HUTCHINSON OPERATORS IN $\mathbb{R}^3$

CARLA M. RIGGI

Abstract. In this project we study the mathematics behind Hutchinson Operators in $\mathbb{R}^3$ with primary attention given to Iterated Function Systems. We have developed software necessary for defining and rendering the attractor of an IFS and studying its properties. To this end we present a contraction factor theorem for affine maps in $\mathbb{R}^n$ along with corollaries for special cases in $\mathbb{R}^3$. We examine the relationships between IFS attractors in $\mathbb{R}^3$ and their orthogonal projections onto the xy, yz, and xz planes; specifically the case where a 2d projection is itself the attractor of an IFS in $\mathbb{R}^2$. We also define the group of symmetry operations for the Sierpinski Triangle and the Sierpinski Tetrahedron. Our work is ongoing in the study of a family of 48^4 IFSs related to the Sierpinski Right Tetrahedron and the 6 symmetry groups of their attractors. Our goal is to count and classify all distinct attractors in this family; a task that requires a solution to what remains an open question in this field as far as we know. Finally, the chaos3d module is written for Maple in order to compute the contraction factors of affine maps, define various Hutchinson Operators in $\mathbb{R}^3$, and render their attractors.

1. *Introduction*

In my undergraduate Chaos and Fractals course at the University of Scranton we studied two-dimensional fractal structures generated using a type of Hutchinson Operator known as an *Iterated Function System (IFS)*. The Chaos module for Maple written by Ken Monks allowed us to render two-dimensional IFS attractors, but knowing that Hutchinson's IFS theory holds for $\mathbb{R}^n$, we were naturally curious to see a fractal in 3d. By generalizing the definitions and theorems applicable to Hutchinson Operators in $\mathbb{R}^2$ that were presented in the course, we can determine the formulas necessary for computing the affine maps used to describe fractals in three dimensions as well as a contraction factor for an n-dimensional affine map. These results allow us to write the Chaos3d Maple module for computing and displaying fractals in $\mathbb{R}^3$. This program will be a resource for future students in the Chaos and Fractals course and provides us with a tool for further investigation into the properties of 3d attractors having 2d fractal projections, and for studying the symmetry groups of IFS attractor families.

The paper is organized as follows. We first present the mathematical background and notation needed for our research. In the second section we present our main results. Here is found a contraction factor theorem for affine maps in $\mathbb{R}^n$ followed by corollaries for special cases in $\mathbb{R}^3$. We then go on to investigate some conditions on affine maps in an IFS that guarantee its $\mathbb{R}^3$ attractor to have a fractal projection into $\mathbb{R}^2$. Next we study the symmetry groups of two famous fractals and then investigate the classification of an IFS family based on the symmetry groups of their attractors. Peitgen, Jürgens, and Saupe

Date: April 15, 2001.

2000 *Mathematics Subject Classification.* 28A80.

Key words and phrases. fractals, IFS, Hutchinson operators, symmetry groups, computer graphics.

I would like to thank Dr. Monks without whose guidance and mathematical expertise I could not have completed this project.

[4] counted and classified all the distinct attractors produced by the family of 512 IFS's related to the Sierpinski right triangle. In this paper we study a similar family in $\mathbb{R}^3$: the family of 48^4 IFS's related to the Sierpinski right tetrahedron. All proofs of theorems and corollaries stated in our main results are presented in the following section. The appendices contain additional notation, Maple programming documentation, and data resulting from our research. In Appendix A we define our notation for the symmetry group of the cube and list sets of elements useful in defining IFS's having symmetric attractors including the center, and a few centralizer, and normalizer subgroups. In order to write the Chaos3d Maple module, it was first necessary to create the data structures with which to define affine maps in $\mathbb{R}^3$. These five affine data forms are defined in detail in Appendix B. Appendix C holds the Chaos3d source code itself and Appendix D contains help on using these procedures for uniting maps into Hutchinson operators and then rendering their attractors. Appendix E lists the IFS definitions for 8 distinct attractors related to the Sierpinski right tetrahedron that are each symmetric with respect to all six possible symmetry operations on this family of attractors. When using Maple to investigate properties of elements in our symmetry group of the cube, it is very convenient to maintain the notation we have defined. Therefore, an extension to Maple's group package is written by Ken Monks specifically for this symmetry notation and these procedures are listed in Appendix F.

2. *Mathematical Background and Notation*

A *set theoretic discrete dynamical system* is an ordered pair, (X, f), where X is a set and $f : X \to X$. For any $f : X \to X$ we define $f^0 = i_X$and $f^n = f \circ f^{n-1}$, for all $n \in \mathbb{N}^+$ where i_X is the *identity map* $i_X : X \to X$ given by $i_X(s) = s$, for all $s \in X$.

A *metric space* is a pair (X, d) where X is a set and $d : X \times X \to \mathbb{R}$ such that $\forall x, y, z \in X$, the following properties hold: $d(x, y) \geq 0$, $d(x, y) = 0 \Leftrightarrow x = y$, $d(x, y) = d(y, x)$, and $d(x, y) + d(y, z) \geq d(x, z)$. Here d is called a *metric* (or *distance function*) on X, and the elements of X are *points* in the metric space. An infinite sequence $x_0, x_1, x_2, ...$ of points in X is a *Cauchy sequence* if and only if $\forall \epsilon \in \mathbb{R}^+$ there exists an $N \in \mathbb{Z}^+$ such that $\forall i > N$ and $\forall j > N$, $d(x_i, x_j) < \epsilon$. A sequence $x_0, x_1, x_2, ...$ converges to $x \in X$ if and only if $\forall \epsilon \in \mathbb{R}^+$ there exists an $M \in \mathbb{N}$ such that $\forall m > M$, $d(x_m, x) < \epsilon$ (in this case we write $\lim_{m\to\infty} x_m = x$). A metric space is *complete* if and only if every Cauchy sequence in (X, d) converges to a limit $x \in X$. One family of complete metric spaces is $(\mathbb{R}^n, d_{Euc})$ where d_{Euc} is the *Euclidean metric* given by $d_{Euc}((x_1, x_2, ..., x_n), (y_1, y_2, ..., y_n)) = \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2}$. However, many of the definitions and theorems in this paper hold for any complete metric space.

Given a metric space (X, d), $f : X \to X$ we define the *scaling factor* of f to be any value $c \in \mathbb{R}^+$ such that $\forall x, y \in X$, $d(f(x), f(y)) \leqq cd(x, y)$ if such a value exists. We say f is a *contraction mapping* if and only if f has a scaling factor $c \in (0, 1)$. In this situation c is called a *contraction factor* of f. Given a set theoretic discrete dynamical system (X, f), the sequence $f^0(x_0), f^1(x_0), f^2(x_0), ...$ of iterations of f acting on a *seed* value, $x_0 \in X$, is called the *f-orbit* of x_0. For ease of notation, we will also sometimes write the f-orbit of x_0 as the sequence, $x_0, x_1, x_2, ...$ where $x_n = f(x_{n-1})$ for all $n \in \mathbb{N}^+$. The *Contraction Mapping Theorem* states that for any contraction mapping $f : X \to X$ on a complete metric space (X, d) with contraction factor c, (1) f has a unique fixed point, q, (2) $\forall\ x \in X$, $\lim_{n\to\infty} f^n(x) = q$ (i.e. the f-orbit of every $x \in X$ converges to the fixed point q), and (3) if $x_0, x_1, x_2, ...$ is the f-orbit of $x_0 \in X$ then $d(x_n, q) \leq \frac{c^n}{1-c} d(x_0, x_1)$.

Define $K_n = \{A \subseteq \mathbb{R}^n : A \text{ is compact}\}$. The *Hausdorff metric* d_H on K_n is defined as follows. Let $d_H : K_n \times K_n \to \mathbb{R}$ and $S, T \in K_n$. Define the *closed collar of radius* δ about

S to be the set $\overline{B}(S;\delta)$ consisting of all points contained in the union of closed balls of radius δ about all $\alpha \in S$. We define $d_H(S,T) = \inf\left\{\delta : S \subseteq \overline{B}(T;\delta) \text{ and } T \subseteq \overline{B}(S;\delta)\right\}$. It is known that (K_n, d_H) is a complete metric space.

Let $w_1, w_2, ..., w_k$ be contraction mappings on $\mathbb{R}^n$ with contraction factors $c_1, c_2, ..., c_k$ respectively. The *Hutchinson operator,* $W : K_n \to K_n$, associated with $w_1, w_2, ..., w_k$ is defined such that $\forall A \in K_n$,

$$W(A) = w_1(A) \cup w_2(A) \cup ... \cup w_k(A) \tag{1}$$

Hutchinson showed that such a map W is always a contraction map on (K_n, d_H) with contraction factor $c = \max\{c_1, c_2, ..., c_k\}$. Thus by the contraction mapping theorem, W has a unique fixed point and the W-orbit of any compact set converges to this fixed point. The unique fixed point of W is called the *attractor of* W and is denoted F_W. We will often abbreviate an operator of the form (1) by writing $W = w_1 \cup w_2 \cup ... \cup w_k$. We also will find it convenient to specify such an operator by its action on an arbitrary one point set, $\{v\}$, and further abbreviate $W(\{v\}) = w_1(\{v\}) \cup w_2(\{v\}) \cup ... \cup w_k(\{v\})$ as $W(v) = w_1(v) \cup w_2(v) \cup ... \cup w_k(v)$ as usual in this situation.

From the above definitions we see that

$$F_W = W(F_W) = w_1(F_W) \cup w_2(F_W) \cup ... \cup w_k(F_W)$$

Therefore, F_W is the unique compact set in K_n that is composed entirely of contracted w_i-copies of itself. A shape satisfying such a geometric condition is often called a *fractal*. For all $S \subseteq \mathbb{R}^n$, define $\mathrm{Sym}(S)$ to be the symmetry group of S.

Define the *sequence space on n letters*, Σ_n, to be the set of infinite sequences composed of natural numbers in 1 thru n, i.e. $\Sigma_n = \{s : s = s_1 s_2 ... \text{ and } s_1, s_2, ... \in \{1..n\}\}$. Define the function $\Phi_W : \Sigma_n \to \mathbb{R}^n$ by $\Phi_W(s) = \lim_{m\to\infty} w_{s_1} \circ w_{s_2} \circ ... \circ w_{s_m}\left(\overrightarrow{0_n}\right)$ where $\overrightarrow{0_n} = (0, 0, ..., 0) \in \mathbb{R}^n$. It is known that $F_W = \Phi_W(\Sigma_n)$. So $\forall a \in F_W$, $\exists r \in \Sigma_n$, $a = \Phi_W(r) = \lim_{m\to\infty} w_{r_1} \circ w_{r_2} \circ ... \circ w_{r_m}\left(\overrightarrow{0_n}\right)$. Such a value $r \in \Sigma_n$ is called an address of the point $a \in F_W$ and we say that a is the point corresponding to the address r.

In this paper we focus on Hutchinson operators of a special type known as an *Iterated Function System (IFS).* An IFS is a Hutchinson operator composed entirely of *affine* contraction maps [3]. A function $T : \mathbb{R}^n \to \mathbb{R}^n$ is an *affine map* (or *affine transformation*) if and only if $T(x) = Mx + B$ for some matrix $M \in M_n(\mathbb{R})$ and some $B \in \mathbb{R}^n$. Here B defines the translation of the origin under T, and the columns of M define the image of each basis vector in $\mathbb{R}^n$ under T before translation.

3. *Statement of the Main Results*

3.1. **Contraction factor for Affine Maps in $\mathbb{R}^n$.** A very basic question to ask regarding any operator of the form (1) is whether it is an IFS. Since an IFS can only be formed using affine maps which are also contraction maps, it is necessary to be able to determine if a given affine map is a contraction map. In this situation, knowing a contraction factor for such maps would allow us to determine the rate of convergence of the IFS as well. The most accurate convergence rate for an affine map is given by its smallest contraction factor. This value is the best possible contraction factor for that map. For affine maps in $\mathbb{R}^2$, the use of complex analysis makes this computation relatively simple, but naturally the question arises: How can we compute the best possible contraction factor for affine maps in other dimensions? These problems are solved by the following theorem.

Theorem 3.1. ***(Contraction Factor)*** *Let $T : \mathbb{R}^n \to \mathbb{R}^n$ by $T(x) = Mx + B$ where $M \in M_n(\mathbb{R})$. Let*
$c = \max\left\{\lambda : \lambda \text{ is an eigenvalue of } M^T M\right\}$. T is a contraction map iff $\sqrt{c} < 1$. If T is a contraction map then $\sqrt{c}$ is the best possible contraction factor.

As a special case we can derive specific formulas for the contraction factor of affine maps in $\mathbb{R}^3$.

Corollary 3.2. *Let $T : \mathbb{R}^3 \to \mathbb{R}^3$ be the affine map given by $T(x) = Mx + B$, where M is a diagonal matrix. Then a scaling factor for T is*

$$\max\{|m| : m \text{ is a value on the main diagonal of } M\}.$$

If $\max\{|m|\} < 1$ then T is a contraction mapping and this is the best possible contraction factor.

Corollary 3.3. *Let $T : \mathbb{R}^3 \to \mathbb{R}^3$ be the affine map given by $T(x) = Mx + B$ where $M^T M$ is diagonal. Then a scaling factor for T is*

$$\max\left\{\sqrt{\alpha} : \alpha \text{ is a value on the main diagonal of } M^T M\right\}.$$

If $\max\{\sqrt{\alpha}\} \in (0, 1)$ then T is a contraction mapping and this is the best possible contraction factor.

It is easy to show that $M^T M$ is diagonal if and only if M is a matrix that rotates and scales the unit cube such that T of the unit cube is still an orthogonal parallelepiped. A matrix, M, of this type can be defined according to *Geometric form* given by

$$M = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} r & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & t \end{bmatrix} \tag{2}$$

$$= \begin{bmatrix} r\cos\phi\cos\gamma & s\sin\theta\sin\phi\cos\gamma - s\cos\theta\sin\gamma & t\cos\theta\sin\phi\cos\gamma + t\sin\theta\sin\gamma \\ r\cos\phi\sin\gamma & s\sin\theta\sin\phi\sin\gamma + s\cos\theta\cos\gamma & t\cos\theta\sin\phi\sin\gamma - t\sin\theta\cos\gamma \\ -r\sin\phi & s\sin\theta\cos\phi & t\cos\theta\cos\phi \end{bmatrix}$$

This matrix first scales the $\vec{i}$, $\vec{j}$, and $\vec{k}$ vectors by real-valued constants r, s,and t respectively. The resulting orthogonal parallelepiped rotates about the x, y, and z axes consecutively by angles θ, ϕ,and γ respectively. When M is defined in this manner it is easy to determine the contraction values for its associated affine map according to the following corollary.

Corollary 3.4. *Let $T : \mathbb{R}^3 \to \mathbb{R}^3$ be the affine map given by $T(x) = Mx + B$, where M is a matrix in the form (2) with $r, s, t \in \mathbb{R}^+$. Then a scaling factor for T is*

$$\max\{r, s, t\}$$

If $\max\{r, s, t\} < 1$ then T is a contraction mapping and this is the best possible contraction factor.

3.2. **2d Projections of 3d IFS Attractors.** We can analyze much of the structure of a given 3d Euclidean geometric shape based upon the properties of its orthogonal projections. It is natural then to ask what a projection into $\mathbb{R}^2$ of a fractal in $\mathbb{R}^3$ can tell us about the structure of the whole and vice versa. Specifically, we ask, when might a 2-dimensional projection of some IFS attractor in $\mathbb{R}^3$ itself be the attractor of an IFS in $\mathbb{R}^2$? By examining the standard projections onto the xy, yz, and xz planes we find that a fractal projection onto one of these planes is guaranteed if every affine map in the IFS is in one of the forms stated in the following theorems.

Definition 3.5. *Define* $P_x : \mathbb{R}^3 \to \mathbb{R}^2$, $P_y : \mathbb{R}^3 \to \mathbb{R}^2$, *and* $P_z : \mathbb{R}^3 \to \mathbb{R}^2$ *by*

$$\begin{aligned} P_x(x,y,z) &= (y,z) \\ P_y(x,y,z) &= (x,z) \\ P_z(x,y,z) &= (x,y) \end{aligned}$$

for all $x, y, z \in \mathbb{R}$.

Theorem 3.6. *Let* $w : \mathbb{R}^3 \to \mathbb{R}^3$ *and* $v : \mathbb{R}^2 \to \mathbb{R}^2$ *be affine maps. Then* $P_z \circ w = v \circ P_z$ *if and only if there exists* $a, b, c, d, e, f, g, h, i \in \mathbb{R}$ *such that*

$$w(x,y,z) = \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ e & f & g \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} h \\ i \\ j \end{bmatrix}$$

and

$$v(x,y) = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} h \\ i \end{bmatrix}$$

for all $(x, y, z) \in \mathbb{R}^3$.

Theorem 3.7. *Let* $W = w_1 \cup \cdots \cup w_k$ *and* $V = v_1 \cup \cdots \cup v_k$ *be IFS's on* $\mathbb{R}^3$ *and* $\mathbb{R}^2$ *respectively such that* $P_z \circ w_i = v_i \circ P_z$ *for all* $i \in \{1, \ldots, k\}$. *Then*

$$P_z \circ \Phi_W = \Phi_V.$$

Theorem 3.8. *Let* $W = w_1 \cup \cdots \cup w_k$ *and* $V = v_1 \cup \cdots \cup v_k$ *be IFS's on* $\mathbb{R}^3$ *and* $\mathbb{R}^2$ *respectively such that* $P_z \circ w_i = v_i \circ P_z$ *for all* $i \in \{1, \ldots, k\}$. *Then*

$$P_z(F_W) = F_V.$$

Theorems similar to each of the three above hold when P_z is replaced by P_y and when w in Theorem 3.6 is replaced by

$$w(x,y,z) = \begin{bmatrix} a & 0 & b \\ c & d & e \\ f & 0 & g \end{bmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{bmatrix} h \\ i \\ j \end{bmatrix} \text{ and } v(x,z) = \begin{bmatrix} a & b \\ f & g \end{bmatrix} \begin{pmatrix} x \\ z \end{pmatrix} + \begin{bmatrix} h \\ j \end{bmatrix}$$

and again in the case where P_z is replaced by P_x and when w in Theorem 3.6 is replaced by

$$w(x,y,z) = \begin{bmatrix} a & b & c \\ 0 & d & e \\ 0 & f & g \end{bmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{bmatrix} h \\ i \\ j \end{bmatrix} \text{ and } v(x,y) = \begin{bmatrix} d & e \\ f & g \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{bmatrix} i \\ j \end{bmatrix}.$$

As a natural consequence of these theorems, we know that an IFS whose affine maps all have matrices of the form $\begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{bmatrix}$ for some $a, b, c \in \mathbb{R}$, will have a fractal projection onto each of the three planes defined according to the following corollary.

