{i,j}, A^{-1}T')$.

Proof. $d(T_i, T_j) = \inf_{\substack{i,j\in\{1,\ldots,q\}\\ t_k,t_m\in T}} \{|A^{-1}(t_k + d_i) - A^{-1}(t_m + d_j)|\} = \inf_{\substack{i,j\in\{1,\ldots,q\}\\ t_k,t_m\in T}} \{|A^{-1}d_{i,j} - A^{-1}(t_k - t_m)|\} = \inf_{\substack{d_{i,j}\in D'\\ t_{k,m}\in T'}} \{|A^{-1}d_{i,j} - A^{-1}t_{k,m}|\} = d(A^{-1}d_{i,j}, A^{-1}T')$

This proposition shows that $(i, j) \in \Sigma_\varepsilon$ if and only if $d(A^{-1}d_{i,j}, A^{-1}T') \leq \varepsilon$. Now we need to check the distance between a point $A^{-1}d_{i,j}$ and the set $A^{-1}T'$. For the following proof, we note that

$$A^{-1}T' = \bigcup_{d_{i,j}\in D'} A^{-1}(A^{-1}T' + A^{-1}d_{i,j})$$

Now by letting $D'' = A^{-1}D'$ and $T'' = A^{-1}T'$ we obtain

$$T'' = \bigcup_{d\in D''} A^{-1}(T'' + d)$$

The following proposition shows that we can check if $d(p, T'') > 0$ or $d(p, T'') \leq \varepsilon$ for each $p \in D''$.

Proposition 2.8. Let $T'' = T(A, D'')$ and let p be a point and let $\varepsilon > 0$. Then in finite steps, it can be decided that $d(p, T'') > 0$ or $d(p, T'') \leq \varepsilon$.

Proof. We use an argument similar to that in Theorem 2.6. Let B be any compact set containing T''. Let

$$W_j(x) = A^{-1}(x + d_j), \quad d_j \in D''$$

and for $J = (j_1, \ldots, j_k)$, we let $W_J = W_{j_1} \circ \ldots \circ W_{j_k}$. The proof will follow from the fact that $\bigcup_{|J|=k} W_J(B)$ converges to the attractor $T'' = T(A, D'')$ and $T'' \subseteq \bigcup_{|J|=k} W_J(B)$ for any k. Because A^{-t} is a contraction for some positive t, we can find a positive k_0 such that the diameter of the set, $diam(W_J(B))$, will be equal to $\epsilon \leq \varepsilon$ for $|J| = k_0$. We start from $k = 1$ and check whether $p \in \bigcup_{|J|=k} W_J(B)$. If for some $k \leq k_0$, $p \notin \bigcup_{|J|=k} W_J(B)$ then $d(p, T'') > 0$ and we are done. Otherwise, $p \in \bigcup_{|J|=k_0} W_J(B)$ and hence $d(p, T'') \leq \epsilon$ because each $W_J(B)$, $|J| = k_0$, contains at least one point from T''. So our search will end in at most k_0 steps.

A weaker result of the above proposition says that if the distance between point p and $A^{-1}T'$ is further than a given $\varepsilon > 0$, it is guaranteed to be detected in finite steps. Note that the resulting inequalities of the above theorem is not mutually exclusive; that is, a point p may satisfy both conditions $d(p, A^{-1}T') > 0$ and $d(p, A^{-1}T') \leq \varepsilon$. In such a case, the procedure in the proof may end with any of two depending on the point p. The number of $W_J(B)$ to check in the above proof is at most q^k where q is the number of elements in the digit set. But practically the complexity of the algorithm is much less than this, in some cases it is even a linear time algorithm.

By Proposition 2.7 and 2.8 we know that for each T_i and T_j, we can decide whether they are disjoint or $d(T_i, T_j) \leq \varepsilon$. So we can construct the symmetric

matrix $F = F(\varepsilon) = [f_{i,j}]$, where

$$f_{i,j} = \begin{cases} 0, & \text{if } T_i \text{ and } T_j \text{ are disjoint,} \\ 1, & \text{if } d(T_i, T_j) \leq \varepsilon. \end{cases}$$

By using the theorem above and previous two propositions, we give the following result on ε- connectedness.

Theorem 2.9. Given any $\varepsilon > 0$, let $A \in M_n(\mathbb{Z})$ be an expanding matrix and let $D \subseteq \mathbb{R}^n$ be a digit set with q elements. If F^q has any zero entry, then T is disconnected; otherwise, T is ε-connected.

Proof. Let $F^q = [b_{i,j}]$. Then by induction,

$$b_{i,j} = \sum_{1 \leq i_1, \ldots, i_{q-1} \leq q} f_{i,i_1} \; f_{i_1,i_2} \; f_{i_2,i_3} \; \cdots \; f_{i_{q-1},j}.$$

If $b_{m,n} = 0$ for some m and n, then for the pair $(m, n) \in I$, there exist no indices $i_1, \ldots, i_l$ such that $i_1 = m$, $i_l = n$ and $(i_k, i_{k+1}) \in \Sigma$ for $k \in \{1, 2, \ldots, l-1\}$. Therefore, T is not Σ-connected. By Theorem 2.5, T is disconnected.

If $b_{i,j} \neq 0$ for every pair $(i, j) \in I$, there exist indices $i_1, \ldots, i_l$ such that $i_1 = i$, $i_l = j$ and $(i_k, i_{k+1}) \in \Sigma_\varepsilon$ for $k \in \{1, 2, \ldots, l-1\}$. Therefore, T is Σ_ε-connected. By Theorem 2.6, T is ε-connected.

By using the above theorem, in finite steps it can be decided T is disconnected or T is ε-connected. In particular, the disconnectedness of T can be decided if T is not ε-connected.

3. A METHOD FOR FINDING THE CONVEX HULL

Definition 3.1. The convex hull of a set $S \in \mathbb{R}^n$, denoted by $\mathcal{C}(S)$, is the intersection of all convex sets in $\mathbb{R}^n$ containing S. A point p is called a vertex of a convex polytope C if and only if $p \neq tc_1 + (1-t)c_2$, for all t, c_1, c_2 such that $0 < t < 1$, $c_1, c_2 \in C$ and $c_1 \neq c_2$. We call a finite set $P = \{p_1, \ldots, p_k\}$ a convex polyhedron (or just a polyhedron for brevity) if P is the set of vertices (or extreme points) of a convex polytope. A point x is said to be a convex combination of points $a_1, \ldots, a_m$ in $\mathbb{R}^n$ if there exists scalars $\lambda_1, \ldots, \lambda_m \geq 0$ with $\lambda_1 + \ldots + \lambda_m = 1$ such that $x = \lambda_1 a_1, \ldots \lambda_m a_m$.

Thus we identify a convex polytope with its vertices. The following two are important results ([17]) relating the convex hull to convex combinations of points of a set.

Theorem 3.2. Let A be a set in $\mathbb{R}^n$. Then the convex hull of A is the set of all convex combinations of points of A.

Theorem 3.3. (Krein-Milman) Every compact convex set in $\mathbb{R}^n$ is the convex hull of its extreme points.