Corollary 3.9. *Let* $W = w_1 \cup \cdots \cup w_k$ *be an IFS on* $\mathbb{R}^3$ *and let* $V = v_1 \cup \cdots \cup v_k$, $U = u_1 \cup \cdots \cup u_k$, *and* $T = t_1 \cup \cdots \cup t_k$ *be IFS's on* $\mathbb{R}^2$ *such that for all* $i \in \{1, \ldots, k\}$

$$\begin{aligned} P_z \circ w_i &= v_i \circ P_z \\ P_x \circ w_i &= u_i \circ P_x \\ P_y \circ w_i &= t_i \circ P_y \end{aligned}$$

Then

$$\begin{aligned} P_z(F_W) &= F_V \\ P_x(F_W) &= F_U \\ P_y(F_W) &= F_T \end{aligned}$$

One such IFS produces the Sierpinski Tetrahedron, whose three standard projections in $\mathbb{R}^2$ are each the Sierpinski Triangle. This IFS is given by

$$W(v) = Mv \cup Mv + b_1 \cup Mv + b_2 \cup Mv + b_3 \tag{3}$$

for all $v \in \mathbb{R}^3$ where $M = \begin{bmatrix} .5 & 0 & 0 \\ 0 & .5 & 0 \\ 0 & 0 & .5 \end{bmatrix}$, $b_1 = \begin{bmatrix} 0.5 \\ 0 \\ 0 \end{bmatrix}$, $b_2 = \begin{bmatrix} 0 \\ 0.5 \\ 0 \end{bmatrix}$, and $b_3 = \begin{bmatrix} 0 \\ 0 \\ 0.5 \end{bmatrix}$.
Its projection onto the xy plane is the Sierpinski Triangle, which is the attractor of

$$V(u) = M'u \cup M'u + b_1' \cup M'u + b_2' \tag{4}$$

for all $u \in \mathbb{R}^2$ where $M' = \begin{bmatrix} .5 & 0 \\ 0 & .5 \end{bmatrix}$, $b_1' = \begin{bmatrix} 0.5 \\ 0 \end{bmatrix}$ and $b_2' = \begin{bmatrix} 0 \\ 0.5 \end{bmatrix}$.

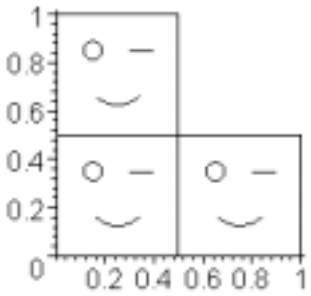
Since the above theorems are not if and only if cases, it is natural to ask if there exist IFS's on $\mathbb{R}^3$ of a form other than that suggested by the previous theorems whose attractors would also have projections into $\mathbb{R}^2$ that were themselves the attractors of 2d IFS's. Furthermore we would like to know if it is possible to create a fractal in $\mathbb{R}^3$ having any three two-dimensional IFS attractors as its projections onto each one of the three coordinate planes. In the general case, the answer is clearly no, as can be seen by the following counterexample. Given that a single point can be the attractor of an IFS, we can ask if a 3-dimensional fractal exists which has a single point as its projection onto the xy, and yz planes and having the Sierpinski Triangle as its projection onto the xz plane. This clearly cannot be done as the only shape which can have a single point for two of its projections is a single point.

However, a typical attractor of an IFS is an uncountable set of points. But even if we restrict ourselves to such attractors for the choice of the three projections, the answer is still clearly no. It is clear that any object in $\mathbb{R}^3$ must be contained completely within the intersection of the three sets formed by extending each projection into its respective third dimension. Similarly, a 3d IFS attractor would have to be entirely within the intersection of the x, y, or z extensions formed by each attractor in $\mathbb{R}^2$, but this intersection can be empty. Thus it is not possible in general to find an IFS on $\mathbb{R}^3$ whose attractor has a given set of projections, even if those projections are the attractors of two-dimensional IFS's.

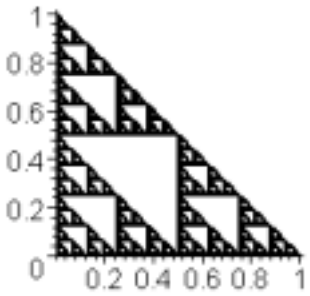
3.3. **Relatives of the Sierpinski Tetrahedron.** There are many other IFS's besides V given by (4) which produce the Sierpinski Triangle as their attractor. Peitgen, Jürgens, and Saupe [4] (pg 246) studied and classified the attractors of a family of IFS's related to V. Similarly, we now investigate a family of IFS's related to W given by (3).

In the case of the Sierpinski triangle, notice that each affine map in V has the effect of scaling each of the $\vec{i}$ and $\vec{j}$ basis vectors by 1/2 and then translating them to one of three fixed positions while keeping the vectors orthogonal to one another without any rotations or reflections. A graphical representation of the first iteration of this IFS using MrFace (Appendix C) as a seed shows the relation of each affine map to one another. Each face

corresponds to a single affine map.



First iteration of V using MrFace as seed



6^{th} iteration of V using a solid square as seed

(5)

We can reorient any of these faces according to the 8 symmetry operations of the square without changing their relative positions to one another. Each such orientation and position corresponds to a unique affine map. Therefore it is possible to define $8^3 = 512$ distinct IFS's just by using this Sierpinski-related L-shaped configuration of maps. Peitgen, Jürgens, and Saupe studied and classified the attractors of this IFS family. In particular, they found within this family alone 8 distinct IFS's that each generate the Sierpinski Triangle as their attractor. Moreover, there are 7 other distinct attractors each of which is generated by 8 distinct IFS's from this family.

Naturally we would like to know if analogous results hold in $\mathbb{R}^3$. Therefore we consider a family of IFS's related to the Sierpinski Tetrahedron as produced by the IFS W given by (3). Rendering the first iteration of W using MrBlockHead (Appendix C) as our seed we can see a geometric representation of each affine map involved.

(6)

First iteration of W using MrBlockHead as seed

Just as the MrFaces above are all contained within the unit square, here all four blocks are similarly contained within the unit cube. Each of the four cubes corresponds to one of the four affine maps in the definition of W. Since there are 48 symmetry operations for the cube, there are 48^4 different possible ways of orienting these four blocks within this basic tetragonal formation, with each configuration corresponding to a unique IFS. We describe this family of IFS's precisely in the following definition.

Definition 3.10. *Let v_1, v_2, v_3, and v_4 be the affine maps on $\mathbb{R}^3$ defined as follows*

$$\begin{aligned} v_1(x,y,z) &= (1/2x, 1/2y, 1/2z) \\ v_2(x,y,z) &= v_1(x,y,z) + (1/2,0,0) \\ v_3(x,y,z) &= v_1(x,y,z) + (0,0,1/2) \\ v_4(x,y,z) &= v_1(x,y,z) + (0,1/2,0) \end{aligned}$$

for all $(x,y,z) \in \mathbb{R}^3$. Let C denote the group of symmetry operations of the cube. Then we define the set of IFS relatives of the Sierpinski Tetrahedron to be the set

$$\Psi = \{W : W = v_1 d_1 \cup v_2 d_2 \cup v_3 d_3 \cup v_4 d_4 \text{ and } d_1, d_2, d_3, d_4 \in C\}$$

Note that in this definition, we are identifying an element d of C with the unique affine map on $\mathbb{R}^3$ whose restriction to the cube $[0,1]^3$ is the symmetry operation d. Our notation for the elements in the group of symmetry operations on the cube is defined in Appendix A. Six of these symmetry operations are needed in order to investigate important reflective and rotational symmetries according to which we can categorize the attractors of Ψ. We define this set of reflections and rotations as follows.

$$\Delta = \left\{ \begin{array}{l} e = \text{ the identity map} \\ \alpha_1 = -t = \text{ reflection across the } x = y \text{ plane} \\ \alpha_2 = -r^2 s^3 = \text{ reflection across the } y = z \text{ plane} \\ \alpha_3 = -r^3 = \text{ reflection across the } x = z \text{ plane} \\ \alpha_4 = rt = \text{ rotation about the line } x = y = z \text{ by } 120^\circ \text{ directed from the } x\text{-axis to } z\text{-axis} \\ \alpha_5 = r^3 s^3 = \text{ rotation about the line } x = y = z \text{ by } 120^\circ \text{ directed from the } x\text{-axis to } y\text{-axis} \end{array} \right\} \tag{7}$$

The following definition clarifies a subscripting notation that will be used frequently in the following theorems.

Definition 3.11. *Let $\alpha \in \Delta$ and $\bigcup_{i=1}^{4} v_i d_i \in \Psi$. Let $\sigma : \Delta \times \{1,2,3,4\} \to \{1,2,3,4\}$ be a map such that for each α, define $\sigma_\alpha : \{1,2,3,4\} \to \{1,2,3,4\}$ such that $\sigma_\alpha(i) = \sigma(\alpha, i)$ is given by*

$$v_{\sigma(\alpha,i)} = \alpha^{-1} v_i \alpha.$$

For all α, $\sigma_\alpha \in S_4$. By direct calculation, one can verify that sigma is a well defined function. We will usually abbreviate $\sigma(\alpha, i) = \overline{i}$ when the value of α is clear from context. The values of $\overline{i}$ are listed in the following table.

i	*1*	*2*	*3*	*4*
$\sigma(e,i)$	1	2	3	4
$\sigma(\alpha_1,i)$	1	4	3	2
$\sigma(\alpha_2,i)$	1	2	4	3
$\sigma(\alpha_3,i)$	1	3	2	4
$\sigma(\alpha_4,i)$	1	4	2	3
$\sigma(\alpha_5,i)$	1	3	4	2

Given any $W \in \Psi$ we can find a corresponding $W' \in \Psi$ whose attractor is a reflection or rotation of F_W with respect to some $\alpha \in \Delta$ according to the following theorem.

Theorem 3.12. *Let* $\alpha \in \Delta$ *and* $W = \bigcup_{i=1}^{4} v_i d_i \in \Psi$. *Define* $W' = \bigcup_{i=1}^{4} v_i d_i' \in \Psi$ *by*

$$d_i' = \alpha d_{\overline{i}} \alpha^{-1} \text{ for all } i \in \{1,2,3,4\}$$

then $\alpha(F_W) = F_{W'}$.

The Sierpinski tetrahedron and triangle are the prototypes for the families of IFS's being considered. As a starting point for the study of symmetries of attractors in these families, we can determine its group of symmetry operations for these two fractals.

Theorem 3.13. *Let* $(\mathrm{Sym}(t), \circ)$ *be the group of all symmetry operations on the Sierpinski right triangle and let* $(\mathrm{Sym}(T), \circ)$ *be the group of all symmetry operations on the Sierpinski right tetrahedron. Then*

a) $\mathrm{Sym}(t)$ *is isomorphic to* $\mathbb{Z}_2$ *(and hence* D_2*)*
b) $\mathrm{Sym}(T)$ *is isomorphic to*S_3 *(and hence* D_3*)*

As mentioned above, in the case of the Sierpinski triangle there are 8 distinct V-related (4) IFS's that will produce it. So naturally we would like to know how many distinct $W \in \Psi$ produce the Sierpinski right tetrahedron as their attractor. In general, we would like to be able to classify all the attractors of the set Ψ according to the following criteria.

Given $W \in \Psi$ such that $W(A) = A$ we wish to know
1) What is the order of the symmetry group of A? ($|\mathrm{Sym}(A)| = ?$)
2) How many distinct $W' \in \Psi$ also produce A as their attractor?

We can answer the second question if we know the answer to the first as shown in the next theorem.

Theorem 3.14. *Let* $W \in \Psi$ *such that* $W(A) = A$. *Let* $\mathrm{Sym}(A)$ *be the group of symmetry operations on* A.

Case I : *If* $|\mathrm{Sym}(A)| = 1$ *then only* 1 *IFS in* Ψ *has attractor* A.
Case II : *If* $|\mathrm{Sym}(A)| = 2$ *then* $2^4 = 16$ *distinct IFS's in* Ψ *have attractor* A.
Case III : *If* $|\mathrm{Sym}(A)| = 3$ *then* $3^4 = 81$ *distinct IFS's in* Ψ *have attractor* A.
Case IV : *If* $|\mathrm{Sym}(A)| = 6$ *then* $6^4 = 1296$ *distinct IFS's in* Ψ *have attractor* A.

For example, by this theorem, we know that there must be 1296 distinct IFS's in Ψ that all produce the Sierpinski right tetrahedron. Using our Maple program for rendering attractors, we have found 8 distinct attractors, A, of IFS's in our family, Ψ, having $|\mathrm{Sym}(A)| = 6$, each of which is the attractor of 1296 distinct IFS's. The IFS definitions for each of these attractors are listed in Appendix E. The proof of this theorem, given in the following section, reveals how once we know the symmetry group belonging to the attractor of a given IFS $W \in \Psi$ we can define all other IFS's in our family that also converge to F_W. Moreover, if we know that W produces an asymmetric attractor, i.e. $|\mathrm{Sym}(F_W)| = 1$, then Theorem 3.12 allows us to define the five other distinct $W_i \in \Psi$ that each converge to $\alpha_i(F_W)$ for $i \in \{1..5\}$.

Clearly, in order to count the total number of distinct attractors in our family and to define the IFS's that produce them, we require a condition on all d_i in a given $W \in \Psi$ such that $W(A) = A$ is guaranteed to have a symmetry group of a certain order. The following theorems make some progress toward solving this problem, but further work is required.

Theorem 3.15. *Let $\alpha \in \Delta$ and let $W \in \Psi$ such that $A = W(A) = \bigcup_{i=1}^{4} v_i d_i(A)$.*

$$\alpha \in \mathrm{Sym}(A) \Leftrightarrow \forall i,\ d_i^{-1} \alpha d_{\overline{i}} \in \mathrm{Sym}(A).$$

Theorem 3.16. *Let $\alpha \in \Delta$ and let $W \in \Psi$ such that $A = W(A) = \bigcup_{i=1}^{4} v_i d_i(A)$.*

$$\text{If } \forall i,\ \alpha d_{\overline{i}} = d_i \alpha \text{ then } \alpha \in \mathrm{Sym}(A).$$

Theorem 3.17. *Let $\alpha \in \Delta$ and let $W, W' \in \Psi$ such that $W = \bigcup_{i=1}^{4} v_i d_i$, $W'(A) = \bigcup_{i=1}^{4} v_i d_i'(A)$ and $A = W'(A)$.*

$$A = W(A) \Leftrightarrow \forall i, d_i^{-1} d_i' \in \mathrm{Sym}(A)$$

We leave these as reference for those continuing to research the full classification of this IFS family and also in search of the answer to the broader question which asks how to determine the symmetry group of the attractor of a given IFS.

4. *Proofs*

4.1. **Contraction Factor Theorem.** The proof of the following contraction factor theorem relies upon the following definitions and theorems taken from Lay's linear algebra book [2] where the proofs of Theorems 4.3 and 4.4 can be found.

Definition 4.1. *A quadratic form on $\mathbb{R}^n$ is a function $Q : \mathbb{R}^n \to \mathbb{R}^n$ defined by $Q(\vec{x}) = \vec{x}^T A \vec{x}$ for all $\vec{x} \in \mathbb{R}^n$ where $A \in M_n(\mathbb{R})$ is symmetric ($A^T = A$).*

Definition 4.2. *A quadratic form Q is*
*a) **positive definite** if $Q(\vec{x}) > 0$ for all $\vec{x} \neq 0$*
*b) **negative definite** if $Q(\vec{x}) < 0$ for all $\vec{x} \neq 0$*
*c) **indefinite** if $Q(\vec{x})$ assumes both positive and negative values*

Theorem 4.3 (A)**.** *A quadratic form $\vec{x}^T A \vec{x}$ is*
a) positive definite iff the eigenvalues of A are all positive.
b) negative definite iff the eigenvalues of A are all negative.
c) indefinite iff A has both positive and negative eigenvalues.

Theorem 4.4 (B)**.** *Let A be a symmetric matrix and define $m = \min\left\{\vec{x}^T A \vec{x} : ||\vec{x}|| = 1\right\}$ and $M = \max\left\{\vec{x}^T A \vec{x} : ||\vec{x}|| = 1\right\}$. Then M is the largest eigenvalue, λ_1, of A and m is the smallest eigenvalue of A. The value of $\vec{x}^T \vec{x}$ is M when $\vec{x}$ is a unit eigenvector, $\vec{v}_1$, corresponding to M.The value of $\vec{x}^T A \vec{x}$ is m when $\vec{x}$ is a unit eigenvector corresponding to m.*

With this background we can now prove Theorem 3.1 which we restate here for convenience.

Theorem 3.1. ***(Contraction Factor)***
Let $T : \mathbb{R}^n \to \mathbb{R}^n$ by $T(x) = Mx + B$ where $M \in M_n(\mathbb{R})$. Let

$$c = \max\left\{\lambda : \lambda \text{ is an eigenvalue of } M^T M\right\}.$$

T is a contraction map iff $\sqrt{c} < 1$. If T is a contraction map then $\sqrt{c}$ is the best possible contraction factor.

Proof. Let $\vec{v}, \vec{w} \in \mathbb{R}^n$ such that $\vec{v} \neq \vec{w}$. Let $\vec{q} = \vec{v} - \vec{w}$ and let $\vec{u} = \frac{\vec{q}}{||\vec{q}||}$. The vector $\vec{u}$ is well defined since $q \neq \vec{0}$. Furthermore, by definition of unit vector, $||\vec{u}|| = 1$. Let $r = ||\vec{q}||$. Then by substitution, $\vec{q} = r\vec{u}$. Let d be the Euclidean metric in.$\mathbb{R}^n$. Then by definition, $d(\vec{v}, \vec{w}) = ||\vec{v} - \vec{w}|| = ||\vec{q}|| = r$. Now, let $T : \mathbb{R}^n \to \mathbb{R}^n$ such that $T(x) = Mx + \vec{B}$ where $M \in M_n(\mathbb{R})$ and $\vec{B} \in \mathbb{R}^n$. Then

$$\begin{aligned}
d\left(T\left(\vec{w}\right), T\left(\vec{v}\right)\right) &= ||T(\vec{v}) - T(\vec{w})|| \\
&= || \left(M\vec{v} + \vec{B}\right) - \left(M\vec{w} + \vec{B}\right) || \text{ by definition of } d \\
&= ||M\vec{v} - M\vec{w}|| \\
&= ||M(\vec{v} - \vec{w})|| \\
&= ||M\vec{q}|| \\
&= ||M(r\vec{u})|| \\
&= ||rM\vec{u}|| \\
&= |r| \ \ ||M\vec{u}|| \\
&= r\,||M\vec{u}|| \\
&= r\sqrt{M\vec{u} \cdot M\vec{u}} \text{ by definition of vector length} \\
&= r\sqrt{(M\vec{u})^T M\vec{u}} \text{ by the property of the dot product} \\
&= r\sqrt{\vec{u}^T M^T M\vec{u}} \text{ by property of the transpose}
\end{aligned}$$

Let $A = M^T M$. It is well known that A is a symmetric matrix. Since $M\vec{u} \cdot M\vec{u}$ is always positive, $\vec{u}^T A\vec{u} > 0$,for all $\vec{u}$. Define $c = \max\left\{\vec{u}^T A\vec{u} : ||u|| = 1\right\}$. Then by theorem 4.4, $c = \max\left\{\lambda : \lambda \text{ is an eigenvalue of } A\right\}$. Moreover, since $\vec{u}^T A\vec{u}$ is positive definite, we know that $c > 0$ by Theorem 4.3 part a. According to the equations above, we have

$$\begin{aligned}
d\left(T\left(\vec{v}\right), T\left(\vec{w}\right)\right)^2 &= r^2(\vec{u}^T A\vec{u}) \\
&\leq r^2 c
\end{aligned}$$

and so since $\sqrt{}$ is an increasing function,

$$\begin{aligned}
d\left(T\left(\vec{v}\right), T\left(\vec{w}\right)\right) &\leq r\sqrt{c} \\
&= \sqrt{c}d(\vec{v}, \vec{w})
\end{aligned}$$

(<=)
Assume $\sqrt{c} < 1$ and since we have shown that $d\left(T\left(\vec{v}\right), T\left(\vec{w}\right)\right) \leq \sqrt{c}d(\vec{v}, \vec{w})$ for all $\vec{v}, \vec{w} \in \mathbb{R}^n$, we know by definition that T is a contraction map with contraction factor $\sqrt{c}$.
(=>)
To show that $\sqrt{c}$ is the best possible contraction factor, assume that T is a contraction map with contraction factor $s < 1$. Let $\vec{x}$ be the unit eigenvector corresponding to eigenvalue c.