Note that given any finite set in $\mathbb{R}^n$, the convex hull $\mathcal{C}$ of these sets will be a compact set, and by the Krein-Milman theorem, the convex hull of its vertices is equal to $\mathcal{C}$. By Theorem 3.2 we know that convex combinations of these vertices will also give the convex hull $\mathcal{C}$. These facts will be used in the following parts.

Let $D = \{d_1, \ldots, d_q\} \in \mathbb{R}^n$, and

$$S_j(x) = A^{-1}(x + d_j), \quad j = 1, \ldots, q$$

so that $T = \bigcup_{j=1}^{q} S_j(T)$. For $J = (j_1, \ldots, j_k)$, we let $S_J = S_{j_1} \circ \ldots \circ S_{j_k}$. It can be seen that

$$A_k = \bigcup_{|J|=k} S_J(0) \qquad for\ k = 1, 2, \ldots,$$

converges to T as $k \to \infty$. Note that

$$A_1 = \{A^{-1}d_j : d_j \in D\}$$

$$A_2 = \{A^{-2}d_{j_2} + A^{-1}d_{j_1} : d_{j_1}, d_{j_2} \in D\}$$

$$\ldots$$

$$A_k = \{A^{-k}d_{j_k} + \ldots + A^{-1}d_{j_1} : d_{j_1}, \ldots, d_{j_k} \in D\}$$

Let $S \in \mathbb{R}^n$ be a finite set. We will denote the set of vertices of $\mathcal{C}(S)$ by $\mathcal{V}(S)$. Now let $\mathcal{V}_k = \mathcal{V}(A_k)$. Thus $\mathcal{V}_k$ is a polyhedron. Assuming 0 is in D, it can be easily seen that $A_k \subseteq A_{k'}$ if $k \leq k'$ and $\mathcal{C}(A_k) \subseteq \mathcal{C}(A_{k'})$.

Definition 3.4. A k-string of a sequence Y is a finite sequence which contains k consecutive terms of Y. An ∞-string of a sequence Y is a subsequence of Y which contains consecutive terms of Y. A recurring sequence Y is a sequence of the form $Y = (d_1, d_2, \ldots, d_n, d_1, d_2, \ldots, d_n, d_1, d_2, \ldots) = \overline{(d_1, d_2, \ldots, d_n)}$.

For example, let $Y = (d_1, d_2, \ldots, d_n, \ldots)$ then $d_2d_3d_4$ is a 3-string of Y, and $d_5d_6 \ldots d_{n+4}$ is a n-string of Y, and finally $d_2d_3 \ldots$ is an ∞-string of Y. For another example, we let $Y = \overline{(d_1, d_2, \ldots, d_n)}$, then there are at most n different ∞-string of Y. Also there are at most n different k-string of Y for any $k > 0$.

We will represent elements of A_k as strings. Let $p \in A_k$ then it will be in the form $p = A^{-k}d_{j_k} + \ldots + A^{-1}d_{j_1}$. We will represent p as a k-string $p = d_{j_1}d_{j_2} \ldots d_{j_k}$ where $d_{j_i} \in D$, $1 \leq i \leq k$.

Remark 3.5. We identify each element of A_k with a k-string. It is possible that two different k-strings may represent the same value in A_k (i.e. $A^{-1}d_1 + \ldots + A^{-k}d_k = A^{-1}b_1 + \ldots + A^{-k}b_k$ and $d_1d_2 \ldots d_k \neq b_1b_2 \ldots b_k$). We consider such k-strings as equivalent and they form an equivalence class. We actually identify each element of A_k with an equivalence class of k-strings and each class corresponds

to only one element. Therefore, we take only one k-string as the representative from each class. Two k-strings s_1, s_2 are equivalent if they represent the same point in A_k. We assume that two recurring sequences are equivalent if the sets of their k-strings are the same for all k.

We note that $\mathcal{C}(T)$ of a self-affine set T is a compact set. ([17], Thm 2.2.6., p.57.) Let $|S|$ denote the number of elements of a set S.

Theorem 3.6. $\mathcal{C}(T)$ of a self-affine set T is a polytope if $|\mathcal{V}_{i+1}| = |\mathcal{V}_i| = t$ for some i. Moreover if $|\mathcal{V}_{i+1}| = |\mathcal{V}_i| = t$ for some i, then there exist distinct sequences $Y_0, Y_1, \ldots, Y_m$ such that ∞-strings of them give exactly t different vertices of the polytope $\mathcal{C}(T)$.

The following lemma is Corollary 2.1.9 of the book [17], page 52.

Lemma 3.7. Let A, B, C be sets in $\mathbb{R}^n$. Suppose that A is non-empty and bounded, that B and C are closed and convex, and that $A + B = A + C$. Then $B = C$.

The following lemma can be deduced from the proof of Theorem 3.1.1 of the book [17], page 106.

Lemma 3.8. Let $P_1 = \{a_1, \ldots, a_n\}$ and $P_2 = \{b_1, \ldots, b_m\}$ be convex polyhedra. Then $\mathcal{C}(P_1) + \mathcal{C}(P_2) = \mathcal{C}(P_1 + P_2)$.

In order to prove the theorem, we need to prove a lemma.

Lemma 3.9. Let $P_1 = \{a_1, \ldots, a_n\}$ and $P_2 = \{b_1, \ldots, b_m\}$ be convex polyhedra and let P is the convex polyhedron defined by vertices of $\mathcal{C}(P_1 + P_2)$; i.e., $P = \mathcal{V}(P_1 + P_2)$. Then there exists a one-to-one map f from P_1 to P such that $f(a_i) \mapsto a_i + b$ for some $b \in P_2$.

Proof. If there exists no map defined in P_1 then there exists a point $a \in P_1$ such that $f(a)$ is not defined; that is, there exists no point in P in the form $a+b$, $b \in P_2$.

Now let $P_1' = P_1 \backslash \{a\}$, then, by Lemma 3.8, $\mathcal{C}(P_1)+\mathcal{C}(P_2) = \mathcal{C}(P_1')+\mathcal{C}(P_2) = \mathcal{C}(P)$. Therefore by Lemma 3.7 we get $\mathcal{C}(P_1) = \mathcal{C}(P_1')$ and we get a contradiction.

Now assume that there are maps defined but there is no one-to-one map from P_1 to P. In this case there will be some $a_1, a_2 \in P_1$ such that $a_1 \neq a_2$ and $f(a_1) = f(a_2)$; that is $f(a_1) = a_1 + b_1 = a_2 + b_2 = f(a_2)$ for some $b_1, b_2 \in P_2$. Now we know that $p_1 = a_1 + b_2$ and $p_2 = a_2 + b_1$ are in $\mathcal{C}(P)$. $p_1 = p_2$ and $f(a_1) = f(a_2)$ imply that $a_1 = a_2$. This contradicts with our assumption, so $p_1 \neq p_2$. Moreover the midpoint of p_1 and p_2 is also in $\mathcal{C}(P)$. But $f(a_1) = \frac{1}{2}(p_1 + p_2)$, and this gives a contradiction because a midpoint of two distinct points of $\mathcal{C}(P)$ cannot be a vertex point of $\mathcal{C}(P)$. This finishes the proof.

Now by using this lemma we will give the proof of the main theorem of this section.