By Theorem 4.4 we know that $\vec{x}^T A\vec{x} = c$. We therefore can show

$$\begin{aligned} d(T\left(\vec{0}\right), T(\vec{x}))^2 &= \left|\left|T(\vec{x}) - T\left(\vec{0}\right)\right|\right|^2 \\ &= \left|\left|M\vec{x} + \vec{B} - (M\vec{0} + \vec{B})\right|\right|^2 \\ &= \left|\left|M\vec{x} - M\vec{0}\right|\right|^2 \\ &= \left|\left|M(\vec{x} - \vec{0})\right|\right|^2 \\ &= ||M\vec{x}||^2 \\ &= \vec{x}^T M^T M\vec{x} \\ &= \vec{x}^T A\vec{x} \\ &= c \\ &= c(1^2) \\ &= c\,||\vec{x}||^2 \\ &= c\left|\left|\vec{x} - \vec{0}\right|\right|^2 \\ &= cd(\vec{0}, \vec{x})^2 \end{aligned}$$

and so

$$\begin{aligned} d(T(\vec{0}), T(\vec{x})) &= \sqrt{c}d(\vec{0}, \vec{x}) \\ &\leq sd(\vec{0}, \vec{x}) \end{aligned}$$

which implies that $\sqrt{c} \leq s$. By definition we know that $s < 1$ so $\sqrt{c} < 1$ and since$\sqrt{c} \leq s$, we have shown that $\sqrt{c}$ is the best possible contraction factor for T. □

Corollary 3.2. *Let $T : \mathbb{R}^3 \to \mathbb{R}^3$ be the affine map given by $T(x) = Mx + B$, where M is a diagonal matrix. Then a scaling factor for T is*

$$\max\{|m| : m \text{ is a value on the main diagonal of } M\}.$$

If $\max\{|m|\} < 1$ then T is a contraction mapping and this is the best possible contraction factor.

Proof. Let $T : \mathbb{R}^3 \to \mathbb{R}^3$ be the affine map given by $T(x) = Mx + B$, where M is a diagonal matrix. From Theorem 3.1 we see that the scaling factor for any affine map $T(x) = Mx + B$ is $\sqrt{c}$ where $c = \max\left\{\lambda : \lambda \text{ is an eigenvalue of } M^T M\right\}$. If M is a diagonal matrix, then T has only the effect of scaling each of the basis vectors $\vec{i}, \vec{j}, \vec{k}$ by some constants $x, y, z \in \mathbb{R}$ respectively before translation, and M is given by

$$M = \begin{bmatrix} x & 0 & 0 \\ 0 & y & 0 \\ 0 & 0 & z \end{bmatrix}.$$

Therefore,

$$M^T M = \begin{bmatrix} x & 0 & 0 \\ 0 & y & 0 \\ 0 & 0 & z \end{bmatrix} \begin{bmatrix} x & 0 & 0 \\ 0 & y & 0 \\ 0 & 0 & z \end{bmatrix} = \begin{bmatrix} x^2 & 0 & 0 \\ 0 & y^2 & 0 \\ 0 & 0 & z^2 \end{bmatrix}.$$

Solving for the eigenvalues we get

$$0 = \det(M^T M - \lambda I) = \det \begin{bmatrix} x^2 - \lambda & 0 & 0 \\ 0 & y^2 - \lambda & 0 \\ 0 & 0 & z^2 - \lambda \end{bmatrix} = (x^2 - \lambda)(y^2 - \lambda)(z^2 - \lambda)$$

Therefore, the eigenvalues of $M^T M$ are x^2, y^2, and z^2 and $c = \max\left\{x^2, y^2, z^2\right\}$. Since the scaling factor is $\sqrt{c}$ we know that for an affine map T having a diagonal matrix M the scaling factor is simply $\max\{|m| : m \text{ is a value on the main diagonal of } M\}$. Then by Theorem 3.1 and the definition of contraction map, if $\max\{|m|\} < 1$ then T is a contraction mapping and $\max\{|m|\}$ is the best possible contraction factor. □

Corollary 3.3. *Let $T : \mathbb{R}^3 \to \mathbb{R}^3$ be the affine map given by $T(x) = Mx + B$ where $M^T M$ is diagonal. Then the scaling factor for T is*

$$\max\left\{\sqrt{\alpha} : \alpha \text{ is a value on the main diagonal of } M^T M\right\}.$$

If $\max\{\sqrt{\alpha}\} \in (0, 1)$ then T is a contraction mapping and this is the best possible contraction factor.

Proof. Let $T : \mathbb{R}^3 \to \mathbb{R}^3$ be the affine map given by $T(x) = Mx + B$ and $M^T M$ is diagonal. From linear algebra we know that the eigenvalues of a diagonal matrix are equal to values on the main diagonal of that matrix. Therefore by Theorem 3.1 we have

$$\begin{aligned} c &= \max\left\{\lambda : \lambda \text{ is an eigenvalue of } M^T M\right\} \\ &= \max\left\{\alpha : \alpha \text{ is a value on the main diagonal of } M^T M\right\}. \end{aligned}$$

Thus the scaling factor for T is $\sqrt{c} = \max\left\{\sqrt{\alpha} : \alpha \text{ is a value on the main diagonal of } M^T M\right\}$. Thus by definition of contraction map, if $\max\{\sqrt{\alpha}\} \in (0, 1)$ then T is a contraction mapping and $\max\{\sqrt{\alpha}\}$ is the best possible contraction factor. □

Corollary 3.4. *Let $T : \mathbb{R}^3 \to \mathbb{R}^3$ be the affine map given by $T(x) = Mx + B$, where M is a matrix in the form (2) with $r, s, t \in \mathbb{R}^+$. Then a scaling factor for T is*

$$\max\{r, s, t\}$$

If $\max\{r, s, t\} < 1$ then T is a contraction mapping and this is the best possible contraction factor.

Proof. Let $T : \mathbb{R}^3 \to \mathbb{R}^3$ be the affine map given by $T(x) = Mx + B$. Consider an affine map having a matrix $M = (\vec{v}_i, \vec{v}_j, \vec{v}_k)$ composed of column vectors

$$\vec{v}_i = \begin{bmatrix} a_i \\ b_i \\ c_i \end{bmatrix}, \vec{v}_j = \begin{bmatrix} a_j \\ b_j \\ c_j \end{bmatrix}, \vec{v}_k = \begin{bmatrix} a_k \\ b_k \\ c_k \end{bmatrix}$$

that are images of $\vec{i}, \vec{j}, \vec{k}$ respectively under the affine map T before translation. We have

$$M^T M = \begin{bmatrix} a_i & b_i & c_i \\ a_j & b_j & c_j \\ a_k & b_k & c_k \end{bmatrix} \begin{bmatrix} a_i & a_j & a_k \\ b_i & b_j & b_k \\ c_i & c_j & c_k \end{bmatrix}$$

is diagonal if and only if

$$\begin{aligned} 0 &= a_i a_j + b_i b_j + c_i c_j = \vec{v}_i \cdot \vec{v}_j \\ 0 &= a_i a_k + b_i b_k + c_i c_k = \vec{v}_i \cdot \vec{v}_k \\ 0 &= a_j a_k + b_j b_k + c_j c_k = \vec{v}_j \cdot \vec{v}_k. \end{aligned}$$

Since the dot product of any column of M with the each of the other two columns must be zero, then the columns of M must be orthogonal and linearly independent. Therefore $\vec{v}_i, \vec{v}_j$,and $\vec{v}_k$ form an orthogonal basis for $\mathbb{R}^3$. Geometrically, since $\vec{v}_i, \vec{v}_j$,and $\vec{v}_k$ must all be orthogonal to one another we see that $M^T M$ is diagonal if and only if M is a matrix that rotates and scales the unit cube such that T of the unit cube is still an orthogonal parallelepiped. Certainly then, an affine map determined using the *Geometric form (*2) for M,

$$M = \begin{bmatrix} r\cos\phi\cos\gamma & s\sin\theta\sin\phi\cos\gamma - s\cos\theta\sin\gamma & t\cos\theta\sin\phi\cos\gamma + t\sin\theta\sin\gamma \\ r\cos\phi\sin\gamma & s\sin\theta\sin\phi\sin\gamma + s\cos\theta\cos\gamma & t\cos\theta\sin\phi\sin\gamma - t\sin\theta\cos\gamma \\ -r\sin\phi & s\sin\theta\cos\phi & t\cos\theta\cos\phi \end{bmatrix}$$

will always have a c value, as in Theorem 3.1, given by

$$\begin{aligned} c &= \max\left\{\alpha : \alpha \text{ is a value on the main diagonal of } M^T M\right\} \\ &= \max\left\{r^2, s^2, t^2\right\}. \end{aligned}$$

Here, $r, s, t \in \mathbb{R}^+$ are the scaling coefficients for $\vec{i}, \vec{j}$,and $\vec{k}$ respectively and so cannot be negative. Therefore, the scaling factor for T is

$$\sqrt{c} = \max\{r, s, t\}.$$

Thus by definition of contraction map, if $\max\{r, s, t\} < 1$ then T is a contraction mapping and $\max\{r, s, t\}$ is the best possible contraction factor. □

4.2. **2d Projections of 3d Attractors.** For the definition of P_z see definition 3.5 above.

Theorem 3.6. *Let $w : \mathbb{R}^3 \to \mathbb{R}^3$ and $v : \mathbb{R}^2 \to \mathbb{R}^2$ be affine maps. Then $P_z \circ w = v \circ P_z$ if and only if there exists $a, b, c, d, e, f, g, h, i \in \mathbb{R}$ such that*

$$w(x, y, z) = \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ e & f & g \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} h \\ i \\ j \end{bmatrix}$$

and

$$v(x, y) = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} h \\ i \end{bmatrix}$$

for all $(x, y, z) \in \mathbb{R}^3$.

Proof. (=>)

Let $w : \mathbb{R}^3 \to \mathbb{R}^3$ and $v : \mathbb{R}^2 \to \mathbb{R}^2$ be affine maps such that

$$\begin{aligned} w(x, y, z) &= \begin{bmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{bmatrix} w_{10} \\ w_{11} \\ w_{12} \end{bmatrix} \text{ and} \\ v(x, y) &= \begin{bmatrix} v_1 & v_2 \\ v_3 & v_4 \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{bmatrix} v_5 \\ v_6 \end{bmatrix} \end{aligned}$$

for some $w_i, v_i \in \mathbb{R}$.

Assume $P_z \circ w(x,y,z) = v \circ P_z(x,y,z)$ for all $(x,y,z) \in \mathbb{R}^3$. Then

$$\begin{aligned}
& (w_1x + w_2y + w_3z + w_{10}, w_4x + w_5y + w_6z + w_{11}) \\
= \; & P_z \circ (w_1x + w_2y + w_3z + w_{10}, w_4x + w_5y + w_6z + w_{11}, w_7x + w_8y + w_9z + w_{12}) \\
= \; & P_z \circ w(x,y,z) \\
= \; & v \circ P_z(x,y,z) \\
= \; & v(x,y) \\
= \; & (v_1x + v_2y + v_5, v_3x + v_4y + v_6)
\end{aligned}$$

By definition of equality for ordered pairs we have the following two equations,

$$w_1x + w_2y + w_3z + w_{10} = v_1x + v_2y + v_5 \text{ and } w_4x + w_5y + w_6z + w_{11} = v_3x + v_4y + v_6.$$

Since these equations must hold for all $x, y, z \in \mathbb{R}^3$, the first equation guarantees that

$$\begin{aligned}
w_1 &= v_1 \\
w_2 &= v_2 \\
w_3 &= 0 \\
w_{10} &= v_5
\end{aligned}$$

and by the second equation we have

$$\begin{aligned}
w_4 &= v_3 \\
w_5 &= v_4 \\
w_6 &= 0 \\
w_{11} &= v_6.
\end{aligned}$$

Therefore, w and v are of the form

$$\begin{aligned}
w(x,y,z) &= \begin{bmatrix} w_1 & w_2 & 0 \\ w_4 & w_5 & 0 \\ w_7 & w_8 & w_9 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{bmatrix} w_{10} \\ w_{11} \\ w_{12} \end{bmatrix} \text{ and} \\
v(x,y) &= \begin{bmatrix} w_1 & w_2 \\ w_4 & w_5 \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{bmatrix} w_{10} \\ w_{11} \end{bmatrix}.
\end{aligned}$$

(<=)

Assume that w and v are defined such that

$$\begin{aligned}
w(x,y,z) &= \begin{bmatrix} w_1 & w_2 & 0 \\ w_4 & w_5 & 0 \\ w_7 & w_8 & w_9 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{bmatrix} w_{10} \\ w_{11} \\ w_{12} \end{bmatrix} \text{ and} \\
v(x,y) &= \begin{bmatrix} w_1 & w_2 \\ w_4 & w_5 \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{bmatrix} w_{10} \\ w_{11} \end{bmatrix}
\end{aligned}$$

for some $w_i \in \mathbb{R}$. By the definition of P_z we have

$$\begin{aligned}
P_z \circ w(x,y,z) &= P_z \circ (w_1x + w_2y + w_{10}, w_4x + w_5y + w_{11}, w_7x + w_8y + w_9z + w_{12}) \\
&= (w_1x + w_2y + w_{10}, w_4x + w_5y + w_{11}) \\
&= v(x,y) \\
&= v \circ P_z(x,y,z)
\end{aligned}$$

thus $P_z \circ w(x,y,z) = v \circ P_z(x,y,z)$. □

Theorem 3.7. *Let $W = w_1 \cup \cdots \cup w_k$ and $V = v_1 \cup \cdots \cup v_k$ be IFS's on $\mathbb{R}^3$ and $\mathbb{R}^2$ respectively such that $P_z \circ w_i = v_i \circ P_z$ for all $i \in \{1,\ldots,k\}$. Then*

$$P_z \circ \Phi_W = \Phi_V.$$

Proof. Let $W = w_1 \cup \cdots \cup w_k$ and $V = v_1 \cup \cdots \cup v_k$ be IFS's on $\mathbb{R}^3$ and $\mathbb{R}^2$ respectively such that $P_z \circ w_i = v_i \circ P_z$ for all $i \in \{1,\ldots,k\}$. Define the function Φ as in Section 2 but with the restrictions that $\Phi_W : \Sigma_k \to \mathbb{R}^3$ and $\Phi_V : \Sigma_k \to \mathbb{R}^2$. Then $P_z \circ \Phi_W : \Sigma_k \to \mathbb{R}^2$. Let $s \in \Sigma_k$, then $s = s_1 s_2 ...$ such that for all j, $s_j \in \{1,\ldots,k\}$. By the definition of Φ_W we know that $\exists a \in \mathbb{R}^3$ such that $a = \Phi_W(s) = \lim_{n\to\infty} w_{s_1} \circ w_{s_2} \circ ... \circ w_{s_n}\left(\overrightarrow{0_3}\right)$. Note that P_z is a continuous map. Since limits commute with continuous maps, $\lim_{i\to\infty} P_z(q_i) = P_z(\lim_{i\to\infty} q_i)$ for any convergent sequence of points $q_i \in \mathbb{R}^3$. Therefore,

$$\begin{aligned}
P_z \circ \Phi_W(s) &= P_z\left(\lim_{n\to\infty} w_{s_1} \circ w_{s_2} \circ ... \circ w_{s_n}\left(\overrightarrow{0_3}\right)\right) \\
&= \lim_{n\to\infty} P_z\left(w_{s_1} \circ w_{s_2} \circ ... \circ w_{s_n}\left(\overrightarrow{0_3}\right)\right) \\
&= \lim_{n\to\infty} P_z \circ w_{s_1} \circ w_{s_2} \circ ... \circ w_{s_n}\left(\overrightarrow{0_3}\right) \\
&= \lim_{n\to\infty} v_{s_1} \circ P_z \circ w_{s_2} \circ ... \circ w_{s_n}\left(\overrightarrow{0_3}\right) \\
&\vdots \\
&= \lim_{n\to\infty} v_{s_1} \circ v_{s_2} \circ ... \circ v_{s_n} \circ P_z\left(\overrightarrow{0_3}\right) \\
&= \lim_{n\to\infty} v_{s_1} \circ v_{s_2} \circ ... \circ v_{s_n}\left(\overrightarrow{0_2}\right) \\
&= \Phi_V(s).
\end{aligned}$$

Thus $P_z \circ \Phi_W(s) = \Phi_V(s)$ for all $s = s_1 s_2 ... \in \Sigma_k$. □

Theorem 3.8. *Let $W = w_1 \cup \cdots \cup w_k$ and $V = v_1 \cup \cdots \cup v_k$ be IFS's on $\mathbb{R}^3$ and $\mathbb{R}^2$ respectively such that $P_z \circ w_i = v_i \circ P_z$ for all $i \in \{1,\ldots,k\}$. Then*

$$P_z(F_W) = F_V.$$

Proof. Let $W = w_1 \cup \cdots \cup w_k$ and $V = v_1 \cup \cdots \cup v_k$ be IFS's on $\mathbb{R}^3$ and $\mathbb{R}^2$ respectively such that $P_z \circ w_i = v_i \circ P_z$ for all $i \in \{1,\ldots,k\}$. Let $q \in P_z(F_W)$. Then there must exist an $a \in F_W$ such that $q = P_z(a)$. Since $\Phi_W(\Sigma_k) = F_W$ there exists $s = s_1 s_2 ... \in \Sigma_k$ such that $a = \Phi_W(s)$. Therefore, by the previous theorem

$$\begin{aligned}
q &= P_z \circ \Phi_W(s) \\
&= \Phi_V(s) \ .
\end{aligned}$$

But $\Phi_V(s) \in \Phi_V(\Sigma_k) = F_V$. Thus $q \in F_V$ and so $P_z(F_W) \subseteq F_V$. Now let $q' \in F_V$. As above, we know that there must exist an $r = r_1 r_2 ... \in \Sigma_k$ such that $q' = \Phi_V(r)$. Thus

$$\begin{aligned}
q' &= \Phi_V(r) \\
&= P_z \circ \Phi_W(r).
\end{aligned}$$

Again we know, $\Phi_W(r) \in \Phi_W(\Sigma_k) = F_W$. So $q' \in P_z(F_W)$ and so $F_V \subseteq P_z(F_W)$. Thus $P_z(F_W) = F_V$. □

Similar proofs confirm these three theorems with respect to the alternate definitions for w, v,and P mentioned in this section above.

4.3. **Sierpinski Symmetry groups.** The definition for $\overline{i}$ can be found in definition 3.11 above. For the following proof, note that $\alpha v_{\overline{i}} = v_i \alpha$.

Theorem 3.12. *Let $\alpha \in \Delta$ and $W = \bigcup_{i=1}^{4} v_i d_i \in \Psi$. Define $W' = \bigcup_{i=1}^{4} v_i d'_i \in \Psi$ by*

$$d'_i = \alpha d_{\overline{i}} \alpha^{-1} \text{ for all } i \in \{1,2,3,4\}.$$

Then $\alpha(F_W) = F_{W'}$.

Proof. Let $\alpha \in \Delta$ and $W = \bigcup_{i=1}^{4} v_i d_i \in \Psi$. F_W is the unique $A \in K_3$ such that $A = W(A)$. Define $W' = \bigcup_{i=1}^{4} v_i d'_i \in \Psi$ by $d'_i = \alpha d_{\overline{i}} \alpha^{-1}$ for all $i \in \{1,2,3,4\}$.

$$\begin{aligned}
\alpha A &= \alpha(W(A)) \\
&= \alpha\left(\bigcup_{i=1}^{4} v_i d_i(A)\right) \\
&= \alpha\left(\bigcup_{i=1}^{4} v_{\overline{i}} d_{\overline{i}} A\right) \text{ by definition of } \overline{i} \\
&= \bigcup_{i=1}^{4} \alpha v_{\overline{i}} d_{\overline{i}} A \\
&= \bigcup_{i=1}^{4} \alpha v_{\overline{i}} d_{\overline{i}} \alpha^{-1} \alpha A \\
&= \bigcup_{i=1}^{4} v_i \alpha d_{\overline{i}} \alpha^{-1} \alpha A \\
&= \bigcup_{i=1}^{4} v_i d'_i(\alpha A)
\end{aligned}$$

So by definition and uniqueness of the attractor of an IFS we know that $\alpha A = W'(\alpha A)$, and so $\alpha(F_W) = F_{W'}$. □

Theorem 3.13. *Let $(\mathrm{Sym}(t), \circ)$ be the group of all symmetry operations on the Sierpinski right triangle and let $(\mathrm{Sym}(T), \circ)$ be the group of all symmetry operations on the Sierpinski right tetrahedron. Then*

a) $\mathrm{Sym}(t)$ *is isomorphic to* $\mathbb{Z}_2$ *(and hence* D_2*)*
b) $\mathrm{Sym}(T)$ *is isomorphic to*S_3 *(and hence* D_3*)*

Proof. Part a)

On the Sierpinski Triangle, the only two points separated by a distance of $\sqrt{2}$ are $(1,0)$ and $(0,1)$. Since a symmetry operation must preserve distances, these two points must map to each other. Furthermore, the only point contained in the Sierpinski right triangle that is exactly one unit away from each of these points is the origin. It therefore must map to itself. Since a symmetry map in $\mathbb{R}^2$ is completely determined by where it sends three

non-colinear points, the only possible symmetry operation on the Sierpinski Triangle other than the identity map is $g : \mathbb{R}^2 \to \mathbb{R}^2$ such that $g(x, y) = (y, x)$ for all $x, y \in \mathbb{R}$.

To see that g is indeed a symmetry operation on the Sierpinski right triangle, let $W : \mathbb{R}^2 \to \mathbb{R}^2$ be the IFS whose attractor is the Sierpinski right triangle defined by $V = v_1 \cup v_2 \cup v_3$ where v_i is defined as in (4) above. Using these definitions we can see that for any $(x, y) \in \mathbb{R}^2$

$$\begin{aligned} v_1(g(x,y)) &= v_1(y,x) = \left(\frac{1}{2}y, \frac{1}{2}x\right) = g(\frac{1}{2}x, \frac{1}{2}y) = g(v_1(x,y)) \\ v_2(g(x,y)) &= v_2(y,x) = \left(\frac{1}{2}y + \frac{1}{2}, \frac{1}{2}x\right) = g(\frac{1}{2}x, \frac{1}{2}y + \frac{1}{2}) = g(v_3(x,y)) \\ v_3(g(x,y)) &= v_1(y,x) = \left(\frac{1}{2}y, \frac{1}{2}x + \frac{1}{2}\right) = g(\frac{1}{2}x + \frac{1}{2}, \frac{1}{2}y) = g(v_2(x,y)). \end{aligned}$$

And so we have

$$\begin{aligned} v_1 g &= g v_1 \\ v_2 g &= g v_3 \\ v_3 g &= g v_2. \end{aligned}$$

Now let $A = v_1(A) \cup v_2(A) \cup v_3(A)$ be the Sierpinski right triangle. From the above equations we see that

$$\begin{aligned} V(g(A)) &= v_1(g(A)) \cup v_2(g(A)) \cup v_3(g(A)) \\ &= g(v_1(A)) \cup g(v_3(A)) \cup g(v_2(A)) \\ &= g(v_1(A) \cup v_2(A) \cup v_3(A)) \\ &= g(A). \end{aligned}$$

By uniqueness of the attractor of an IFS we can conclude that $g(A) = A$. So g is a symmetry operation on A and therefore, all possible symmetry operations on the Sierpinski right triangle are

$$\begin{aligned} g^0(x,y) &= (x,y) = e \\ \text{and } g(x,y) &= (y,z). \end{aligned}$$

Let $\mathrm{Sym}(t) = \{e, g\}$ then $\mathrm{Sym}(t)$ is isomorphic to $\mathbb{Z}_2$ (which is in turn isomorphic to D_2).

Part b)

On the Sierpinski right tetrahedron, the only three points separated by a distance of $\sqrt{2}$ are $(1, 0, 0)$ and $(0, 1, 0)$ and $(0, 0, 1)$. Since a symmetry operation must preserve distances, these three points must map to one another. Furthermore, the only point contained in the Sierpinski right tetrahedron that is exactly one unit away from each of these points is the origin. It therefore must map to itself. Since a symmetry map in $\mathbb{R}^3$ is completely determined by where it sends four non-coplanar points, the five possible symmetry operations on the Sierpinski right tetrahedron in addition to the identity map are defined by $R_i : \mathbb{R}^3 \to \mathbb{R}^3$ for $i \in \{1, 2, 3, 4, 5\}$ such that for all $x, y, z \in \mathbb{R}$

$$\begin{aligned} R_1(x,y,z) &= (y,z,x) \\ R_2(x,y,z) &= (z,x,y) \\ R_3(x,y,z) &= (y,x,z) \\ R_4(x,y,z) &= (x,z,y) \\ R_5(x,y,z) &= (z,y,x) \end{aligned}$$

We can show that all R_i can be generated by two of these functions. Therefore, rename R_1 and R_3 such that $Q = R_1$ and $L = R_3$. Let $W : \mathbb{R}^3 \to \mathbb{R}^3$ be the IFS producing the Sierpinski right tetrahedron defined by $W = w_1 \cup w_2 \cup w_3 \cup w_4$ where w_i is defined as in (3) above. From the definitions for Q and L we see that

$$\begin{aligned}
w_1(Q(x,y,z)) &= w_1(y,z,x) = \left(\frac{1}{2}y, \frac{1}{2}z, \frac{1}{2}x\right) = Q\left(\frac{1}{2}x, \frac{1}{2}y, \frac{1}{2}z\right) = Q(w_1(x,y,z)) \\
w_2(Q(x,y,z)) &= w_2(y,z,x) = \left(\frac{1}{2}y + \frac{1}{2}, \frac{1}{2}z, \frac{1}{2}x\right) = Q\left(\frac{1}{2}x, \frac{1}{2}y + \frac{1}{2}, \frac{1}{2}z\right) = Q(w_4(x,y,z)) \\
w_3(Q(x,y,z)) &= w_3(y,z,x) = \left(\frac{1}{2}y, \frac{1}{2}z, \frac{1}{2}x + \frac{1}{2}\right) = Q\left(\frac{1}{2}x + \frac{1}{2}, \frac{1}{2}y, \frac{1}{2}z\right) = Q(w_2(x,y,z)) \\
w_4(Q(x,y,z)) &= w_4(y,z,x) = \left(\frac{1}{2}y, \frac{1}{2}z + \frac{1}{2}, \frac{1}{2}x\right) = Q\left(\frac{1}{2}x, \frac{1}{2}y, \frac{1}{2}z + \frac{1}{2}\right) = Q(w_3(x,y,z))
\end{aligned}$$

and

$$\begin{aligned}
w_1(L(x,y,z)) &= w_1(y,x,z) = \left(\frac{1}{2}y, \frac{1}{2}x, \frac{1}{2}z\right) = L\left(\frac{1}{2}x, \frac{1}{2}y, \frac{1}{2}z\right) = L(w_1(x,y,z)) \\
w_2(L(x,y,z)) &= w_2(y,x,z) = \left(\frac{1}{2}y + \frac{1}{2}, \frac{1}{2}x, \frac{1}{2}z\right) = L\left(\frac{1}{2}x, \frac{1}{2}y + \frac{1}{2}, \frac{1}{2}z\right) = L(w_4(x,y,z)) \\
w_3(L(x,y,z)) &= w_3(y,x,z) = \left(\frac{1}{2}y, \frac{1}{2}x, \frac{1}{2}z + \frac{1}{2}\right) = L\left(\frac{1}{2}x, \frac{1}{2}y, \frac{1}{2}z + \frac{1}{2}\right) = L(w_3(x,y,z)) \\
w_4(L(x,y,z)) &= w_4(y,x,z) = \left(\frac{1}{2}y, \frac{1}{2}x + \frac{1}{2}, \frac{1}{2}z\right) = L\left(\frac{1}{2}x + \frac{1}{2}, \frac{1}{2}y, \frac{1}{2}z\right) = L(w_2(x,y,z)).
\end{aligned}$$

So now we have

$$\begin{array}{ll} w_1Q = Qw_1 & w_1L = Lw_1 \\ w_2Q = Qw_4 & w_2L = Lw_4 \\ w_3Q = Qw_2 \quad \text{and} & w_3L = Lw_3 \\ w_4Q = Qw_3 & w_4L = Lw_2 \end{array}.$$

Let $A = w_1(A) \cup w_2(A) \cup w_3(A) \cup w_4(A)$ be the Sierpinski right tetrahedron. From the above equations we see that

$$\begin{aligned}
W(Q(A)) &= w_1(Q(A)) \cup w_2(Q(A)) \cup w_3(Q(A)) \cup w_4(Q(A)) \\
&= Q(w_1(A)) \cup Q(w_4(A)) \cup Q(w_2(A)) \cup Q(w_3(A)) \\
&= Q(w_1(A) \cup w_2(A) \cup w_3(A) \cup w_4(A)) \\
&= Q(A)
\end{aligned}$$

and

$$\begin{aligned}
W(L(A)) &= w_1(L(A)) \cup w_2(L(A)) \cup w_3(L(A)) \cup w_4(L(A)) \\
&= L(w_1(A)) \cup L(w_4(A)) \cup L(w_3(A)) \cup L(w_2(A)) \\
&= L(w_1(A) \cup w_2(A) \cup w_3(A) \cup w_4(A)) \\
&= L(A)
\end{aligned}$$

By uniqueness of the attractor of an IFS we can conclude that $Q(A) = L(A) = A$. So Q and L are symmetry operations on A. Furthermore, we can see that

$$\begin{aligned} R_1(x,y,z) &= (y,z,x) = Q(x,y,z) \\ R_2(x,y,z) &= (z,x,y) = Q^2(x,y,z) \\ R_3(x,y,z) &= (y,x,z) = L(x,y,z) \\ R_4(x,y,z) &= (x,z,y) = QL(x,y,x) \\ R_5(x,y,z) &= (z,y,x) = Q^2L(x,y,z) \end{aligned}$$

Therefore, all R_i are indeed symmetry operations on the Sierpinski right tetrahedron.

Let $\mathrm{Sym}(T) = \{e, Q, Q^2, L, QL, Q^2L\}$. Then $\mathrm{Sym}(T)$ is isomorphic to S_3 which is in turn isomorphic to D_3. □

4.4. **Classification of Attractors According to Their Symmetry Group.** The following theorems were devised and proven by Ken Monks. The definition for $\overline{i}$ can be found in definition 3.11 above.

Theorem 3.14. *Let $W \in \Psi$ such that $W(A) = A$. Let $\mathrm{Sym}(A)$ be the group of symmetry operations on A.*

$$\begin{aligned} \textit{Case I} &: \textit{If } |\mathrm{Sym}(A)| = 1 \textit{ then only } 1 \textit{ IFS in } \Psi \textit{ can produce } A. \\ \textit{Case II} &: \textit{If } |\mathrm{Sym}(A)| = 2 \textit{ then } 2^4 = 16 \textit{ distinct IFS's in } \Psi \textit{ can produce } A. \\ \textit{Case III} &: \textit{If } |\mathrm{Sym}(A)| = 3 \textit{ then } 3^4 = 81 \textit{ distinct IFS's in } \Psi \textit{ can produce } A. \\ \textit{Case IV} &: \textit{If } |\mathrm{Sym}(A)| = 6 \textit{ then } 6^4 = 1296 \textit{ distinct IFS's in } \Psi \textit{ can produce } A. \end{aligned}$$

Proof. Let $W, W' \in \Psi$ and $W(A) = A$. Assume $W'(A) = A$. By definition of Ψ, $W = v_1d_1 \cup v_2d_2 \cup v_3d_3 \cup v_4d_4$ and $W' = v_1d_1' \cup v_2d_2' \cup v_3d_3' \cup v_4d_4'$. By assumption,

$$v_1d_1(A) \cup v_2d_2(A) \cup v_3d_3(A) \cup v_4d_4(A) = A = v_1d_1'(A) \cup v_2d_2'(A) \cup v_3d_3'(A) \cup v_4d_4'(A)$$

Since the entire attractor must be contained within each v_i, we can reduce the problem to the separate equations $v_1d_1(A) = v_1d_1'(A)$, $v_2d_2(A) = v_2d_2'(A)$, $v_3d_3(A) = v_3d_3'(A)$, and $v_4d_4(A) = v_4d_4'(A)$. Furthermore, each v_i is invertible and therefore $d_1(A) = d_1'(A)$, $d_2(A) = d_2'(A)$, $d_3(A) = d_3'(A)$,and $d_4(A) = d_4'(A)$. Thus

$$A = d_1^{-1}d_1'(A),\ A = d_2^{-1}d_2'(A),\ A = d_3^{-1}d_3'(A) \text{ and } A = d_4^{-1}d_4'(A)$$

In order for this set of equations to be true, it must be the case that $\forall i \in 1..4$, $d_i^{-1}d_i' \in \mathrm{Sym}(A)$. Since we only need to count the possible number of different combinations of d_i' and not their specific values, we have 4 cases to examine.

Case 1: $|\mathrm{Sym}(A)| = 1$

$\mathrm{Sym}(A) = \{e\}$. Therefore $d_1^{-1}d_1' = d_2^{-1}d_2' = d_3^{-1}d_3' = d_4^{-1}d_4' = e$ and so $\forall i$, $d_i' = d_i$. Thus there is only one IFS that produces each asymmetric attractor.

Case 2: $|\mathrm{Sym}(A)| = 2$

$\mathrm{Sym}(A) = \{e, \alpha\}$ where $\alpha \in \{\alpha_1, \alpha_2, \alpha_3\}$. Therefore $d_1^{-1}d_1' \in \{e, \alpha\}$, $d_2^{-1}d_2' \in \{e, \alpha\}$, $d_3^{-1}d_3' \in \{e, \alpha\}$ and $d_4^{-1}d_4' \in \{e, \alpha\}$. Thus $\forall i$, $d_i' = d_i$ or $d_i' = \alpha d_i$. And so there are $2^4 = 16$ different IFS's that produce each attractor that is symmetric with respect to reflection across only one of the 3 diagonal planes.

Case 3: $|\mathrm{Sym}(A)| = 3$

$\mathrm{Sym}(A) = \{e, \alpha_4, \alpha_5\}$. Therefore as above $\forall i$, $d_i' = d_i$ or $d_i' = \alpha_4 d_i$ or $d_i' = \alpha_5 d_i$. Thus there are $3^4 = 81$ different IFS's that produce each attractor that is symmetric only with respect to rotation about $x = y = z$.

Case 4: $|\mathrm{Sym}(A)| = 6$

$\mathrm{Sym}(A) = \{e, \alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5\}$. As above $\forall i,\ d_i' = d_i$ or $d_i' = \alpha_1 d_i$ or $d_i' = \alpha_2 d_i$ or $d_i' = \alpha_3 d_i$ or $d_i' = \alpha_4 d_i$ or $d_i' = \alpha_5 d_i$. Therefore there are $6^4 = 1296$ different IFS's that produce each attractor having all six symmetries. □

Theorem 3.15. *Let $\alpha \in \Delta$ and let $W \in \Psi$ such that $A = W(A) = \bigcup_{i=1}^{4} v_i d_i(A)$.*

$$\alpha \in \mathrm{Sym}(A) \Leftrightarrow \forall i,\ d_i^{-1}\alpha d_{\bar{i}} \in \mathrm{Sym}(A)$$

Proof. Let $\alpha \in \Delta$ and let $W \in \Psi$ such that $A = W(A) = \bigcup_{i=1}^{4} v_i d_i(A)$.

(=>) Assume $\alpha \in \mathrm{Sym}(A)$.

$$\begin{aligned}
\alpha(A) &= A \\
\alpha(W(A)) &= W(A) \\
\alpha\left(\bigcup_{i=1}^{4} v_i d_i A\right) &= \bigcup_{i=1}^{4} v_i d_i A \\
\alpha\left(\bigcup_{i=1}^{4} v_{\bar{i}} d_{\bar{i}} A\right) &= \bigcup_{i=1}^{4} v_i d_i A \text{ By definition of } \bar{i} \\
\bigcup_{i=1}^{4} \alpha v_{\bar{i}} d_{\bar{i}} A &= \bigcup_{i=1}^{4} v_i d_i A \\
\bigcup_{i=1}^{4} v_i \alpha d_{\bar{i}} A &= \bigcup_{i=1}^{4} v_i d_i A
\end{aligned}$$

since the attractor is contained completely within each v_i we have

$$\begin{aligned}
\forall i,\ v_i \alpha d_{\bar{i}} A &= v_i d_i A \\
\forall i,\ \alpha d_{\bar{i}} A &= d_i A \\
\forall i,\ d_i^{-1}\alpha d_{\bar{i}} A &= A
\end{aligned}$$

Thus $\forall i,\ d_i^{-1}\alpha d_{\bar{i}} \in \mathrm{Sym}(A)$.

(<=) Assume $\forall i,\ d_i^{-1}\alpha d_{\overline{i}} \in \mathrm{Sym}\,(A)$. Then $\forall i,\ d_i^{-1}\alpha d_{\overline{i}} = \beta_i$ for some $\beta_i \in \mathrm{Sym}\,(A)$ and so $\forall i,\ \alpha d_{\overline{i}} = d_i\beta_i$.

$$\begin{aligned}
\alpha(A) &= \alpha(W(A)) \\
&= \alpha\left(\bigcup_{i=1}^{4} v_i d_i A\right) \\
&= \alpha\left(\bigcup_{i=1}^{4} v_{\overline{i}} d_{\overline{i}} A\right) \\
&= \bigcup_{i=1}^{4} \alpha v_{\overline{i}} d_{\overline{i}} A \\
&= \bigcup_{i=1}^{4} v_i \alpha d_{\overline{i}} A \\
&= \bigcup_{i=1}^{4} v_i d_i \beta_i A \\
&= \bigcup_{i=1}^{4} v_i d_i A \\
&= W(A) \\
&= A
\end{aligned}$$

Thus $\alpha \in \mathrm{Sym}\,(A)$. □

Theorem 3.16. *Let $\alpha \in \Delta$ and let $W \in \Psi$ such that $A = W(A) = \bigcup_{i=1}^{4} v_i d_i(A)$.*

$$\textit{If } \forall i,\ \alpha d_{\overline{i}} = d_i\alpha \textit{ then } \alpha \in \mathrm{Sym}\,(A)$$

Proof. Let $\alpha \in \Delta$ and let $W \in \Psi$ such that $A = W(A) = \bigcup_{i=1}^{4} v_i d_i(A)$. Assume $\forall i,\ \alpha d_{\overline{i}} = d_i\alpha$ then

$$\begin{aligned}
\alpha(A) &= \alpha(W(A)) \\
&= \alpha \bigcup_{i=1}^{4} v_i d_i(A) \\
&= \bigcup_{i=1}^{4} \alpha v_{\overline{i}} d_{\overline{i}}(A) \\
&= \bigcup_{i=1}^{4} v_i \alpha d_{\overline{i}}(A) \\
&= \bigcup_{i=1}^{4} v_i d_i \alpha(A) \\
&= W(\alpha(A))
\end{aligned}$$

Since the attractor of an IFS is unique, it follows that $\alpha A = A$ and so $\alpha \in \mathrm{Sym}\,(A)$. □

Theorem 3.17. *Let $\alpha \in \Delta$ and let $W, W' \in \Psi$ such that $W = \bigcup_{i=1}^{4} v_i d_i$, $W'(A) = \bigcup_{i=1}^{4} v_i d_i'(A)$ and $A = W'(A)$.*

$$A = W(A) \Leftrightarrow \forall i, d_i^{-1} d_i' \in \mathrm{Sym}(A)$$

Proof. Let $\alpha \in \Delta$ and let $W, W' \in \Psi$ such that $W = \bigcup_{i=1}^{4} v_i d_i$ and $A = W'(A) = \bigcup_{i=1}^{4} v_i d_i'(A)$

(=>) Assume $W(A) = A$

$$\begin{aligned} W(A) &= W'(A) \\ \bigcup_{i=1}^{4} v_i d_i A &= \bigcup_{i=1}^{4} v_i d_i' A \\ \forall i,\ v_i d_i A &= v_i d_i' A \\ \forall i,\ d_i A &= d_i' A \\ \forall i,\ A &= d_i^{-1} d_i' A \end{aligned}$$

therefore, $\forall i,\ d_i^{-1} d_i' \in \mathrm{Sym}(A)$.