Proof. (Theorem 3.6)
Assume that $|\mathcal{V}_{i+1}| = |\mathcal{V}_i| = t$. We consider the following two equations :

$$\mathcal{V}_{j+1} = \mathcal{V}(\mathcal{V}_j + A^{-(j+1)}D) \tag{3.1}$$

$$\mathcal{V}_{j+1} = \mathcal{V}(A^{-1}\mathcal{V}_j + A^{-1}D). \tag{3.2}$$

Now by using Lemma 3.9 and Equation 3.1 we know that there exists a one-to-one function f_j from $\mathcal{V}_j$ to $\mathcal{V}_{j+1}$ for $j \geq 1$. Furthermore, f_i is bijection because $|\mathcal{V}_{i+1}| = |\mathcal{V}_i| = t$. Similarly, by using Lemma 3.9 and Equation 3.2, there exists a one-to-one function g_j from $A^{-1}\mathcal{V}_j$ to $\mathcal{V}_{j+1}$ for $j \geq 1$. Again g_i is a bijection because $|\mathcal{V}_{i+1}| = |A^{-1}\mathcal{V}_i| = t$ and therefore $g_i(A^{-1}\mathcal{V}_i) = \mathcal{V}_{i+1}$.

Let $p = a_1a_2 \dots a_i \in \mathcal{V}_i$

$$f_i(p) = f_i(a_1a_2 \dots a_i) = a_1a_2 \dots a_iy$$

for some $y \in D$. Similarly $A^{-1}p = 0a_1a_2 \dots a_i \in A^{-1}\mathcal{V}_i$, and

$$g_i(A^{-1}p) = g_i(0a_1a_2 \dots a_i) = xa_1a_2 \dots a_i$$

for some $x \in D$. Moreover g_i^{-1} is a bijective function from $\mathcal{V}_{i+1}$ to $A^{-1}\mathcal{V}_i$.

$$g_i^{-1}(xa_1a_2 \dots a_i) = 0a_1a_2 \dots a_i$$

Now we define the bijective function $h_i = f_i A g_i^{-1}$ from $\mathcal{V}_{i+1}$ to $\mathcal{V}_{i+1}$.

$$h_i(xa_1a_2 \dots a_i) = (a_1a_2 \dots a_iy)$$

for some $x, y \in D$, $xa_1a_2 \dots a_i \in \mathcal{V}_{i+1}$ and $a_1a_2 \dots a_iy \in \mathcal{V}_{i+1}$.

Now take any element c_0 from $\mathcal{V}_{i+1}$.

$$c_0 = a_1a_2 \dots a_{i+1}$$

Then set $c_1 = h_i(c_0)$ and $c_k = h_i(c_{k-1})$ such that,

$$c_1 = a_2a_3 \dots a_ia_{i+1}\beta_1.$$

$$c_2 = a_3a_4 \dots a_{i+1}\beta_1\beta_2.$$

$$\dots$$

If we continue this process we will get elements $c_0, c_1, \dots \in \mathcal{V}_{i+1}$ until $c_k = c_t$ for some k and $0 \leq t < k$. We claim that $t = 0$; that is, this process ends when $c_k = c_0$ for some $k > 0$. To prove this claim, assume that k is the smallest number with the property $c_k = c_t$ where $0 \leq t < k$. If $t = 0$ we are done, otherwise because h_i is bijective we get $c_{k-1} = h_i^{-1}(c_k) = h_i^{-1}(c_t) = c_{t-1}$, but this contradicts with the assumption that k is the smallest number satisfying $c_k = c_t$. This finishes the proof of the claim.

We have thus constructed the $(i+1)$-strings

$$c_0 = a_1a_2 \dots a_{i-1}a_ia_{i+1},$$

$$c_1 = a_2a_3 \dots a_ia_{i+1}\beta_1,$$

$$c_2 = a_3a_4 \dots a_{i+1}\beta_1\beta_2,$$

$$\dots$$

$$c_{k-1} = \beta_{k-i-1} \dots \beta_{k-1},$$

$$c_k = \beta_{k-i} \dots \beta_k.$$

From $c_0 = c_k$ we get $a_1 = \beta_{k-i},\ a_2 = \beta_{k-i+1},\ \ldots,\ a_{i+1} = \beta_k$. Now we find a recurring sequence Y_0 such that its $(i+1)$-strings are $c_0, c_1, \ldots, c_k$ and Y_0 is the following recurring sequence

$$Y_0 = (\overline{a_1, a_2, \ldots, a_{i+1}, \beta_1, \ldots, \beta_{k-i-1}}).$$

Note that i-strings of recurring sequence Y_0 are elements of $\mathcal{V}_i$. Also note that the numbering above done with the assumption that $k > i$. It is easy to see that if $k \leq i$ then $Y_0 = \overline{(a_1, a_2, \ldots, a_k)}$.

We associate k elements of set $\mathcal{V}_{i+1}$ with a recurring sequence Y_0. Suppose that there is an element in $\mathcal{V}_{i+1}$ which is not an $(i+1)$-string of Y_0, call this element c_0'. Then we repeat the above construction for c_0'. We get a sequence Y_1. If there is another element in $\mathcal{V}_{i+1}$ which is not an $(i+1)$-string of Y_0 and Y_1, repeat the above procedure for this element and get another sequence Y_2. Proceeding in this way we can exhaust the elements of $\mathcal{V}_{i+1}$. Thus we'll have distinct recurring sequences $Y_0, Y_1, \ldots, Y_m$ such that each element of $\mathcal{V}_{i+1}$ is an $(i+1)$-string of a recurring sequence Y_l for $0 \leq l \leq m$. Also each $(i+1)$-string of Y_l is an element of $\mathcal{V}_{i+1}$. The same relation is also valid for the elements of $\mathcal{V}_i$. Note that each element c of $\mathcal{V}_{i+1}$ can be a $(i+1)$-string of one and only one Y_l.(Because of the unique way of construction of Y_l from c).

Our aim for the next step is to show that for any $j > i$, $\mathcal{V}_{j+1}$ has also t elements. In order to show this we will show that f_j is bijection. We use induction and as the induction hypothesis we have f_{j-1} is bijection. Let $p = a_1 a_2 \ldots a_j a_{j+1} \in \mathcal{V}_{j+1}$. We know that $f_j(a_1 a_2 \ldots a_j) = a_1 a_2 \ldots a_j y \in \mathcal{V}_{j+1}$ for some $y \in D$. Then $p_1 = a_2 \ldots a_j a_{j+1} \in \mathcal{V}_j$ and $p_2 = a_2 \ldots a_j y \in \mathcal{V}_j$. But then ${f_{j-1}}^{-1}(p_1) = a_2 \ldots a_j = {f_{j-1}}^{-1}(p_2)$. Because ${f_{j-1}}^{-1}$ is bijection we get $p_1 = p_2$ and $y = a_{j+1}$. We get $f_j(a_1 a_2 \ldots a_j) = p$ and therefore f_j is surjection. Also we already know that f_j is a one-to-one map by Lemma 3.9. Therefore f_j is a bijection and $\mathcal{V}_{j+1}$ has also t elements.