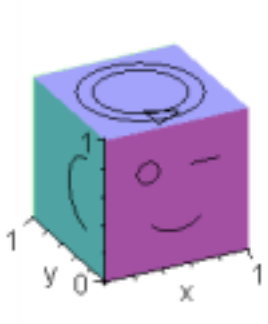
(<=) Assume $\forall i,\ d_i^{-1} d_i' \in \mathrm{Sym}(A)$. Then $\forall i,\ A = d_i^{-1} d_i' A$ and so by substitution we have

$$\begin{aligned} W(A) &= \bigcup_{i=1}^{4} v_i d_i(A) \\ &= \bigcup_{i=1}^{4} v_i d_i \left(d_i^{-1} d_i' A\right) \\ &= \bigcup_{i=1}^{4} v_i d_i' A \\ &= W'(A) \\ &= A \end{aligned}$$

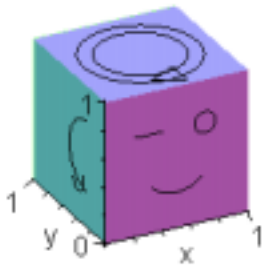
Thus $W(A) = A$. □

5. Appendix A

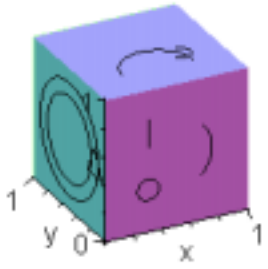
5.1. **Symmetry Group Notation and CubeIFS Input Table.** The group of 48 symmetries of the unit cube can be generated by the five following elements.



e = starting position, no rotation or reflection

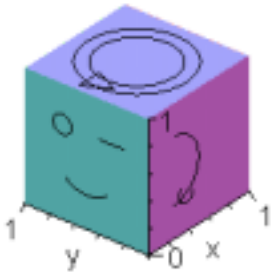


$-$ = reflection through the plane parallel to the yz plane that cuts the cube in half



r

s



t

$r = 90$ deg rotation about the axis through the center of the cube from $\left(\frac{1}{2}, 0, \frac{1}{2}\right)$ to $\left(\frac{1}{2}, 1, \frac{1}{2}\right)$ CCW when looking down the y axis from $-$ to $+$.

$s = 90$ deg rotation about the axis through the center of the cube from $\left(0, \frac{1}{2}, \frac{1}{2}\right)$ to $\left(1, \frac{1}{2}, \frac{1}{2}\right)$ CCW when looking down the x axis from $-$ to $+$.

$t = 90$ deg rotation about the axis through the center of the cube from $\left(\frac{1}{2}, \frac{1}{2}, 0\right)$ to $\left(\frac{1}{2}, \frac{1}{2}, 1\right)$ CCW when looking down the z axis from $-$ to $+$.

Symmetry group elements are defined by composing the generator operations from left to right.

For example:
$-r^2s$ means first reflection followed by 180 degrees rotation in the r direction (literally it means to perform r twice) followed by 90 degrees rotation in the s direction.

The following chart lists all 48 elements of the group. The labels along the top row and left column are equivalent notations that may be used in the CubeIFS routine that is explained in Appendix D.

	F	H	N	R	L	B
Up	e	s	s^3	t	t^3	r^2s^2
-Up	$-e$	$-s$	$-s^3$	$-t$	$-t^3$	$-r^2s^2$
Lt	r	rs	rs^3	rt	rt^3	r^3s^2
-Lt	$-r$	$-rs$	$-rs^3$	$-rt$	$-rt^3$	$-r^3s^2$
Dn	r^2	r^2s	r^2s^3	r^2t	r^2t^3	s^2
-Dn	$-r^2$	$-r^2s$	$-r^2s^3$	$-r^2t$	$-r^2t^3$	$-s^2$
Rt	r^3	r^3s	r^3s^3	r^3t	r^3t^3	rs^2
-Rt	$-r^3$	$-r^3s$	$-r^3s^3$	$-r^3t$	$-r^3t^3$	$-rs^2$

Definition 5.1. *Let G be a group. The* center of G *is the subgroup*

$$Z(G) = \{z \in G : zd = dz \text{ for every } d \in G\}.$$

The center of our group of symmetry operations on the cube is $\left\{e, -s^2\right\}$.

Definition 5.2. *Let G be a group and $\alpha \in G$. The* centralizer of α *is the subgroup*

$$C(\alpha) = \{d \in G : d\alpha = \alpha d\}.$$

As it turns out, the groups of centralizers of $\alpha \in \Delta$ (and sometimes a coset of one of these groups) contain key elements for defining IFS's $W \in \Psi$ having symmetric attractors. Some particular definitions that we know of so far are given in Appendix E. Here we list the groups of centralizers for each $\alpha \in \Delta$ (7).

$$\begin{aligned}
C(\alpha_1) &= C(-t) = \left\{e, r^2s^2, r^2t^3, r^2t, -r^2, -s^2, -t, -t^3\right\} \\
C(\alpha_2) &= C\left(-r^2s^3\right) = \left\{e, -e, r^2s^3, -r^2s^3, r^2s, -r^2s, s^2, -s^2\right\} \\
C(\alpha_3) &= C\left(-r^3\right) = \left\{e, r^3s^2, r^2, rs^2, -r, -r^3, -s^2, -r^2s^2\right\} \\
C(\alpha_4) &= C(rt) = C(\alpha_5) = C\left(r^3s^3\right) = \left\{e, r^3s^3, rt, -rs, -s^2, -rt^3\right\}
\end{aligned}$$

As you can see, for $i \in \{1, 2, 3\}$, $|C(\alpha_i)| = 8$ and for $j \in \{4, 5\}$, $|C(\alpha_j)| = 6$. It is also helpful to note that the only elements common among the centralizer groups are the elements of the center, $\left\{e, -s^2\right\}$.

Definition 5.3. *Let H be a subgroup of a group G. The* normalizer of H *is the set* $N_H = \left\{d \in G : d^{-1}Hd = H\right\}$.

It is well known that N_H is a subgroup of G that contains H.

Lemma 1. *Let* $W = \bigcup_{i=1}^{4} v_i d_i \in \Psi$ *such that* $A = W(A)$. *If* $|\operatorname{Sym}(A)| = 6$, *and hence* $\operatorname{Sym}(A) = \Delta$, *then* $d_1 \in N_\Delta$.

Here we list the normalizer of Δ and the normalizer of $C(\alpha_i)$ for all $i \in \{1,2,3,4,5\}$ since they have been useful in determining the definitions of symmetric attractors in Ψ.

$$\begin{aligned}
N_\Delta &= N_{C(\alpha_4)} = N_{C(\alpha_5)} = \left\{e, rs^2, r^2s, r^3s^3, rt, r^2t^3, -r^3, -s^2, -t, -rs, -r^2s^3, -rt^3\right\}\\
N_{C(\alpha_1)} &= \left\{e, r^2, s^2, t, t^3, r^2s^2, r^2t, r^2t^3, -e, -r^2, -s^2, -t, -t^3, -r^2s^2, -r^2t, -r^2t^3\right\}\\
N_{C(\alpha_2)} &= \left\{e, r^2, s, s^2, s^3, r^2s, r^2s^2, r^2s^3, -e, -r^2, -s, -s^2, -s^3, -r^2s, -r^2s^2, -r^2s^3\right\}\\
N_{C(\alpha_3)} &= \left\{e, r, r^2, r^3, s^2, rs^2, r^2s^2, r^3s^2, -e, -r, -r^2, -r^3, -s^2, -rs^2, -r^2s^2, -r^3s^2\right\}
\end{aligned}$$

Notice here that $|N_\Delta| = 12$ and for $i \in \{1,2,3\}$, $|N_{C(\alpha_i)}| = 16$. Here also, the only elements common to all normalizer groups are $\left\{e, -s^2\right\}$. However, the elements common to $N_{C(\alpha_i)}$ for $i \in \{1,2,3\}$ are $\left\{e, r^2, s^2, r^2s^2, -e, -r^2, -s^2, -r^2s^2\right\}$. Additionally notice that N_Δ and $N_{C(\alpha_1)}$ share $\left\{e, -s^2, r^2t^3, -t\right\}$, N_Δ and $N_{C(\alpha_2)}$ share $\left\{e, -s^2, r^2s, -r^2s^3\right\}$, and N_Δ and $N_{C(\alpha_3)}$ share $\left\{e, -s^2, rs^2, -r^3\right\}$.

6. Appendix B

6.1. **Affine Forms in $\mathbb{R}^3$ for Use in the Chaos3d Maple Module.** The following representations of affine maps are presented for use in the Maple Chaos3d module as structures for defining an affine map in $\mathbb{R}^3$. The *Vector, Spherical,* and *Geometric* forms are the three core methods for defining an affine map, $T(x) = Mx + B$. For the convenience of maple users I additionally define the *Matrix* form and *Standard* form data structures to allow greater flexibility in the syntax used to create an affine map using known values for M and B.

An affine map in $\mathbb{R}^3$ is completely determined by where it sends four non-coplanar points. The simplest points to consider are $(1,0,0)$, $(0,1,0)$, $(0,0,1)$, and the origin. Therefore, it is possible to define an affine map by its effect on the three base vectors, $\vec{i}$, $\vec{j}$, $\vec{k}$, and the translation of the origin. If we know that the origin is mapped to point t, and we know that point $(1,0,0)$ is mapped to point p_i, then we can calculate $p_i - t$ which we call $\vec{v}_i$, the image of $\vec{i}$ under the affine map before translation. A similar computation works for determining $\vec{v}_j$ and $\vec{v}_k$whose values are used in the affineV3d data structure. For convenience in the Chaos3d Maple module, these values may also be given by spherical coordinates using the affineS3d data structure.

Definition 6.1. *Let $T : \mathbb{R}^3 \to \mathbb{R}^3$ given by $T(x) = Mx + B$. where $M \in M_3(\mathbb{R})$ defines the images of the basis vectors, $\vec{i}$, $\vec{j}$, and $\vec{k}$ under the affine map before translation, and $B \in \mathbb{R}^3$ is the translation map on the origin. The following data structures define equivalent affine maps.*

Vector Form

Let

$$\vec{v}_i = \begin{bmatrix} a_i \\ b_i \\ c_i \end{bmatrix}, \vec{v}_j = \begin{bmatrix} a_j \\ b_j \\ c_j \end{bmatrix}, \vec{v}_k = \begin{bmatrix} a_k \\ b_k \\ c_k \end{bmatrix}, \vec{t} = \begin{bmatrix} l \\ m \\ n \end{bmatrix}$$

affineV3d$(\vec{v}_i, \vec{v}_j, \vec{v}_k, \vec{t})$ *defines*

$$T\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{bmatrix} a_i & a_j & a_k \\ b_i & b_j & b_k \\ c_i & c_j & c_k \end{bmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{bmatrix} l \\ m \\ n \end{bmatrix} \tag{8}$$

Spherical Form

Let $r, s,$ and $t \in \mathbb{R}$ be scaling coefficients for $\vec{i}, \vec{j},$ and $\vec{k}$ respectively. Let $\theta_i, \theta_j, \theta_k, \phi_i, \phi_j,$and ϕ_k be angle measurements in degrees. Then according to standard spherical coordinates, affineS3d $\left(r, s, t, \theta_i, \phi_i, \theta_j, \phi_j, \theta_k, \phi_k, l, m, n\right)$ *defines*

$$\begin{aligned} T\begin{pmatrix} x \\ y \\ z \end{pmatrix} &= \begin{bmatrix} \cos\theta_i \sin\phi_i & \cos\theta_j \sin\phi_j & \cos\theta_k \sin\phi_k \\ \sin\theta_i \sin\phi_i & \sin\theta_j \sin\phi_j & \sin\theta_k \sin\phi_k \\ \cos\phi_i & \cos\phi_j & \cos\phi_k \end{bmatrix} \begin{bmatrix} r & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & t \end{bmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{bmatrix} l \\ m \\ n \end{bmatrix} \\ &= \begin{bmatrix} r\cos\theta_i \sin\phi_i & s\cos\theta_j \sin\phi_j & t\cos\theta_k \sin\phi_k \\ r\sin\theta_i \sin\phi_i & s\sin\theta_j \sin\phi_j & t\sin\theta_k \sin\phi_k \\ r\cos\phi_i & s\cos\phi_j & t\cos\phi_k \end{bmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{bmatrix} l \\ m \\ n \end{bmatrix} \end{aligned} \tag{9}$$

Standard Form
affine3d $(a_1, a_2, a_3, b_1, b_2, b_3, c_1, c_2, c_3, t_1, t_2, t_3)$ *defines*

$$T\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{bmatrix}\begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} \tag{10}$$

Matrix Form
Let M be a Maple-defined "Matrix" or "matrix". Let B be a Maple-defined "Vector"
affineM3d (M, B) *defines*

$$T(x) = Mx + B \tag{11}$$

Any three-dimensional affine map will stretch and twist the unit cube into any desired parallelepiped fixing the origin and then translate that shape to any coordinate in the $\mathbb{R}^3$ plane. However, for many applications it may be useful to rotate a cube-like figure uniformly around one axis without destroying its orthogonality. For this purpose, the affineG3d data structure produces an affine map that first scales $\vec{i}$, $\vec{j}$, and $\vec{k}$ by real-valued constants r, s,and t then consecutively rotates this orthogonal parallelepiped about the x, y, and z axes by angles θ, ϕ,and γ respectively before translating the origin to B.

Definition 6.2. *Let $T : \mathbb{R}^3 \to \mathbb{R}^3$ given by $T(x) = Mx + B$. where $M \in M_3(\mathbb{R})$ defines the images of the basis vectors, $\vec{i}$, $\vec{j}$, and $\vec{k}$ under the affine map before translation, and $B \in \mathbb{R}^3$ is the translation map on the origin. Let θ, ϕ, γ be angle measurements in degrees and let*

$$\begin{aligned} M &= \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}\begin{bmatrix} r & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & t \end{bmatrix} \\ &= \begin{bmatrix} r\cos\phi\cos\gamma & s\sin\theta\sin\phi\cos\gamma - s\cos\theta\sin\gamma & t\cos\theta\sin\phi\cos\gamma + t\sin\theta\sin\gamma \\ r\cos\phi\sin\gamma & s\sin\theta\sin\phi\sin\gamma + s\cos\theta\cos\gamma & t\cos\theta\sin\phi\sin\gamma - t\sin\theta\cos\gamma \\ -r\sin\phi & s\sin\theta\cos\phi & t\cos\theta\cos\phi \end{bmatrix} \end{aligned} \tag{12}$$

The data structure that defines an affine map for any orthogonal parallelepiped is given in ***Geometric form*** *by* affineG3d$(r, s, t, \theta, \phi, \gamma, l, m, n)$ *which defines*

$$T\begin{pmatrix} x \\ y \\ z \end{pmatrix} = M\begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{bmatrix} l \\ m \\ n \end{bmatrix} \tag{13}$$

7. Appendix C

7.1. **Chaos3d Maple Module.** The following Maple programming was written as a three-dimensional extension to the Chaos module written by Ken Monks for working with IFS's and their attractors in $\mathbb{R}^2$. The Chaos3d module uses the five new affine data structures defined in Appendix B in order to define IFS's on $\mathbb{R}^3$ and then render their attractors either randomly or deterministically. A procedure is included for calculating a scaling/contraction factor for any affine map. The CubeIFS procedure is very useful for studying the family, Ψ, of relatives to the Sierpinski right tetrahedron as defined by 3.10, and other cube-based IFS's. The basic IFS definitions for the Menger Sponge and the standard Sierpinski Tetrahedron are stored within the Chaos3d module and can be used within any procedure accepting IFS input. Multiple seed figures are also stored for use with the DrawDetIFS3d command for rendering an attractor. One procedure is included for writing a POV-Ray (Persistence Of Vision Ray tracing) file to store data for small spheres corresponding to points in the attractor of an IFS. However, this procedure uses a random iteration method. To produce a more accurate fractal image, the procedure should be altered to use a deterministic method.

Detailed explanations of the procedure calls and how to use them are given in Appendix D.

```
# Define data types
#
# 3d Vector form       -- affineV3d(vi,vj,vk,T)
# 3d Spherical form    -- affineS3d(r,s,t,thetaY,thetaZ,phiX,phiZ,gammaX,gammaY,l,m,n)
# 3d Geometric form    -- affineS3d(r,s,t,theta,phi,gamma,l,m,n)
# 3d Matrix form       -- affineM3d(M,B)
# 3d Standard form     -- affine3d(a1,a2,a3,b1,b2,b3,c1,c2,c3,t1,t2,t3)
# IFS = List of affines (of any type)

chaos3d:=module()
export
   affineV3dFromPoints,ContractionFactor3d,IFSFromList,CubeIFS,MengerSponge,
   SierpinskiTetrahedron,MrBlockHead,MrBlockFrame,MrCorner,MrBlock,MrPyramid,MrPoint3d,
   DrawDetIFS3d,DrawIFS3d,WriteIFSPOV;
global
   `convert/affineV3d`,`convert/affineS3d`,`convert/affineM3d`,
   `convert/affine3d`,`convert/affine3df`:
local
   affineV3d2affineS3d,affineS3d2affineV3d,affineV3d2affineM3d,affineM3d2affineV3d,
   affineG3d2affineV3d,affine3d2affineM3d,affine3d2affineV3d,MapOne3d,
   Map3d,Transform3d,ApplyIFS3d;

with(plots):
with(plottools):
protect(affineS3d,affineG3d,affineV3d,affineM3d,affine3d,affine3df):

# internal affine form conversion

affineV3d2affineS3d:=proc(vi,vj,vk,T)
   local r,s,t,thetaI,phiI,thetaJ,phiJ,thetaK,phiK,l,m,n:
```

```
   r:= sqrt(op(1,vi)^2+op(2,vi)^2+op(3,vi)^2);
   phiI:= arccos(op(3,vi)/r);
   thetaI:= arctan(op(2,vi),op(1,vi));
   s:= sqrt(op(1,vj)^2+op(2,vj)^2+op(3,vj)^2);
   phiJ:= arccos(op(3,vj)/s);
   thetaJ:= arctan(op(2,vj),op(1,vj));
   t:= sqrt(op(1,vk)^2+op(2,vk)^2+op(3,vk)^2);
   phiK:= arccos(op(3,vk)/t);
   thetaK:= arctan(op(2,vk),op(1,vk));
   l:=op(1,T);
   m:=op(2,T);
   n:=op(3,T);
   RETURN(r,s,t,evalf(thetaI*180/Pi),evalf(phiI*180/Pi),
            evalf(thetaJ*180/Pi),evalf(phiJ*180/Pi),
            evalf(thetaK*180/Pi),evalf(phiK*180/Pi),l,m,n);
end:

affineS3d2affineV3d:=proc(r,s,t,thetaI,phiI,thetaJ,phiJ,thetaK,phiK,l,m,n)
   local vi,vj,vk,T:
   vi:=[evalf(r*cos(thetaI*(Pi/180))*sin(phiI*(Pi/180))),
      evalf(r*sin(thetaI*(Pi/180))*sin(phiI*(Pi/180))),
      evalf(r*cos(phiI*(Pi/180)))];
   vj:=[evalf(s*cos(thetaJ*(Pi/180))*sin(phiJ*(Pi/180))),
      evalf(s*sin(thetaJ*(Pi/180))*sin(phiJ*(Pi/180))),
      evalf(s*cos(phiJ*(Pi/180)))];
   vk:=[evalf(t*cos(thetaK*(Pi/180))*sin(phiK*(Pi/180))),
      evalf(t*sin(thetaK*(Pi/180))*sin(phiK*(Pi/180))),
      evalf(t*cos(phiK*(Pi/180)))];
   T:=[l,m,n];
   RETURN(vi,vj,vk,T);
end:

affineG3d2affineV3d:=proc(r,s,t,theta,phi,gamma,l,m,n)
   local vi,vj,vk,T:
   vi:=[evalf(r*cos(phi*Pi/180)*cos(gamma*Pi/180)),
      evalf(r*cos(phi*Pi/180)*sin(gamma*Pi/180)),
      evalf(-r*sin(phi*Pi/180))];
   vj:=[evalf(s*sin(theta*Pi/180)*sin(phi*Pi/180)*cos(gamma*Pi/180)
         -s*cos(theta*Pi/180)*sin(gamma*Pi/180)),
      evalf(s*sin(theta*Pi/180)*sin(phi*Pi/180)*sin(gamma*Pi/180)
         +s*cos(theta*Pi/180)*cos(gamma*Pi/180)),
      evalf(s*sin(theta*Pi/180)*cos(phi*Pi/180))];
   vk:=[evalf(t*cos(theta*Pi/180)*sin(phi*Pi/180)*cos(gamma*Pi/180)
         +t*sin(theta*Pi/180)*sin(gamma*Pi/180)),
      evalf(t*cos(theta*Pi/180)*sin(phi*Pi/180)*sin(gamma*Pi/180)
         -t*sin(theta*Pi/180)*cos(gamma*Pi/180)),
      evalf(t*cos(theta*Pi/180)*cos(phi*Pi/180))];
   T:=[l,m,n];
```

```
   RETURN(vi,vj,vk,T);
end:

affineV3d2affineM3d:=proc(vi,vj,vk,T)
   local M,B:
   M:=Matrix([[op(1,vi),op(2,vi),op(3,vi)],[op(1,vj),
      op(2,vj),op(3,vj)],[op(1,vk),op(2,vk),op(3,vk)]],scan=columns);
   B:=Vector([op(1,T),op(2,T),op(3,T)]);
   RETURN(M,B);
end:

affineM3d2affineV3d:=proc(M,B)
   local vi,vj,vk,T:
   vi:=[M[1,1],M[2,1],M[3,1]];
   vj:=[M[1,2],M[2,2],M[3,2]];
   vk:=[M[1,3],M[2,3],M[3,3]];
   T :=[B[1],B[2],B[3]];
   RETURN(vi,vj,vk,T);
end:

affine3d2affineM3d:=proc(a1,a2,a3,b1,b2,b3,c1,c2,c3,t1,t2,t3)
   local M,B:
   M:=Matrix([[a1,a2,a3],[b1,b2,b3],[c1,c2,c3]]):
   B:=Vector([t1,t2,t3]):
   RETURN(M,B);
end:

affine3d2affineV3d:=proc(a1,a2,a3,b1,b2,b3,c1,c2,c3,t1,t2,t3)
   local vi,vj,vk,T:
   vi:=[a1,b1,c1]:
   vj:=[a2,b2,c2]:
   vk:=[a3,b3,c3]:
   T:=[t1,t2,t3]:
   RETURN(vi,vj,vk,T);
end:

# user affine form conversion

`convert/affineM3d`:=proc(A)
   local nm:
   if type(A,list) then RETURN(map(`convert/affineM3d`,A)) fi;
   nm:=op(0,A):
   if nm=affineS3d then
      RETURN(affineM3d(affineV3d2affineM3d(affineS3d2affineV3d(op(A)))))
   elif nm=affineG3d then
      RETURN(affineM3d(affineV3d2affineM3d(affineG3d2affineV3d(op(A)))))
   elif nm=affineV3d then
      RETURN(affineM3d(affineV3d2affineM3d(op(A))))
```

```
   elif nm=affine3d then
      RETURN(affineM3d(affine3d2affineM3d(op(A))))
   else RETURN(A)
   fi;
end:

‘convert/affineV3d‘:=proc(A)
   local nm;
   if type(A,list) then RETURN(map(‘convert/affineV3d‘,A)) fi;
   nm:=op(0,A):
   if nm=affineS3d then
      RETURN(affineV3d(affineS3d2affineV3d(op(A))))
   elif nm=affineG3d then
      RETURN(affineV3d(affineG3d2affineV3d(op(A))))
   elif nm=affineM3d then
      RETURN(affineV3d(affineM3d2affineV3d(op(A))))
   elif nm=affine3d then
      RETURN(affineV3d(affine3d2affineV3d(op(A))))
   else RETURN(A)
   fi;
end:

‘convert/affineS3d‘:=proc(A)
   local nm:
   if type(A,list) then RETURN(map(‘convert/affineS3d‘,A)) fi;
   nm:=op(0,A):
   if nm=affineV3d then
      RETURN(affineS3d(affineV3d2affineS3d(op(A))))
   elif nm=affine3d then
      RETURN(affineS3d(affineV3d2affineS3d(affine3d2affineV3d(op(A)))))
   elif nm=affineM3d then
      RETURN(affineS3d(affineV3d2affineS3d(affineM3d2affineV3d(op(A)))))
   elif nm=affineG3d then
      RETURN(affineS3d(affineV3d2affineS3d(affineG3d2affineV3d(op(A)))))
   else RETURN(A)
   fi;
end:

‘convert/affine3d‘:=proc(A)
   local nm,aff;
   if type(A,list) then RETURN(map(‘convert/affine3d‘,A)) fi;
   nm:=op(0,A):
   if nm=affineM3d then
      RETURN(affine3d(op(1,A)[1,1],op(1,A)[1,2],op(1,A)[1,3],
         op(1,A)[2,1],op(1,A)[2,2],op(1,A)[2,3],op(1,A)[3,1],
         op(1,A)[3,2],op(1,A)[3,3],op(2,A)[1],op(2,A)[2],op(2,A)[3]))
   elif nm=affineV3d then
      RETURN(affine3d(op(1,A)[1],op(2,A)[1],op(3,A)[1],op(1,A)[2],
```

```
         op(2,A)[2],op(3,A)[2],op(1,A)[3],op(2,A)[3],
         op(3,A)[3],op(4,A)[1],op(4,A)[2],op(4,A)[3]))
   elif nm=affineS3d then
      aff:=affineV3d(affineS3d2affineV3d(op(A))):
      RETURN(affine3d(op(1,aff)[1],op(2,aff)[1],op(3,aff)[1],
         op(1,aff)[2],op(2,aff)[2],op(3,aff)[2],op(1,aff)[3],
         op(2,aff)[3],op(3,aff)[3],op(4,aff)[1],op(4,aff)[2],op(4,aff)[3]))
   elif nm=affineG3d then
      aff:=affineV3d(affineG3d2affineV3d(op(A))):
      RETURN(affine3d(op(1,aff)[1],op(2,aff)[1],op(3,aff)[1],
         op(1,aff)[2],op(2,aff)[2],op(3,aff)[2],op(1,aff)[3],
         op(2,aff)[3],op(3,aff)[3],op(4,aff)[1],op(4,aff)[2],op(4,aff)[3]))
   else RETURN(A)
   fi;
end:

`convert/affine3df`:=proc(A)
   if type(A,list) then RETURN(map(`convert/affine3df`,A)) fi;
   map(evalf,convert(A,affine3d)):
end:

# alternate affine map definition

affineV3dFromPoints:=proc(p1,p2,p3,p4,Tp1,Tp2,Tp3,Tp4)
   local x1,y1,z1,x2,y2,z2,x3,y3,z3,x4,y4,z4,
      ai,bi,ci,aj,bj,cj,ak,bk,ck,l,m,n,
      e1,e2,e3,e4,e5,e6,e7,e8,e9,e10,e11,e12:
   x1:=p1[1];   y1:=p1[2];   z1:=p1[3];
   x2:=p2[1];   y2:=p2[2];   z2:=p2[3];
   x3:=p3[1];   y3:=p3[2];   z3:=p3[3];
   x4:=p4[1];    y4:=p4[2];    z4:=p4[3];
   e1:=ai*x1+aj*y1+ak*z1+l=Tp1[1];
   e2:=bi*x1+bj*y1+bk*z1+m=Tp1[2];
   e3:=ci*x1+cj*y1+ck*z1+n=Tp1[3];
   e4:=ai*x2+aj*y2+ak*z2+l=Tp2[1];
   e5:=bi*x2+bj*y2+bk*z2+m=Tp2[2];
   e6:=ci*x2+cj*y2+ck*z2+n=Tp2[3];
   e7:=ai*x3+aj*y3+ak*z3+l=Tp3[1];
   e8:=bi*x3+bj*y3+bk*z3+m=Tp3[2];
   e9:=ci*x3+cj*y3+ck*z3+n=Tp3[3];
   e10:=ai*x4+aj*y4+ak*z4+l=Tp4[1];
   e11:=bi*x4+bj*y4+bk*z4+m=Tp4[2];
   e12:=ci*x4+cj*y4+ck*z4+n=Tp4[3];
   assign(solve({e1,e2,e3,e4,e5,e6,e7,e8,e9,e10,e11,e12}));
   affineV3d([ai,bi,ci],[aj,bj,cj],[ak,bk,ck],[l,m,n]);
end:

# calculate scaling factor of an affine map
```

```
ContractionFactor3d:=proc(aff)
   local MV,Mt,E,A:
   if type(aff,list) then
      RETURN(map(‘ContractionFactor3d‘,aff))
   fi:
   MV:=op(convert(aff,affineM3d)):
   Mt:=LinearAlgebra[Transpose](MV[1]):
   A:=Matrix(evalm(Mt*MV[1])):
   E:= linalg[’eigenvals’](A):
   RETURN(Re(evalf(sqrt(E[-1]))));
end:

# methods for defining an IFS

IFSFromList:=proc(List)    # assumed to be a list of affineV3d coefficients
   map(x->affineV3d(op(x)),List);
end:

CubeIFS:=proc()
   local check,tcheck,n,m,p,i,j,k,Lst,T,data,h,A,lab,d,names:
   check:=evalf(nargs^1/3));   tcheck:=trunc(check);
   if not check=tcheck then
      ERROR(’’The number of arguments must be a perfect cube.’’)
   fi:
   n:=tcheck:   m:=1:   Lst:=NULL;   p:=1/n;
   A:={args};   data:=[args];
   if not hastype(A,list) then
      A:=[args];   data:=[];
      lab:=[e,-e,r,-r,r2,-r2,r3,-r3,s,-s,rs,-rs,r2s,-r2s,r3s,-r3s,
         s2,-s2,rs2,-rs2,r2s2,-r2s2,r3s2,-r3s2,s3,-s3,rs3,-rs3,
         r2s3,-r2s3,r3s3,-r3s3,t,-t,rt,-rt,r2t,-r2t,r3t,-r3t,t3,-t3,rt3,
         -rt3,r2t3,-r2t3,r3t3,-r3t3];
      names:=[
         [e,[F,Up]],[-e,[F,-Up]],[r,[F,Lt]],[-r,[F,-Lt]],
         [r2,[F,Dn]],[-r2,[F,-Dn]],[r3,[F,Rt]],[-r3,[F,-Rt]],
         [s,[H,Up]],[-s,[H,-Up]],[rs,[H,Lt]],[-rs,[H,-Lt]],
         [r2s,[H,Dn]],[-r2s,[H,-Dn]],[r3s,[H,Rt]],[-r3s,[H,-Rt]],
         [s2,[B,Dn]],[-s2,[B,-Dn]],[rs2,[B,Rt]],[-rs2,[B,-Rt]],
         [r2s2,[B,Up]],[-r2s2,[B,-Up]],[r3s2,[B,Lt]],[-r3s2,[B,-Lt]],
         [s3,[N,Up]],[-s3,[N,-Up]],[rs3,[N,Lt]],[-rs3,[N,-Lt]],
         [r2s3,[N,Dn]],[-r2s3,[N,-Dn]],[r3s3,[N,Rt]],[-r3s3,[N,-Rt]],
         [t,[R,Up]],[-t,[R,-Up]],[rt,[R,Lt]],[-rt,[R,-Lt]],
         [r2t,[R,Dn]],[-r2t,[R,-Dn]],[r3t,[R,Rt]],[-r3t,[R,-Rt]],
         [t3,[L,Up]],[-t3,[L,-Up]],[rt3,[L,Lt]],[-rt3,[L,-Lt]],
         [r2t3,[L,Dn]],[-r2t3,[L,-Dn]],[r3t3,[L,Rt]],[-r3t3,[L,-Rt]]];
      for h to nargs do
         if member(A[h],lab) then
```

```
         for d to 48 do
            if A[h]=names[d][1] then
               data:=[op(data),names[d][2]];
               break;
            fi;
         od;
      else
         data:=[op(data),A[h]];
      fi;
   od;
fi;
for j from 0 to n-1 do
   for k from 0 to n-1 do
      for i from 0 to n-1 do
         if op(1,data[m])=F then
            if op(2,data[m])=Up then
               T:=affineM3d(Matrix(Vector([p,p,p]),
                  shape=diagonal),Vector([i*p,j*p,k*p])):
            elif op(2,data[m])=-Up then
               T:=affineM3d(Matrix(Vector([-p,p,p]),
                  shape=diagonal),Vector([(i+1)*p,j*p,k*p])):
            elif op(2,data[m])=Dn then
               T:=affineM3d(Matrix(Vector([-p,p,-p]),
                  shape=diagonal),Vector([(i+1)*p,j*p,(k+1)*p])):
            elif op(2,data[m])=-Dn then
               T:=affineM3d(Matrix(Vector([p,p,-p]),
                  shape=diagonal),Vector([i*p,j*p,(k+1)*p])):
            elif op(2,data[m])=Rt then
               T:=affineM3d(Matrix([[0,0,-p],[0,p,0],[p,0,0]],
                  scan=columns),Vector([i*p,j*p,(k+1)*p])):
            elif op(2,data[m])=-Rt then
               T:=affineM3d(Matrix([[0,0,p],[0,p,0],[p,0,0]],
                  scan=columns),Vector([i*p,j*p,k*p])):
            elif op(2,data[m])=Lt then
               T:=affineM3d(Matrix([[0,0,p],[0,p,0],[-p,0,0]],
                  scan=columns),Vector([(i+1)*p,j*p,k*p])):
            elif op(2,data[m])=-Lt then
               T:=affineM3d(Matrix([[0,0,-p],[0,p,0],[-p,0,0]],
                  scan=columns),Vector([(i+1)*p,j*p,(k+1)*p])):
            else T:=NULL;
            fi:
         elif op(1,data[m])=B then
            if op(2,data[m])=Up then
               T:=affineM3d(Matrix(Vector([-p,-p,p]),
                  shape=diagonal),Vector([(i+1)*p,(j+1)*p,k*p])):
            elif op(2,data[m])=-Up then
               T:=affineM3d(Matrix(Vector([p,-p,p]),
                  shape=diagonal),Vector([i*p,(j+1)*p,k*p])):
```

```
   elif op(2,data[m])=Dn then
      T:=affineM3d(Matrix(Vector([p,-p,-p]),
         shape=diagonal),Vector([i*p,(j+1)*p,(k+1)*p])):
   elif op(2,data[m])=-Dn then
      T:=affineM3d(Matrix(Vector([-p,-p,-p]),
         shape=diagonal),Vector([(i+1)*p,(j+1)*p,(k+1)*p])):
   elif op(2,data[m])=Rt then
      T:=affineM3d(Matrix([[0,0,-p],[0,-p,0],[-p,0,0]],
         scan=columns),Vector([(i+1)*p,(j+1)*p,(k+1)*p])):
   elif op(2,data[m])=-Rt then
      T:=affineM3d(Matrix([[0,0,p],[0,-p,0],[-p,0,0]],
         scan=columns),Vector([(i+1)*p,(j+1)*p,k*p])):
   elif op(2,data[m])=Lt then
      T:=affineM3d(Matrix([[0,0,p],[0,-p,0],[p,0,0]],
         scan=columns),Vector([i*p,(j+1)*p,k*p])):
   elif op(2,data[m])=-Lt then
      T:=affineM3d(Matrix([[0,0,-p],[0,-p,0],[p,0,0]],
         scan=columns),Vector([i*p,(j+1)*p,(k+1)*p])):
   else T:=NULL;
   fi:
elif op(1,data[m])=L then
   if op(2,data[m])=Up then
      T:=affineM3d(Matrix([[0,p,0],[-p,0,0],[0,0,p]],
         scan=columns),Vector([(i+1)*p,j*p,k*p])):
   elif op(2,data[m])=-Up then
      T:=affineM3d(Matrix([[0,-p,0],[-p,0,0],[0,0,p]],
         scan=columns),Vector([(i+1)*p,(j+1)*p,k*p])):
   elif op(2,data[m])=Dn then
      T:=affineM3d(Matrix([[0,-p,0],[-p,0,0],[0,0,-p]],
         scan=columns),Vector([(i+1)*p,(j+1)*p,(k+1)*p])):
   elif op(2,data[m])=-Dn then
      T:=affineM3d(Matrix([[0,p,0],[-p,0,0],[0,0,-p]],
         scan=columns),Vector([(i+1)*p,j*p,(k+1)*p])):
   elif op(2,data[m])=Rt then
      T:=affineM3d(Matrix([[0,0,-p],[-p,0,0],[0,p,0]],
         scan=columns),Vector([(i+1)*p,j*p,(k+1)*p])):
   elif op(2,data[m])=-Rt then
      T:=affineM3d(Matrix([[0,0,p],[-p,0,0],[0,p,0]],
         scan=columns),Vector([(i+1)*p,j*p,k*p])):
   elif op(2,data[m])=Lt then