Now for any $j \geq i$ we get $|\mathcal{V}_{j+1}| = |\mathcal{V}_j| = t$, therefore we can use the construction method above inductively. Now take any element p from $\mathcal{V}_{i+2}$. By using the construction method for $Y_0, Y_1, \ldots, Y_m$ above, we can obtain a recurring sequence

$\widetilde{Y}_0$ such that $(i+2)$-strings of $\widetilde{Y}_0$ are elements of $\mathcal{V}_{i+2}$ and $(i+1)$-strings of $\widetilde{Y}_0$ are elements of $\mathcal{V}_{i+1}$. Because this sequence is constructed in a unique way we conclude that $\widetilde{Y}_0 = Y_l$ for a unique $l \in \{0, \ldots, m\}$. So the sets of recurring sequences $\{Y_0, Y_1, \ldots, Y_m\}$, $\{\widetilde{Y}_0, \widetilde{Y}_1, \ldots, \widetilde{Y}_m\}$ whose $(i+1)$-strings and $(i+2)$-strings respectively constitute $\mathcal{V}_{i+1}$ and $\mathcal{V}_{i+2}$ are equal since $|\mathcal{V}_{i+1}| = |\mathcal{V}_{i+2}| = t$.

Finally, we will construct the convex hull $\mathcal{C}(T)$ of T. Let $\mathcal{V}'_T$ be the set of ∞-strings of Y_l $(0 \leq l \leq m)$. Because $\mathcal{V}'_T \subseteq T$, we get $\mathcal{C}(\mathcal{V}'_T) \subseteq \mathcal{C}(T)$. We now assume that $\mathcal{C}(\mathcal{V}'_T) \neq \mathcal{C}(T)$ and get a contradiction. If $\mathcal{C}(\mathcal{V}'_T) \neq \mathcal{C}(T)$ then there exists $\sum_{i=1}^{\infty} A^{-i} d_{j_i} \in T$ which is not in $\mathcal{C}(\mathcal{V}'_T)$. Then there exists $k_1 \in \mathbb{N}$ such that $\sum_{i=1}^{k} A^{-i} d_{j_i} \notin \mathcal{C}(\mathcal{V}'_T)$ for $k \geq k_1$. Let

$$a_k = \sum_{i=1}^{k} A^{-i} d_{j_i}, \qquad d_{j_i} \in D.$$

Hence there exists $\varepsilon > 0$ such that $d(a_k, \mathcal{C}(\mathcal{V}'_T)) > \varepsilon$ for all $k \geq k_1$. Now we know that $a_k \in A_k$ therefore and $a_k \in \mathcal{C}(\mathcal{V}_k)$. Because $a_k \in \mathcal{C}(\mathcal{V}_k)$, it can be written in the form

$$a_k = \sum_{j=1}^{t} \lambda_j^k v_j$$

for $v_j \in \mathcal{V}_k$ and $\sum_{j=1}^{t} \lambda_j^k = 1,\ \lambda_j^k \geq 0$. Also there exists $k_2 > k_1$ such that for each $v'_j \in \mathcal{V}'_T$ there exists $v_j \in \mathcal{V}_k$ with

$$d(v_j, v'_j) \leq \frac{\varepsilon}{2}$$

for $k \geq k_2$. Then

$$d(a_k, \sum_{j=1}^{t} \lambda_j^k v'_j) = d(\sum_{j=1}^{t} \lambda_j^k v_j, \sum_{j=1}^{t} \lambda_j^k v'_j) \leq \sum_{j=1}^{t} \lambda_j^k |v_j - v'_j| \leq \sum_{j=1}^{t} \lambda_j^k \frac{\varepsilon}{2} = \frac{\varepsilon}{2}.$$

But $\sum_{j=1}^{t} \lambda_j^k v'_j \in \mathcal{C}(\mathcal{V}'_T)$, and therefore $d(a_k, \mathcal{C}(\mathcal{V}'_T)) \leq \frac{\varepsilon}{2}$. We get the desired contradiction. So we conclude that $\mathcal{C}(\mathcal{V}'_T)$ is convex hull of T.

We need to show that this polytope has actually t vertices. That is, equivalent to showing that any two ∞-strings are not equal. Let two ∞-strings $\overline{a_1 a_2 \ldots a_n}$ and $\overline{b_1 b_2 \ldots b_m}$ be equal. Then

$$\sum_{k=0}^{\infty} A^{-mnk}(a_1 \ldots a_{mn}) = \sum_{k=0}^{\infty} A^{-mnk}(b_1 \ldots b_{mn})$$

and because both m and n divide mn, we get

$$\left(\sum_{k=0}^{\infty} A^{-mnk}\right)(a_1 \dots a_{mn}) = \left(\sum_{k=0}^{\infty} A^{-mnk}\right)(b_1 \dots b_{mn})$$

We know that $\sum_{k=0}^{\infty} A^{-mnk} = (I - A^{mn})^{-1}$ and because all the eigenvalues of A^{-mn} have moduli < 1, $det(I - A^{mn}) \neq 0$ and therefore $(a_1 \dots a_{mn}) = (b_1 \dots b_{mn})$. But this contradicts with the fact that $\mathcal{V}_{mn}$ has t distinct vertices, which we already showed above. Therefore there are t distinct ∞-strings, which are corner points of the convex hull of T.

4. COMPUTER IMPLEMENTATION AND EXAMPLES

4.1. Deciding Disconnectedness or ε-connectedness

In Chapter 2, we have proved a theorem which implies that the disconnectedness of a self-affine set can be decided in finite steps if it is not ε-connected for a given $\varepsilon > 0$. The algorithm we present below is based on our findings in Chapter 2. The algorithm needs a bounding set B for the self-affine set as in the proof of Proposition 2.8. We approximate to the convex hull and use this approximation as a bounding set for the self-affine set. The following algorithm summarizes the main steps of the method given previously.

Algorithm 4.1. - Inputs : A, D, p as a point, ε as distance

- Calculate D'' as defined in Chapter 2

- Calculate an approximation to convex hull of T and use it as a bounding set B.

- For each $d_{i,j} \in D''$ Begin Loop

    - Calculate distance betweeen $d_{i,j}$ and T'' as defined in Chapter 2 using B.

        - if they are disjoint then set the $f_{i,j}$ to 0.

        - if they are closer then epsilon set the $f_{i,j}$ to 1.

- End of Loop

- Use matrix F to decide the disconnectedness or ε-connectedness of T.

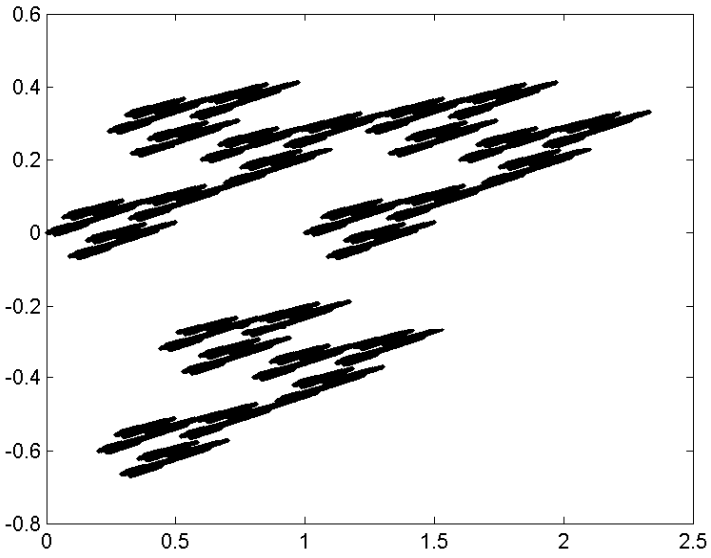

Figure 4.1: A Visually Apparent Disconnected Attractor

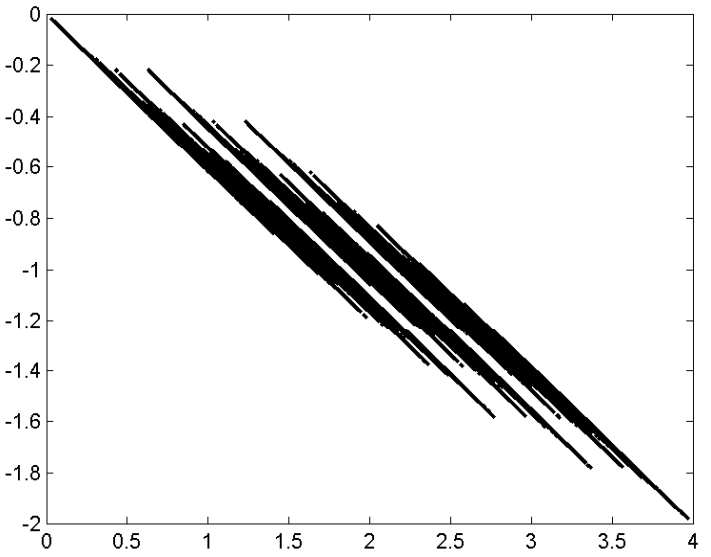

Figure 4.2: A Disconnected Attractor

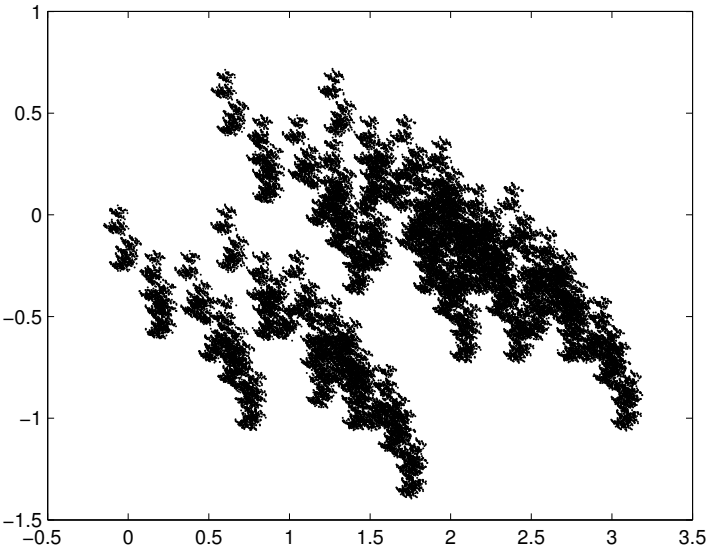

Figure 4.3: Another Disconnected Attractor

The method described can be used to show the disconnectedness of a given self-affine set whose disconnectedness can be seen clearly by visual observation like the ones in Figure 4.1 and Figure 4.2. The attractor $T(A, D)$ is obtained from the matrix $A = \begin{bmatrix} 2 & -1 \\ 1 & -3 \end{bmatrix}$ and the digit set $D = \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \end{bmatrix} \right\}$. Figure 4.2 is the attractor obtained from $A = \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}$ and $D = \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ 0 \end{bmatrix} \right\}$.

It is difficult to make a comment about the connectedness of some self-affine sets. In such cases, the method can be used to show that the self-affine set is ε-connected for given small ε. For example, Figure 4.3 is disconnected and Figure 4.4 is ε-connected for $\varepsilon > \frac{1}{10000}$. Figure 4.3 is the attractor obtained from $A = \begin{bmatrix} 1 & -1 \\ 1 & 2 \end{bmatrix}$ and $D = \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \begin{bmatrix} 0 \\ 2 \end{bmatrix} \right\}$. Figure 4.4 is the attractor obtained from $A = \begin{bmatrix} 2 & 2 \\ 1 & -1 \end{bmatrix}$ and $D = \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \begin{bmatrix} 4 \\ 0 \end{bmatrix} \right\}$.

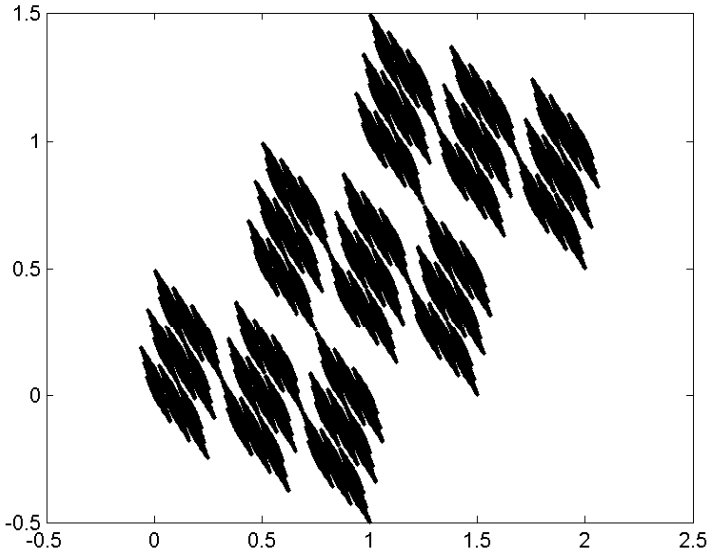

Figure 4.4: An ε-connected Attractor for $\varepsilon > \frac{1}{10000}$

4.2. Finding the Convex Hull

In this part, we give examples of self-affine sets whose convex hulls can be determined by the method given in Chapter 3.

For $A = \begin{bmatrix} 0 & -3 \\ 1 & 0 \end{bmatrix}$, $D = \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right\}$, the convex hull of $T(A, D)$ is an 8-gon as depicted in Figure 4.5.

For $A = \begin{bmatrix} -2 & -4 \\ 1 & 0 \end{bmatrix}$, $D = \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right\}$, the convex hull of $T(A, D)$ is an 6-gon as depicted in Figure 4.6.

For $A = \begin{bmatrix} -2 & -4 \\ 1 & 0 \end{bmatrix}$, $D = \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 3 \\ 1 \end{bmatrix} \right\}$, the convex hull of $T(A, D)$ is an 9-gon as depicted in Figure 4.7.