      T:=affineM3d(Matrix([[0,0,p],[-p,0,0],[0,-p,0]],
         scan=columns),Vector([(i+1)*p,(j+1)*p,k*p])):
   elif op(2,data[m])=-Lt then
      T:=affineM3d(Matrix([[0,0,-p],[-p,0,0],[0,-p,0]],
         scan=columns),Vector([(i+1)*p,(j+1)*p,(k+1)*p])):
   else T:=NULL;
   fi:
elif op(1,data[m])=R then
```

```
   if op(2,data[m])=Up then
      T:=affineM3d(Matrix([[0,-p,0],[p,0,0],[0,0,p]],
         scan=columns),Vector([i*p,(j+1)*p,k*p])):
   elif op(2,data[m])=-Up then
      T:=affineM3d(Matrix([[0,p,0],[p,0,0],[0,0,p]],
         scan=columns),Vector([i*p,j*p,k*p])):
   elif op(2,data[m])=Dn then
      T:=affineM3d(Matrix([[0,p,0],[p,0,0],[0,0,-p]],
         scan=columns),Vector([i*p,j*p,(k+1)*p])):
   elif op(2,data[m])=-Dn then
      T:=affineM3d(Matrix([[0,-p,0],[p,0,0],[0,0,-p]],
         scan=columns),Vector([i*p,(j+1)*p,(k+1)*p])):
   elif op(2,data[m])=Rt then
      T:=affineM3d(Matrix([[0,0,-p],[p,0,0],[0,-p,0]],
         scan=columns),Vector([i*p,(j+1)*p,(k+1)*p])):
   elif op(2,data[m])=-Rt then
      T:=affineM3d(Matrix([[0,0,p],[p,0,0],[0,-p,0]],
         scan=columns),Vector([i*p,(j+1)*p,k*p])):
   elif op(2,data[m])=Lt then
      T:=affineM3d(Matrix([[0,0,p],[p,0,0],[0,p,0]],
         scan=columns),Vector([i*p,j*p,k*p])):
   elif op(2,data[m])=-Lt then
      T:=affineM3d(Matrix([[0,0,-p],[p,0,0],[0,p,0]],
         scan=columns),Vector([i*p,j*p,(k+1)*p])):
   else T:=NULL;
   fi:
elif op(1,data[m])=H then
   if op(2,data[m])=Up then
      T:=affineM3d(Matrix([[p,0,0],[0,0,-p],[0,p,0]],
         scan=columns),Vector([i*p,j*p,(k+1)*p])):
   elif op(2,data[m])=-Up then
      T:=affineM3d(Matrix([[-p,0,0],[0,0,-p],[0,p,0]],
         scan=columns),Vector([(i+1)*p,j*p,(k+1)*p])):
   elif op(2,data[m])=Dn then
      T:=affineM3d(Matrix([[-p,0,0],[0,0,-p],[0,-p,0]],
         scan=columns),Vector([(i+1)*p,(j+1)*p,(k+1)*p])):
   elif op(2,data[m])=-Dn then
      T:=affineM3d(Matrix([[p,0,0],[0,0,-p],[0,-p,0]],
         scan=columns),Vector([i*p,(j+1)*p,(k+1)*p])):
   elif op(2,data[m])=Rt then
      T:=affineM3d(Matrix([[0,-p,0],[0,0,-p],[p,0,0]],
         scan=columns),Vector([i*p,(j+1)*p,(k+1)*p])):
   elif op(2,data[m])=-Rt then
      T:=affineM3d(Matrix([[0,p,0],[0,0,-p],[p,0,0]],
         scan=columns),Vector([i*p,j*p,(k+1)*p])):
   elif op(2,data[m])=Lt then
      T:=affineM3d(Matrix([[0,p,0],[0,0,-p],[-p,0,0]],
         scan=columns),Vector([(i+1)*p,j*p,(k+1)*p])):
```

```
               elif op(2,data[m])=-Lt then
                  T:=affineM3d(Matrix([[0,-p,0],[0,0,-p],[-p,0,0]],
                     scan=columns),Vector([(i+1)*p,(j+1)*p,(k+1)*p])):
               else T:=NULL;
               fi:
            elif op(1,data[m])=N then
               if op(2,data[m])=Up then
                  T:=affineM3d(Matrix([[p,0,0],[0,0,p],[0,-p,0]],
                     scan=columns),Vector([i*p,(j+1)*p,k*p])):
               elif op(2,data[m])=-Up then
                  T:=affineM3d(Matrix([[-p,0,0],[0,0,p],[0,-p,0]],
                     scan=columns),Vector([(i+1)*p,(j+1)*p,k*p])):
               elif op(2,data[m])=Dn then
                  T:=affineM3d(Matrix([[-p,0,0],[0,0,p],[0,p,0]],
                     scan=columns),Vector([(i+1)*p,j*p,k*p])):
               elif op(2,data[m])=-Dn then
                  T:=affineM3d(Matrix([[p,0,0],[0,0,p],[0,p,0]],
                     scan=columns),Vector([i*p,j*p,k*p])):
               elif op(2,data[m])=Rt then
                  T:=affineM3d(Matrix([[0,p,0],[0,0,p],[p,0,0]],
                     scan=columns),Vector([i*p,j*p,k*p])):
               elif op(2,data[m])=-Rt then
                  T:=affineM3d(Matrix([[0,-p,0],[0,0,p],[p,0,0]],
                     scan=columns),Vector([i*p,(j+1)*p,k*p])):
               elif op(2,data[m])=Lt then
                  T:=affineM3d(Matrix([[0,-p,0],[0,0,p],[-p,0,0]],
                     scan=columns),Vector([(i+1)*p,(j+1)*p,k*p])):
               elif op(2,data[m])=-Lt then
                  T:=affineM3d(Matrix([[0,p,0],[0,0,p],[-p,0,0]],
                     scan=columns),Vector([(i+1)*p,j*p,k*p])):
               else T:=NULL;
               fi:
            else T:=NULL;
            fi:
            Lst:=Lst,T:
            m:=m+1;
         od:
      od:
   od:
   [  Lst];
end:

# Procedures for rendering IFS attractors

# Random method
DrawIFS3d:=proc(IFS,n)
   local i,m,a,b,c,t,ifs,dice,r,pts,FastIFSData,s,x,y,z;
   FastIFSData:=proc(a,b,c,t,r,n,s)
```

```
      local i,j,k,pts;
      pts:=hfarray(1..n,1..3);
      pts[1,1]:=s[0]: pts[1,2]:=s[1]: pts[1,3]:=s[2]:
      for i from 2 to n do
         k:=r[i];   j:=i-1;
         pts[i,1]:=a[0,k]*pts[j,1]+a[1,k]*pts[j,2]+a[2,k]*pts[j,3]+t[0,k];
         pts[i,2]:=b[0,k]*pts[j,1]+b[1,k]*pts[j,2]+b[2,k]*pts[j,3]+t[1,k];
         pts[i,3]:=c[0,k]*pts[j,1]+c[1,k]*pts[j,2]+c[2,k]*pts[j,3]+t[2,k];
      od:
      pts;
   end:
   ifs:=convert(IFS,affine3df);
   pts:=hfarray(1..n,1..3);
   m:=nops(ifs)-1;
   dice:=rand(0..m);
   r:=hfarray(1..n,[seq(dice(),i=1..n)]);
   a:=hfarray(0..2,0..m,[[seq(op(1,ifs[i+1]),i=0..m)],
      [  seq(op(2,ifs[i+1]),i=0..m)],[seq(op(3,ifs[i+1]),i=0..m)]]);
   b:=hfarray(0..2,0..m,[[seq(op(4,ifs[i+1]),i=0..m)],
      [  seq(op(5,ifs[i+1]),i=0..m)],[seq(op(6,ifs[i+1]),i=0..m)]]);
   c:=hfarray(0..2,0..m,[[seq(op(7,ifs[i+1]),i=0..m)],
      [  seq(op(8,ifs[i+1]),i=0..m)],[seq(op(9,ifs[i+1]),i=0..m)]]);
   t:=hfarray(0..2,0..m,[[seq(op(10,ifs[i+1]),i=0..m)],
      [  seq(op(11,ifs[i+1]),i=0..m)],[seq(op(12,ifs[i+1]),i=0..m)]]);
   # compute a fixed point to use as a seed
   s:=hfarray(0..2);
   assign(solve({x=a[0,0]*x+a[1,0]*y+a[2,0]*z+t[0,0],
   y=b[0,0]*x+b[1,0]*y+b[2,0]*z+t[1,0],
   z=c[0,0]*x+c[1,0]*y+c[2,0]*z+t[2,0]})):
   s[0]:=x:   s[1]:=y:   s[2]:=z:
   plots[display3d](PLOT3D(POINTS( evalhf(FastIFSData(a,b,c,t,r,n,s))) ),
      scaling=constrained,axes=none,symbol=POINT,args[3..nargs]);
end:

# DrawDetIFS3d internal use only
   MapOne3d:=proc(aff)
      local x,y,z,a1,a2,a3,b1,b2,b3,c1,c2,c3,t1,t2,t3,A;
      A:=convert(aff,affine3d);
      a1:=op(1,A);   b1:=op(4,A);   c1:=op(7,A);
      a2:=op(2,A);   b2:=op(5,A);    c2:=op(8,A);
      a3:=op(3,A);   b3:=op(6,A);    c3:=op(9,A);
      t1:=op(10,A);   t2:=op(11,A);   t3:=op(12,A);
      unapply([a1*x+a2*y+a3*z+t1,b1*x+b2*y+b3*z+t2,c1*x+c2*y+c3*z+t3],x,y,z);
   end:

   Map3d:=proc(aff)
      if type(aff,list) then map(MapOne3d,aff);
      else MapOne3d(aff);
```

```
      fi:
   end:

   Transform3d:=proc(aff)
      if type(aff,list) then map(x->transform(Map3d(x)),aff);
      else transform(Map3d(aff));
      fi:
   end:

   ApplyIFS3d:=proc(image,IFSTrans)
      local i;
      plots[display3d]({seq(IFSTrans[i](image),i=1..nops(IFSTrans))});
   end:

# Deterministic method
DrawDetIFS3d:=proc(image,IFS,n)
   local P,T,i;
   P:=image;
   T:=Transform3d(IFS);
   for i to n do
      P:=ApplyIFS3d(P,T);
   od:
   plots[display3d](P,args[4..nargs],scaling=CONSTRAINED);
end:

# POV-Ray Interface
# Random method of attractor rendering analogous to DrawIFS3d
WriteIFSPOV:=proc(NewFile,IFS,n)
   local fileId,ifs,m,dice,r,a,b,c,t,pts,di,i,j,k:
   fileId := fopen(NewFile,WRITE,TEXT):
   ifs:=convert(IFS,affine3df);
   m:=nops(ifs)-1;
   dice:=rand(0..m);
   r:=array(1..n,[seq(dice(),i=1..n)]);
   a:=hfarray(0..2,0..m,[[seq(op(1,ifs[i+1]),i=0..m)],
         [  seq(op(2,ifs[i+1]),i=0..m)],[seq(op(3,ifs[i+1]),i=0..m)]]);
   b:=hfarray(0..2,0..m,[[seq(op(4,ifs[i+1]),i=0..m)],
         [  seq(op(5,ifs[i+1]),i=0..m)],[seq(op(6,ifs[i+1]),i=0..m)]]);
   c:=hfarray(0..2,0..m,[[seq(op(7,ifs[i+1]),i=0..m)],
         [  seq(op(8,ifs[i+1]),i=0..m)],[seq(op(9,ifs[i+1]),i=0..m)]]);
   t:=hfarray(0..2,0..m,[[seq(op(10,ifs[i+1]),i=0..m)],
         [  seq(op(11,ifs[i+1]),i=0..m)],[seq(op(12,ifs[i+1]),i=0..m)]]);
   pts:=hfarray(1..n,1..3);
   # compute a fixed point to use as a seed
   assign(solve({x=a[0,0]*x+a[1,0]*y+a[2,0]*z+t[0,0],
            y=b[0,0]*x+b[1,0]*y+b[2,0]*z+t[1,0],
            z=c[0,0]*x+c[1,0]*y+c[2,0]*z+t[2,0]})):
   pts[1,1]:=x:    pts[1,2]:=y:   pts[1,3]:=z:
```

```
   fprintf(fileId,''sphere { <%13.10f,%13.10f,%13.10f> IFSRadius texture{
                  SphereTexIFS } }n'',pts[1,1],pts[1,2],pts[1,3]);
   for k from 2 to n do
      di:=r[k];   j:=k-1;
      pts[k,1]:=a[0,di]*pts[j,1]+a[1,di]*pts[j,2]+a[2,di]*pts[j,3]+t[0,di];
      pts[k,2]:=b[0,di]*pts[j,1]+b[1,di]*pts[j,2]+b[2,di]*pts[j,3]+t[1,di];
      pts[k,3]:=c[0,di]*pts[j,1]+c[1,di]*pts[j,2]+c[2,di]*pts[j,3]+t[2,di];
      fprintf(fileId,''sphere { <%13.10f,%13.10f,%13.10f> IFSRadius texture{
                     SphereTexIFS } }n'',pts[k,1],pts[k,2],pts[k,3]);
   od:
   fclose(fileId);
end:

# famous 3D IFSs

MengerSponge:=CubeIFS([F,Up],[F,Up],[F,Up],[F,Up],
               none,[F,Up],[F,Up],[F,Up],[F,Up],[F,Up],
               none,[F,Up],none,none,none,[F,Up],
               none,[F,Up],[F,Up],[F,Up],[F,Up],[F,Up],
               none,[F,Up],[F,Up],[F,Up],[F,Up]):

SierpinskiTetrahedron:=[affineG3d(.5,.5,.5,0,0,0,0,0,0),
                   affineG3d(.5,.5,.5,0,0,0,0,-.5,0),
                   affineG3d(.5,.5,.5,0,0,0,-0.5,-.25,0),
                   affineG3d(.5,.5,.5,0,0,0,-.25,-.25,.5)];

# Common starting figures for Deterministic IFS -- DrawDetIFS3d

MrPoint3d:=pointplot3d([0,0,0]):

MrCorner:=display3d(
         {line([0,0,0],[1,0,0]),
         line([0,0,0],[0,1,0]),
         line([0,0,0],[0,0,1])},
         scaling=CONSTRAINED):

MrPyramid:=display3d(
         plots[polygonplot3d]([
            [[.5,-evalf(sqrt(3)/6),0],[0,0,evalf(sqrt(2/3))],
             [0,evalf(sqrt(3)/3),0]],
            [[.5,-evalf(sqrt(3)/6),0],[0,0,evalf(sqrt(2/3))],
             [-.5,-evalf(sqrt(3)/6),0]],
            [[-.5,-evalf(sqrt(3)/6),0],[0,0,evalf(sqrt(2/3))],
             [0,evalf(sqrt(3)/3),0]],
            [[.5,-evalf(sqrt(3)/6),0],[0,evalf(sqrt(3)/3),0],
             [-.5,-evalf(sqrt(3)/6),0]]]),
         scaling=CONSTRAINED,
         style=PATCHNOGRID):
```

```
MrBlock:=display3d(
         cuboid([0,0,0],[1,1,1]),
         scaling=CONSTRAINED,
         style=PATCHNOGRID):

MrBlockHead:=display3d(
         {cuboid([0,0,0],[1,1,1]),
         spacecurve({[.3+0.075*cos(t),0,.7+0.075*sin(t),t=0..2*Pi],
            [t,0,.7,t=0.6..0.8],[.5+.25*cos(t),0,.5+.25*sin(t),t=-3*Pi/4..-Pi/4]}
            ,color=black),
         spacecurve({[.5+.4*cos(t),.5+.4*sin(t),1,t=0..2*Pi],
            [.5+.3*cos(t),.5+.3*sin(t),1,t=0..2*Pi]}
            ,color=black),
         spacecurve({[1,.25+.25*cos(t),.5+.25*sin(t),t=-Pi/2..Pi/2],
            [1,(.25+.25*cos(Pi/3))+.07*cos(t),(.5+.25*sin(-Pi/3))+.07*sin(t),
               t=-5*Pi/4..0]}
            ,color=black),
         spacecurve({[0,.25+.25*sin(t),.5+.25*cos(t),t=0..Pi]}
            ,color=black),
         spacecurve({[t,.5*t-.15,1,t=.4..0.6],[t,-.5*t+.45,1,t=.4..0.6],
            [.4,s,1,s=.05..0.25]}
            ,color=black),
         spacecurve({[(t+.54)/1.6,1,t,t=.41..0.9],[(t+.3)/1.6,1,t,t=.1..0.3],
            [(t-1.06)/(-1.6),1,t,t=.5..0.9],[(t-1.3)/(-1.6),1,t,t=.1..0.3],
            [(t+1.7)/4,1,t,t=.44..0.9],[(t+1.5)/4,1,t,t=.1..0.3],
            [(t-2.3)/(-4),1,t,t=.47..0.9],[(t-2.5)/(-4),1,t,t=.1..0.3],
            [t,1,-.4*t+.6,t=.3..0.7],
            [s,1,.32,s=.3..0.7],
            [.3,1,s,s=.32..0.48]}
            ,color=black),
         spacecurve({[t,.5*t-.05,0,t=.3..0.7],[t,(-.5)*t+.45,0,t=.3..0.7],
            [t,5*t-2.3,0,t=.5..0.6],[t,(-5)*t+2.7,0,t=.4..0.5],
            [t,t+.3,0,t=.4..0.6],[t,1.5-t,0,t=.6..0.8],
            [.3,t,0,t=.1..0.3],[.7,t,0,t=.1..0.3],[t,.7,0,t=.4..0.8]}
            ,color=black)
         },
         scaling=CONSTRAINED,
         style=PATCHNOGRID):

MrBlockFrame:=display3d(
         {line([0,0,0],[0,0,1]), line([0,0,1],[1,0,1]), line([1,0,0],[1,0,1]),
         line([0,0,0],[1,0,0]), line([0,0,0],[0,1,0]), line([0,1,0],[0,1,1]),
         line([0,1,1],[1,1,1]), line([0,1,0],[1,1,0]), line([1,0,0],[1,1,0]),
         line([1,1,0],[1,1,1]), line([1,0,1],[1,1,1]), line([0,0,1],[0,1,1]),
         spacecurve({[.3+0.075*cos(t),0,.7+0.075*sin(t),t=0..2*Pi],
            [t,0,.7,t=0.6..0.8],[.5+.25*cos(t),0,.5+.25*sin(t),t=-3*Pi/4..-Pi/4]}
            ,color=red),
```

```
        spacecurve({[.5+.4*cos(t),.5+.4*sin(t),1,t=0..2*Pi],
           [.5+.3*cos(t),.5+.3*sin(t),1,t=0..2*Pi]}
           ,color=red),
        spacecurve({[1,.25+.25*cos(t),.5+.25*sin(t),t=-Pi/2..Pi/2],
           [1,(.25+.25*cos(Pi/3))+.07*cos(t),(.5+.25*sin(-Pi/3))+.07*sin(t),
              t=-5*Pi/4..0]}
           ,color=red),
        spacecurve({[0,.25+.25*sin(t),.5+.25*cos(t),t=0..Pi]}
           ,color=red),
        spacecurve({[t,.5*t-.15,1,t=.4..0.6],[t,-.5*t+.45,1,t=.4..0.6],
           [.4,s,1,s=.05..0.25]}
           ,color=red),
        spacecurve({[(t+.54)/1.6,1,t,t=.41..0.9],[(t+.3)/1.6,1,t,t=.1..0.3],
           [(t-1.06)/(-1.6),1,t,t=.5..0.9],[(t-1.3)/(-1.6),1,t,t=.1..0.3],
           [(t+1.7)/4,1,t,t=.44..0.9],[(t+1.5)/4,1,t,t=.1..0.3],
           [(t-2.3)/(-4),1,t,t=.47..0.9],[(t-2.5)/(-4),1,t,t=.1..0.3],
           [t,1,-.4*t+.6,t=.3..0.7],
           [s,1,.32,s=.3..0.7],
           [.3,1,s,s=.32..0.48]}
           ,color=red),
        spacecurve({[t,.5*t-.05,0,t=.3..0.7],[t,(-.5)*t+.45,0,t=.3..0.7],
           [t,5*t-2.3,0,t=.5..0.6],[t,(-5)*t+2.7,0,t=.4..0.5],
           [t,t+.3,0,t=.4..0.6],[t,1.5-t,0,t=.6..0.8],
           [.3,t,0,t=.1..0.3],[.7,t,0,t=.1..0.3],[t,.7,0,t=.4..0.8]}
           ,color=red)
        },
        scaling=CONSTRAINED,
        style=PATCHNOGRID):

end module:
```