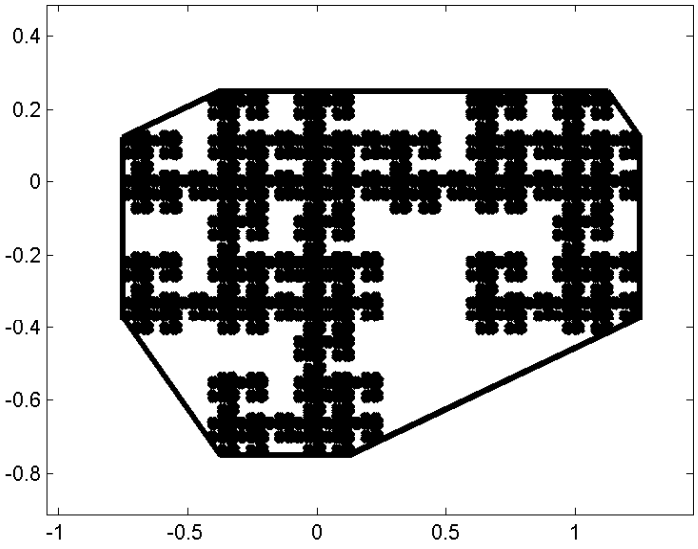

Figure 4.5: An Attractor with 8-gon Convex Hull

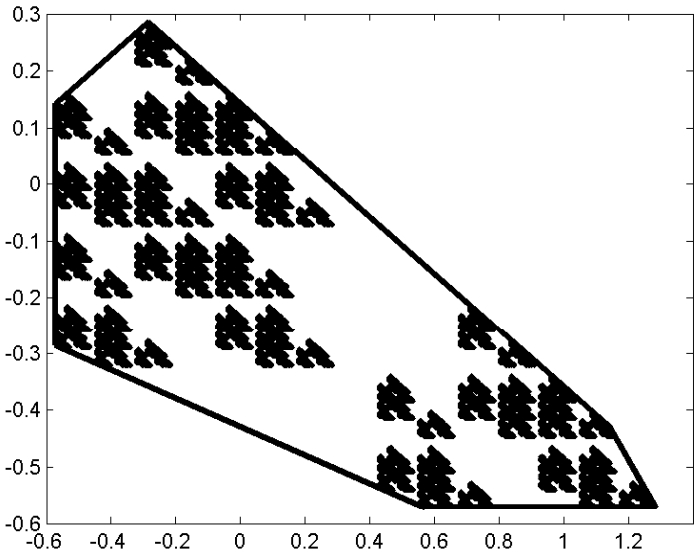

Figure 4.6: An Attractor with 6-gon Convex Hull

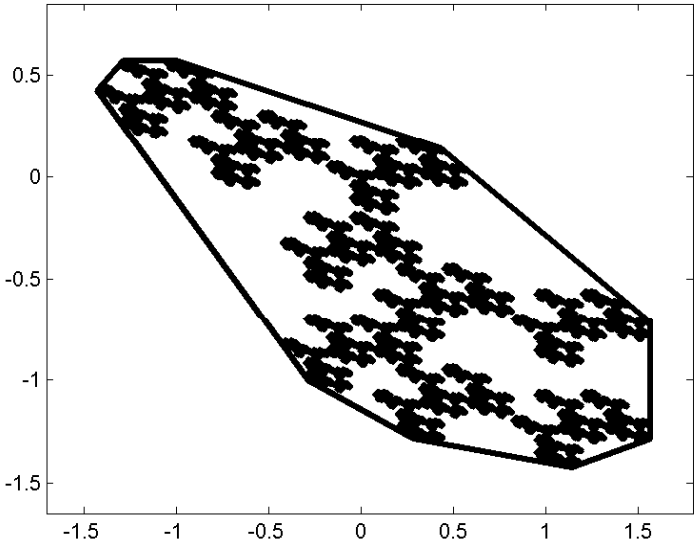

Figure 4.7: An Attractor with 9-gon Convex Hull

4.3. Routines Used in Computer Implementation

Function: findConvexHull

Description: Tries to find the convex hull of a given self affine set, using the method defined in Chapter 3.

Inputs: Required inputs are listed below.

1) A: An expanding 2x2 matrix. Must be in symbolic data type.

2) D: The digit set in $\mathbb{R}^2$. Must be in symbolic data type.

3) trunc: If the convex hull is not found by the condition given in the work then the algorithm stops when the number of the vertices exceeds this number 'trunc'.

Output: If convex hull is found then a message is given and the convex hull is drawn.

Function: isDisconnected

Description: Decides whether the given self affine set is disconnected or it is epsilon-connected using the method defined in Chapter 2.

Input: Required inputs are listed below.

1) A : An expanding 2x2 matrix. Must be in symbolic data type.

2) D : The digit set in $\mathbb{R}^2$. Must be in symbolic data type.

3) epsilon: The algorithm tries to decide the disconnectedness of the self-affine set. The algorithm stops if it cannot decide and the set is epsilon-connected.

4) step-limit: This parameter is for practical usage. User can set a step limit, which means that algorithm cancels after exceeding this limit without properly finishing the algorithm.

Output: Three different messages given. The method decides Disconnectedness or epsilon-connectedness. The third message is user cancelation by reaching the step-limit.

Function: distBetweenPointAndAttractor

Description: Calculates the distance between a given point and the given attractor.

Inputs: Required inputs are listed below.

1) A: An expanding 2x2 matrix. Must be in symbolic data type.

2) D : The digit set in $\mathbb{R}^2$. Must be in symbolic data type.

3) epsilon: The algorithm tries to decide the point and the self-affine set are disjoint. The algorithm stops if they are closer than epsilon.

4) step-limit: This parameter is for practical usage. User can set a step limit, which means that algorithm cancels after exceeding this limit without properly finishing the algorithm.

5) cHull: A boundary needed for the algorithm.

6) cHull-err: A real number which defines the maximum error of the given boundary.

Output: The following three different values may return.

1) 0 : Point and attractor are disjoint.

2) 1 : Point and attractor are closer than epsilon.

3) -1 : User cancelation because of exceeding step limit.

Function: approxConvexHull

Description: Approximates the convex hull of a given self affine set.

Input:

1) A : An expanding 2x2 matrix. Must be in symbolic data type.

2) D : The digit set in $\mathbb{R}^2$. Must be in symbolic data type.

3) maxNumOfSides : If the approximate convex hull has more vertices than 'maxNumOfSides' then algorithm stops and returns the polygon as an approximation.

4) epsilon : If the approximate convex hull and the actual convex hull are closer than epsilon, then algorithm stops and returns the polygon as an approximation.

Output:

1) eps : The radius of the error. That is, the distance between any point of the actual convex hull and approximate convex hull is less than eps.

2) chull : The approximate convex hull.

4.4. Notes about the Source Codes

All source codes, presented in the Appendix, are written in either MATLAB or Maple. We use symbolic calculation data types in order to avoid possible truncation errors. The data types of the parameters of the routines and the local variables are mostly symbolic data types. Some sample outputs for the routines are given below.

Sample call of the routine "isDisconnected":

```
A=[ 0  -3;
    1  0 ];
D1=[ 0 0  ;
     0 1  ] ;
isDisconnected (sym(A),sym(D1), sym(0.1) , 175 )
```

Output of the sample call of "isDisconnected":

```
     1     0
     0     1
This attractor is disconnected.
```

Note: The matrix given in the output contains zero entries. Therefore, according to Theorem 2.9, the program concludes that the attractor is disconnected.

Sample call of the routine "distBetweenPointAndAttractor":

```
A=[1 -1;
   1  2];
D1=[0 -1 2 ;
    0  1 1] ;
p=[0.32;-0.13] ;
distBetweenPointAndAttractor (sym(A),sym(D1),
    sym(0.1) , sym(p) ,10000, cH,cH_err )
```

Output of the sample call of "distBetweenPointAndAttractor":

```
     0
```

Note: Recall that the output value 0 means that the point and attractor are disjoint, as explained before.