8. Appendix D

8.1. **Introduction to the Chaos3d module. Calling Sequence:**

readlib(chaos3d)
function (args)

Description:

– To use a chaos3d function, you must first execute the readlib(chaos3d) command.

The functions available are:

IFS's and AFFINE maps

In the following routines, we define two data types: AFFINE and IFS.

– An IFS is a single AFFINE or a list of one or more AFFINEs.

– An AFFINE represents an affine map of 3-dimensional space and comes in five forms:

affine3d($a_1, a_2, a_3, b_1, b_2, b_3, c_1, c_2, c_3, t_1, t_2, t_3$) - standard 3d form

affineS3d $\left(r, s, t, \theta_i, \phi_i, \theta_j, \phi_j, \theta_k, \phi_k, l, m, n\right)$ - spherical 3d form

affineG3d($r, s, t, \theta, \phi, \gamma, l, m, n$) - geometric 3d form

affineV3d($\vec{v}_i, \vec{v}_j, \vec{v}_k, \vec{t}$) - vector 3d form (where each of $\vec{v}_i, \vec{v}_j, \vec{v}_k, \vec{t}$ are *Vectors* or three-element *lists*)

affineM3d(M, B) - matrix 3d form

These are inert forms representing the maps defined in Appendix B.

Built-in IFS's:

The following IFS data types are built-in:

MengerSponge SierpinskiTetrahedron

Built-in starting figures:

The following plot structures are built-in, and can be used as the starting figure for plotting the iterations of the deterministic method for an IFS (see *DrawDetIFS3d()* below).

MrPoint3d MrBlockHead MrBlockFrame MrBlock MrCorner MrPyramid

IFS routines:

'convert/affineV3d'(A::AFFINE) - converts an AFFINE or IFS object to its affineV3d() form

'convert/affineS3d'(A::AFFINE) - converts an AFFINE or IFS object to its affineS3d() form

'convert/affineM3d'(A::AFFINE) - converts an AFFINE or IFS object to its affineM3d() form

'convert/affine3d' (A::AFFINE) - converts an AFFINE or IFS object to its affine3d() form

'convert/affine3df'(A::AFFINE) - converts an AFFINE or IFS object to its affine3d() form

having all floating point arguments

affineV3dFromPoints($p_1, p_2, p_3, p_4, T_{p1}, T_{p2}, T_{p3}, T_{p4}$)

– Computes the AFFINE object affineV3d($\vec{v}_i, \vec{v}_j, \vec{v}_k, \vec{t}$) for the unique affine map $T(\vec{u}) = M(\vec{v}_i, \vec{v}_j, \vec{v}_k)(\vec{u}) + \vec{t}$ that maps points p_1 to T_{p1}, p_2 to T_{p2}, p_3 to T_{p3}, and p_4 to T_{p4}. These arguments are each lists containing three elements. A valid procedure call contains 8 lists of the form $[x, y, z]$.

IFSFromList3d(List)

-creates an IFS data structure from a list of m lists of the form $\left[\vec{v}_i, \vec{v}_j, \vec{v}_k, \vec{t}\right]$ where each of $\vec{v}_i, \vec{v}_j, \vec{v}_k, \vec{t}$ is a *Vector* or a *list* of the form $[a, b, c]$ as would be found in an affineV3d call.

Map3d(IFS)

- converts the AFFINE or IFS object to the map $T(x, y, z) = (a_1 * x + a_2 * y + a_3 * z + t_1,$ $b_1 * x + b_2 * y + b_3 * z + t_2,$ $c_1 * x + c_2 * y + c_3 * z + t_3)$.

Transform3d(IFS)

- converts the AFFINE or IFS object to a transform which can be applied to plot objects.

DrawDetIFS3d(figure,IFS,n)

- plots the n^{th} iteration of the attractor of *IFS* using the deterministic method starting with *figure* as the seed. *figure* can be a predefined starting figure (see above) or another well-defined plot structure.

DrawIFS3d(IFS,n)

- plots the attractor of *IFS* by the random iteration method using n points. (Note: An image rendered by maple using this command will be extremely fuzzy unless MANY more points are rendered than you have time to wait for. Use of a value for n larger than about 3000 is not recommended. In general, DrawDetIFS3d() using MrPoint3d as figure will render a more accurate image more quickly.)

ContractionFactor3d(Aff)

- where *Aff* is an AFFINE object or a list of such objects. Returns the scaling/contraction factor(s) for the affine transformation(s) defined by the AFFINE object.

WriteIFSPOV(NewFile,IFS,n)

- creates a POV-Ray include file at the location specified by *NewFile* containing a list of n POV-Ray spheres each centered at a point in the attractor of *IFS*. *NewFile* is defined in quotes in this form: "c:\\documents\\NewFileName.inc". The .inc file includes a list of these spheres with placeholders for user-defined radius, texture, etc.

In addition, classic IFS's can be quickly produced with the following routine:

CubeIFS(seq)

- This routine takes n^3 arguments. Each argument defines one affine map in an IFS based on the following structure: Begin with the unit cube $[0..1] \times [0..1] \times [0..1]$ situated in x, y, z space such that the front lower left corner is located at the origin, the rear lower left corner is at $(0, 1, 0)$, and the front upper left corner is located at $(0, 0, 1)$. Thereby, the cube is resting with its base in quadrant I of the x, y plane and its body entirely in the $+z$ half plane. This is *standard position*. Divide the unit cube into n by n smaller cubes all contained within the area of the original unit cube. Number the smaller cubes from 1 to n^3 starting in the front lower left corner and moving from $x = 0$ to $x = 1$ (left to right) and then from $z = 0$ to $z = 1$ (bottom to top) and then from $y = 0$ to $y = 1$ (front to back). For example: If the unit cube is divided into 2×2 sub-cubes, view the structure by looking in the positive y-direction with the xz plane in front of you and the $+x$ axis pointing to your right. Then cube #1 has its front lower left corner at the origin, cube #4 has its front upper right corner at $(1, 0, 1)$, and cube #6 has its rear lower right corner at $(1, 1, 0)$. The i^{th} argument of the CubeIFS procedure call describes an affine transformation which maps the unit cube into the ith smaller cube in a manner corresponding one of the 48 symmetry operations on the unit cube as defined in Appendix A.

- There are 2 acceptable forms for entering arguments (affine maps) into this IFS structure; *symmetry notation form* and *MrBlockHead form*. Differing notational forms may not be mixed in one CubeIFS call.

Symmetry Notation Form

Acceptable compositions of the generating elements for the group of symmetry operations on the cube are listed in the table in Appendix A. The terms along the top row and left column correspond to *MrBlockHead form* and may not be combined with symmetry notation in the same CubeIFS call.

When the i^{th} argument of a CubeIFS call contains *none* (or any unrecognized symbol), no affine map is assigned to that cube cell. If the i^{th} argument contains a symmetry operation from the table above, then the corresponding affine map performs the action of first positioning the cube according to the symmetry operation stated, then contracting the cube from its usual $1 \times 1 \times 1$ size and finally relocating it to the i^{th} cell position.

MrBlockHead Form

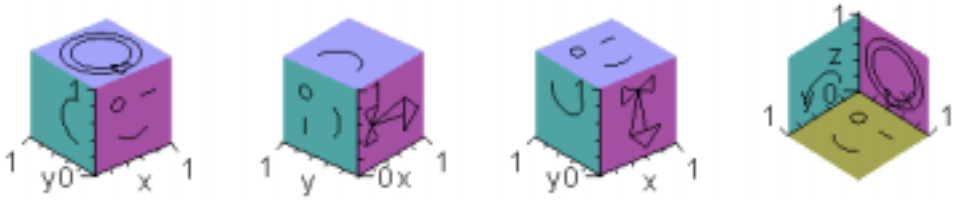
Each argument in MrBlockHead form describes the appearance of the pre-defined starting figure, MrBlockHead, in one of his 48 possible orientations corresponding to the symmetries of the cube. An argument comes from the set

$$\{[P,Q], none : P \in \{F,B,R,L,H,N\} \text{ and } Q \in \{Lt, Rt, Up, Dn, -Lt, -Rt, -Up, -Dn\}\}.$$

A precise definition of these arguments is given by the symmetry group table in Appendix A. The P value is along the top row and the Q value descends the left column. The ordered pair $[P,Q]$ represents the affine map corresponding to one of the 48 elements in the symmetry group as defined by the said composition of the generating elements. When the i^{th} argument contains *none*, no affine map is assigned to that cube cell.

The visual interpretation of $[P,Q]$ notation is as follows. If the i^{th} argument contains an ordered pair $[P,Q]$, then the corresponding affine map performs the action of shrinking MrBlockHead from his usual $1 \times 1 \times 1$ size and upright position into the i^{th} cell such that when viewed from standard position his face is now on the side of his head normally occupied by P and tipped in the direction defined by Q. The abbreviations in the set P stand for **F**ace, **B**ack, his **R**ight ear, his **L**eft ear, **H**at, **N**ecktie. The abbreviations in the set Q correspond to the direction his face is tipped (assuming you are now viewing him head-on) and mean: Up, eyes near top with open eye on your left; Dn, eyes near bottom with open eye on your right; Rt, eyes near your right with open eye on top; Lt, eyes near your left with open eye on bottom; $-Up$, eyes near top with open eye on your right; $-Dn$, eyes near bottom with open eye on your left; $-Rt$, eyes near your right with open eye on bottom; $-Lt$, eyes near your left with open eye on top.

Example: The pair $[R,-Lt]$ is equivalent to the symmetry group element rt and it means that MrBlockHead's face is now on the side of his head normally occupied by HIS Right ear, and if we now look at that side of his head, we notice that his eyes are toward OUR left with the open eye on top. (Note: when using this notation it is important to understand the correct position for $[H,Up]$ and $[N,Up]$ especially.)



$[F,Up]$ $[R,-Lt]$ $[H,Up]$ $[N,Up]$

Symmetry Notation form and *MrBlockHead form* produce identical IFS's.

For example, one IFS that produces the Menger Sponge can be entered:

```
CubeIFS([F,Up],[F,Up],[F,Up],[F,Up],none,[F,Up],[F,Up],[F,Up],[F,Up],
        [F,Up],none,[F,Up],none,none,none,[F,Up],none,[F,Up],
        [F,Up],[F,Up],[F,Up],[F,Up],none,[F,Up],[F,Up],[F,Up],[F,Up]);
```

or using the symmetry notation:

```
CubeIFS(e,e,e,e,none,e,e,e,e,
        e,none,e,none,none,none,e,none,e,
        e,e,e,e,none,e,e,e,e);
```

However, it is INCORRECT to enter:

```
CubeIFS([F,Up],[F,Up],[F,Up],[F,Up],none,[F,Up],[F,Up],[F,Up],[F,Up],
         e,none,e,none,none,none,e,none,e,
         e,e,[F,Up],e,none,e,e,[F,Up],e);
```

9. Appendix E

9.1. **Known Attractors of $W \in \Psi$ having $|S(A)| = 6$.** Let $W \in \Psi$ be given by $W(A) = v_1 d_1(A) \cup v_2 d_2(A) \cup v_3 d_3(A) \cup v_4 d_4(A)$. Then $|\operatorname{Sym}(A)| = 6$ when d_i is chosen according to one of the eight possibilities below. Each attractor can be rendered by 6^4 distinct IFS's as determined by Theorem 3.14. The notation below lists all possible values for d_i in order to produce the corresponding attractor. These values are elements of the group of symmetry operations on the cube as defined in Appendix A.

Sierpinski Right Tetrahedron

$$\forall i \in 1..4,\ d_i \in \left\{e, rt, r^3s^3, -r^3, -r^2s^3, -t\right\}$$

The Jax

$$\forall i \in 1..4,\ d_i \in \left\{-s^2, -rt^3, -rs, rs^2, r^2t^3, r^2s\right\}$$

Triangular Plane (of area $\sqrt{3}$)

$$\begin{aligned} d_1 &\in \left\{-s^2, -rt^3, -rs, rs^2, r^2t^3, r^2s\right\} \\ \forall i &\in 2..4,\ d_i \in \left\{e, rt, r^3s^3, -r^3, -r^2s^3, -t\right\} \end{aligned}$$

The Bloom

$$\begin{aligned} d_1 &\in \left\{e, rt, r^3s^3, -r^3, -r^2s^3, -t\right\} \\ \forall i &\in 2..4,\ d_i \in \left\{-s^2, -rt^3, -rs, rs^2, r^2t^3, r^2s\right\} \end{aligned}$$

Deadly Butterfly

$$\begin{aligned} d_1 &\in \left\{e, rt, r^3s^3, -r^3, -r^2s^3, -t\right\} \\ d_2 &\in \left\{r^3t, -r^2t, s^2, -r^2s, r^3s, -r^3s^2\right\} \\ d_3 &\in \alpha_4 d_2 = \left\{-t^3, rt^3, rs^3, r^2s^2, -rs^2, -s^3\right\} \\ d_4 &\in \alpha_4 d_3 = \left\{r^2, -r, rs, -r^2t^3, r^3t^3, -s\right\} \end{aligned}$$

Galactica Rocket Ship

$$\begin{aligned} d_1 &\in \left\{-s^2, -rt^3, -rs, rs^2, r^2t^3, r^2s\right\} \\ d_2 &\in \left\{r^3t, -r^2t, s^2, -r^2s, r^3s, -r^3s^2\right\} \\ d_3 &\in \alpha_4 d_2 = \left\{-t^3, rt^3, rs^3, r^2s^2, -rs^2, -s^3\right\} \\ d_4 &\in \alpha_4 d_3 = \left\{r^2, -r, rs, -r^2t^3, r^3t^3, -s\right\} \end{aligned}$$

"Mr. T"

$$\begin{aligned} d_1 &\in \left\{e, rt, r^3s^3, -r^3, -r^2s^3, -t\right\} \\ d_2 &\in \left\{r, r^2s^3, t^3, -e, -rs^3, -r^3t^3\right\} \\ d_3 &\in \alpha_4 d_2 = \left\{r^2t, s, -r^3s, -r^2, r^3, -rt\right\} \\ d_4 &\in \alpha_4 d_3 = \left\{s^3, r^3s^2, -r^2s^2, -r^3t, t, -r^3s^3\right\} \end{aligned}$$

Klingon Battle Cruiser

$$\begin{aligned} d_1 &\in \left\{-s^2, -rt^3, -rs, rs^2, r^2t^3, r^2s\right\} \\ d_2 &\in \left\{r, r^2s^3, t^3, -e, -rs^3, -r^3t^3\right\} \\ d_3 &\in \alpha_4 d_2 = \left\{r^2t, s, -r^3s, -r^2, r^3, -rt\right\} \\ d_4 &\in \alpha_4 d_3 = \left\{s^3, r^3s^2, -r^2s^2, -r^3t, t, -r^3s^3\right\} \end{aligned}$$

10. Appendix F

10.1. **Extension to Maple's *group* package; for using notation for the elements in the symmetry group of the cube as defined in Appendix A.** The following collection of procedures was written by Ken Monks.

```
with(group):

# The generators of the Cube Group

M:=[[1,2],[3,4],[5,6],[7,8]]:
R:=[[1,2,4,3],[5,6,8,7]]:
S:=[[1,3,7,5],[2,4,8,6]]:
T:=[[1,5,6,2],[3,7,8,4]]:
E:=[]:

# This routine allows mulperms to work with a list of more than two
# elements at time composing left to right.

Mult:=proc()
   local a,p:
   a:=[];
   for p in args do a:=mulperms(a,p) od:
   a;
end:

# lookup table to name the elements according to the notation in Appendix A.

Names:=[
   [E,e],[M,-e],
   [R,r],[Mult(M,R),-r],
   [Mult(R,R),r2],[Mult(M,R,R),-r2],
   [Mult(R,R,R),r3],[Mult(M,R,R,R),-r3],
   [S,s],[Mult(M,S),-s],
   [Mult(R,S),rs],[Mult(M,R,S),-rs],
   [Mult(R,R,S),r2s],[Mult(M,R,R,S),-r2s],
   [Mult(R,R,R,S),r3s],[Mult(M,R,R,R,S),-r3s],
   [Mult(S,S),s2],[Mult(M,S,S),-s2],
   [Mult(R,S,S),rs2],[Mult(M,R,S,S),-rs2],
   [Mult(R,R,S,S),r2s2],[Mult(M,R,R,S,S),-r2s2],
   [Mult(R,R,R,S,S),r3s2],[Mult(M,R,R,R,S,S),-r3s2],
   [Mult(S,S,S),s3],[Mult(M,S,S,S),-s3],
   [Mult(R,S,S,S),rs3],[Mult(M,R,S,S,S),-rs3],
   [Mult(R,R,S,S,S),r2s3],[Mult(M,R,R,S,S,S),-r2s3],
   [Mult(R,R,R,S,S,S),r3s3],[Mult(M,R,R,R,S,S,S),-r3s3],
   [T,t],[Mult(M,T),-t],
   [Mult(R,T),rt],[Mult(M,R,T),-rt],
   [Mult(R,R,T),r2t],[Mult(M,R,R,T),-r2t],
   [Mult(R,R,R,T),r3t],[Mult(M,R,R,R,T),-r3t],
```

```
   [Mult(T,T,T),t3],[Mult(M,T,T,T),-t3],
   [Mult(R,T,T,T),rt3],[Mult(M,R,T,T,T),-rt3],
   [Mult(R,R,T,T,T),r2t3],[Mult(M,R,R,T,T,T),-r2t3],
   [Mult(R,R,R,T,T,T),r3t3],[Mult(M,R,R,R,T,T,T),-r3t3],
   [''Error'',ERROR] ]:

# To use a permutation as a function

Apply:=proc(p,n)
   local a,i,j:
   a:=n;
   for i from 1 to nops(p) do
      if has(p[i],a) then
         for j to nops(p[i]) while p[i][j]<>a do
         od:
         if j=nops(p[i]) then
            a:=p[i][1]
         else a:=p[i][j+1]
         fi;
      fi:
   od:
a
end:

# to determine if two permutations are the same element of the group

Equal:=proc(n,p,q)
   local i;
   for i to n while Apply(p,i)=Apply(q,i) do od:
   if i>n then true else false fi:
end:

# to find the name of a permutation in the group

PermToName:=proc(a)
   local i; global Names:
   for i in Names while not Equal(8,i[1],a) do od;
   i[2];
end:

# to find the permutation associated with a name in the group

NameToPerm:=proc(n)
   global Names:
   select(has,Names,n)[1][1];
end:

# To multiply the elements using the names defined above instead
```

```
# of permutation notation

Times:=proc(a,b)
   PermToName(mulperms(NameToPerm(a),NameToPerm(b)))
end:
```

References

[1] Monks, K. *Chaos: A Maple package for Chaos and Fractals.* version 3.16. 2000. on-line, www.scranton.edu/~monks
[2] Lay, David C. *Linear Algebra and Its Applications.* New York: Addison-Wesley Publishing Company, 1994.
[3] Barnsley, M. *Fractals Everywhere.* San Diego: Academic Press, 1988.
[4] Peitgen, Heinz-Otto, Hartmut Jürgens, and Dietmar Saupe. *Chaos and Fractals: New Frontiers of Science* New York: Springer-Verlag, 1992.

152 E Elm St. Apt 4, Dunmore, PA 18512
E-mail address: riggic2@scranton.edu
URL: http://facweb.uofs.edu/~monks/fsrp.html