Sample call of the routine "approxConvexHull":

```
A =[ 2, 0;
     1, 3]
D1 =[ 0, 0, 0, 1, 1, 1;
      0, -1, -2, 0, -1, -2]
```

```
[eps,chull]=approxConvexHull(sym(A),sym(D1),10,sym(0.0003))
```

Output of the sample call of "approxConvexHull":

```
eps =
    0.0761
chull =
    0.9688    0.9688    0.9375    0.8750    0.7500    0.5000
   -1.4667   -0.4708   -0.4437   -0.3935   -0.3056   -0.1667


         0         0    0.0313    0.0938    0.2188    0.4688
         0   -0.9959   -1.0230   -1.0732   -1.1611   -1.3000
```

Note: The output chull gives the vertices of the approximate convex hull, and eps gives the maximum approximation error. Each of the 12 columns above corresponds to the coordinates of a vertex in the plane.

Sample call of the routine "distPointAndPolygonSym":

```
V=sym([0 1 20 0 ; 0 0 20 1]) ; x= 19; y= 0.1 ;
double(distPointAndPolygonSym([sym(x);sym(y)],V) )
```

Output of the sample call of "distPointAndPolygonSym":

```
ans =
   12.9811
```

Note: The output gives the distance between the given point and the given polygon.

BIBLIOGRAPHY

[1] **Hutchinson, J. E.**, 1981. Fractals and self-similarity, *Indiana Uni. Math. J.*, **30**, 713–747.

[2] **Grochenig, K. and Madych, W.R.**, 1992. Multiresolution analysis, haar bases, and self similar tilings of $\mathbb{R}^n$, *IEEE Transactions on Information Theory*, **38**, 556–568.

[3] **Falconer, K.J.**, 1990. Fractal geometry: Mathematical Foundations and Applications, John Wiley, New York.

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

[5] **Ghrist, R.W., Holmes, P.J. and Sullivan, M.C.**, 1997. Knots and Links in Three-Dimensional Flows, volume 1654 of Springer Lecture Notes in Math., Springer-Verlag, Berlin.

[6] **Mindlin, G. and Gilmore, R.**, 1992. Topological analysis and synthesis of chaotic time series, *Physica D*, **58**, 229–242.

[7] **Tufillaro, N.B., Wyckoff, P., Brown, R., Schreiber, T. and T. Molteno**, 1995. Topological time-series analysis of a string experiment and its synchronized model, *Phys. Rev. E*, **51(1)**, 164–174.

[8] **Dey, T.K., Edelsbrunner, H. and Guha, S.**, 1999. Computational topology, advances in discrete and computational geometry, eds.: Chazelle, Goodman and Pollack, *American Mathematical Society*, **223**, 109–143.

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

[10] **Bandt,C. and Wang Y.**, 2001. Disk-like self-affine tiles in $\mathbb{R}^2$, *Discrete and Computational Geometry*, **26(4)**, 591–601.

[11] **Leung, K.S. and Lau, K.S.**, 2007. Disklikeness of planar self-affine tiles, *Trans. Amer. Math. Soc.*, **359**, 3337–3355.

[12] **Robins, V., Meiss, J.D. and Bradley E.**, 1998. Computing connectedness: An exercise in computational topology, *Nonlinearity*, **11**, 913–922.

[13] **Robins, V., Meiss, J.D. and Bradley E.**, 2000. Computing connectedness: Disconnectedness and discreteness, *Physica D*, **139**, 276–300.

[14] **Strichartz, R. and Wang, Y.**, 1999. Geometry of self-affine tiles I., *Indiana Univ. Math. J.*, **48**, 1–23.

[15] **Kenyon, R., Li, J., Strichartz,R. and Wang, Y.**, 1999. Geometry of self-affine tiles II., *Indiana Univ. Math. J.*, **48**, 25–42.

[16] **Hocking, J.G. and Young, G.S**, 1961. Topology, Addison-Wesley, Massachusetts.

[17] **Webster, R.**, 1994. Convexity, Oxford University Press, New York.

APPENDIX : SOURCE CODES

```
%************************************************************************
% function: isDisconnected
%************************************************************************
function v = isDisconnected (A,D, epsilon ,step_limit )
format long g; format compact ; v=[] ;
invA=inv(A) ; [m1,n1]=size (A); [m2,n2]=size (D);
if ( (m1 ~= n1 ) | ((n1 ~= m2 ) & (n1 ~= 1) ) )
    disp('INPUT ERROR') ;
    disp('Matrix must be a square matrix,') ;
    disp('Matrix col. dimension must be same with row dimension of ...
    digit set') ;
    disp('Each col. vector in digitSet is a distinct element of ...
    digitSet ') ;
    return ;
end
N=zeros(n2,n2) ; D_dbl_prime = invA*setAdd(D , -1*D) ; max_dist = 0;
point_cache = [] ; stepLimitFlg = false ;
% First step is to find a boundary. We use an approx. to convex hull
%   as a boundary.
[cHull_err,cHull]=approxConvexHull(A,D_dbl_prime,50,epsilon/2) ;
for i=1:n2
    for j=i:n2
        d_ij = invA*(D(:,i) - D(:,j)) ;
        if (i~=j)
            % Because the operation function distBetweenPointAndAttractor
            % is costly we cache it.
            cache_hit = false ;
            for k=point_cache
                if ( (k(1)==d_ij(1)) & (k(2)==d_ij(2)) )
                    % A Cache Hit occurs
                    cache_hit = true ;
                    % Note k(3) and k(4) stores indexes of cache value
                    N(i,j) = N (int8(k(3)) , int8(k(4)) ) ;
                    N(j,i) = N(i,j) ;
                    break ;  % break the loop for k=point_cache
                end
            end
            if (~cache_hit)
                ret = distBetweenPointAndAttractor (A,D_dbl_prime, ...
                      epsilon , d_ij ,step_limit, cHull, cHull_err  );
                if (ret==0)
                    % Point is disjoint
                    N(i,j) = 0 ;  N(j,i) = 0 ;
```

```
                else
                 if (ret== -1)
                    % this case is STEP_LIMIT_REACHED
                    stepLimitFlg = true ;
                    N(i,j) = 1 ;   N(j,i) = 1 ;
                 else
                    % POINT is CLOSER_THAN_EPSILON
                    N(i,j) = 1 ;  N(j,i) = 1 ;
                 end
                end
                % Insert the result to cache
                point_cache=[point_cache [d_ij ; i ; j ; ret] ] ;
            end
        else
            % if i==j then Ti=Tj so they intersect
            N(i,j) = 1 ;    N(j,i) = 1 ;
        end
    end
end
disp(N); N_m = N^n2 ; col_mins = [] ;
for i=1:n2
    col_mins = [col_mins min(N_m(:,i)) ] ;
end
mi = min(col_mins);
if (mi==0)
    disp(['This attractor is disconnected']) ;
else
  if (stepLimitFlg)
    disp('Cannot decide because the Calculation Limit is reached.') ;
  else
    eps_max=maple('maxRowValue',point_cache(5,:)) ;
    disp(['This attractor is e-connected with an e=epsilon at least'...
           num2str(double(eps_max))]) ;
  end
end end

%*********************************************************************
% Function: findConvexHull
%*********************************************************************
function v = findConvexHull (A,D,trunc)
invA=inv(A) ;
len_chull = 0 ; [rowSizeD,colSizeD]=size(D) ;
sum_seq =-1*ones(1,colSizeD);
maxA=0; d=sym([0;0]) ; d_2=sym([0;0]) ; prevSize = 0 ; currSize = 0;
  for (t=1:1000)   % This never stops for 1000 loop
     d=setAdd(d , invA^t*D ) ;
     %k=convhull(d(1,:)',d(2,:)')';
     k=maple('convexHull2D',d) ;
     k=double(k);
     d=d(:,k);
     di1=ceil(k/colSizeD);
```

```
    di2=mod(k,colSizeD);
    sum_seq = [sum_seq(:,di1) ; di2] ;
    prevSize=currSize ;
    [m1,currSize]=size(sum_seq) ;
    if ((prevSize==currSize) | (currSize>trunc))
        break ;
    end
  end
if (prevSize==currSize)
  disp(['The convex hull is a Polygon. It is ' num2str(prevSize) ...
 '-gon'] ) ;
  line([double(d(1,:)) double(d(1,1))],[double(d(2,:)) ...
    double(d(2,1))],'Color','r');
else
  disp(['Cannot find Convex hull.'] ) ;
end
end

%************************************************************************
% function: approxConvexHull
%************************************************************************
function [eps chull] = approxConvexHull (A,D,maxNumOfSides,epsilon)
invA=inv(A) ; len_chull = 0 ; [rowSizeD,colSizeD]=size(D) ;
sum_seq = -1*ones(1,colSizeD);
maxA=0; d=sym([0;0]) ; d_2=sym([0;0]) ; currSize = 0 ; t = 1 ;
  while (true) % never stops until a break comes
    invA_t=invA^t ;
    d=setAdd(d , invA_t*D ) ;
    %k=convhull(d(1,:)',d(2,:)')';
    k=maple('convexHull2D',d) ;
    k=double(k);
    d=d(:,k);
    di1=ceil(k/colSizeD);
    di2=mod(k,colSizeD);
    sum_seq = [sum_seq(:,di1) ; di2] ;
    [m1,currSize]=size(sum_seq) ;
    d_norm = sqrt(maple('maxRowValue',d(1,:).^2 + d(2,:).^2) );
    % Let B=A^-k then B*d1 + B^2*d2 + B^3*d2  can have norm at most:
    % |d||B|/(1-|B|) where |B| is spectral norm of B
    normInvAk=  sqrt(maple('maxRowValue',(eig(invA_t'*invA_t))')) ;
    if int8(maple('isGT',1,normInvAk))
       eps = d_norm * normInvAk / (1-normInvAk) ;
       if int8(maple('isGEQ',epsilon,eps))
       % End of Search condition satisfied
           break ;
       end
    end
    if ((currSize>maxNumOfSides))
    % Other End of Search condition satisfied
        break ;
    end
```

```
     t = t + 1 ;
  end
chull=d;
end

%************************************************************************
% function: distBetweenPointAndAttractor
%************************************************************************
function ret = distBetweenPointAndAttractor (A,D,epsilon , point
,step_limit, cHull, cHull_err  )
POINT_DISJOINT = 0 ;
POINT_CLOSER_THAN_EPSILON = 1 ;
STEP_LIMIT_REACHED = 2 ;
D=elimiteSameColsAndSort(D) ;
invA = inv(A);
format long g;
format compact ;
v=[] ; steps=0 ;
code= -9999 ;
 epsilon_square = epsilon^2 ;
min_dist_found = sym(0) ;
[m1,n1]=size (invA);
[m2,n2]=size (D);
if ( (m1 ~= n1 ) | ((n1 ~= m2 ) & (n1 ~= 1) ) )
    disp('INPUT ERROR') ;
    disp('Matrix must be a square matrix,') ;
    disp('Matrix col. dimension must be same with row dimension ...
         of digit set') ;
    disp('Each col. vector in digitSet D is a distinct element ...
         of digitSet D') ;
    return ;
end
k=1 ;
Ak= 0*[1:m2]'  ;  % assign it to zero vector.
Ak_next = Ak ;
while (true)
   invA_k = invA^k ;
   invA_k_cHull = invA_k * cHull ;
   % prepare the points of Ak to be checked.
   Ak = setAdd(Ak_next,invA_k*D) ;
   Ak_next = [] ;
   Ak=elimiteSameColsAndSort(Ak) ;
   [m,n] =  size(Ak) ;
   % We know that
   % WJ(cHull) = Ak + invA^k*cHull
   % Where + is the set addition.
   % So we check whether point is in the c. hull (Ak +invA^k*cHull)
   % If not we remove point a_j from Ak. Because no need to check it
   % in the next steps.
    for (j=1:n)
        a_j=Ak(:,j) ;
```

```
        dist_to_cHull = distPointAndPolygonSym(point-a_j,invA_k_cHull);
        % Following is because cHull is cHull_err bigger than cHull.
        dist_to_cHull = dist_to_cHull - cHull_err ;
        % if dist_to_cHull is negative then point is in the cHull.
        % if dist_to_cHull is positive then point is out of the cHull,
        %   then ignore it for next level searches.
        if int8(maple('isGEQ',0,dist_to_cHull) )
            Ak_next = [ Ak_next a_j] ;
            tmp=(point-a_j);
            dist_square = sum(tmp.*tmp) ;
disp(['level:' num2str(k) ' , dist to point:'
num2str(double(sqrt(dist_square)))...
      ' , dist_to_cHull: ' num2str(double(sqrt(dist_to_cHull))) ]) ;
            % If distance between point and a_j is less than epsilon
            %    then we hit to epsilon constraint.
            if (int8(maple('isGT',epsilon_square,dist_square)))
                code = POINT_CLOSER_THAN_EPSILON  ;
                min_dist_found = sqrt(dist_square) ;
                break ;
            end
        end
    end
   steps= steps+n ;
   [m1,n1]=size(Ak_next) ;
    % If there is no element in the list then point is out of all sets.
    if (n1==0 )
        code = POINT_DISJOINT ;
        break ;
    end
    if (code == POINT_CLOSER_THAN_EPSILON)
        break ;
    end
    % If step limit is reached then it will take too much
    % for a practical calculation.
    if (steps>step_limit)
        code = STEP_LIMIT_REACHED;
        break ;
    end
    k=k+1 ;
end
if (code==POINT_DISJOINT)
    ret=0;
end if (code==POINT_CLOSER_THAN_EPSILON)
    ret = min_dist_found ;
end if (code==STEP_LIMIT_REACHED)
    ret = -1 ;
end
end
```

BIOGRAPHY

İlker Kocyigit was born in 1974 in Ankara. He attended Gazi Anatolian High School, Malatya Science High School and Ankara Deneme High School in chronological order. He received his B.Sc. degree in Computer Science from Middle East Technical University. He is currently enrolled to M.Sc. in Mathematics Program at İstanbul Technical University.