# Dynamics in the Plane: Fractals and IFS

D. R. Hundley

October 24, 2001

## 1 Defining Functions in the Plane

1. Definition: Let $\mathbf{x} = (x, y)$ be a point in the plane. An *affine map* $f : R^2 \to R^2$ is a function of the form:

$$f(\mathbf{x}) = A\mathbf{x} + \mathbf{b} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix} = \begin{bmatrix} ax + by + e \\ cx + dy + f \end{bmatrix}$$

   so that each affine function is defined by specifying the 6 values $a, b, c, d, e, f$. $A$ is called a $2 \times 2$ matrix, $\mathbf{x}$ and $\mathbf{b}$ are called vectors.

2. Selected actions of affine maps[1]

   (a) The value $e$ shifts things horizontally by $e$, $f$ shifts things vertically.

   (b) Horizontal/vertical contractions/expansions and flips:

$$\begin{bmatrix} a & 0 \\ 0 & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax \\ dy \end{bmatrix}$$

   - If $a > 1$, we have a horizontal expansion, for $0 < a < 1$, we have a horizontal contraction. If $a < -1$, we have an expansion with a flip across the $y$ axis, and if $-1 < a < 0$, there is a horizontal contraction and a flip.
   - Similarly for $d$, except the flip is across the $x$ axis.

   (c) A reflection across $y = x$:

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} y \\ x \end{bmatrix}$$

   (d) Rotation counterclockwise through angle $\theta$:

$$\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x\cos(\theta) - y\sin(\theta) \\ x\sin(\theta) + y\cos(\theta) \end{bmatrix}$$

[1] There are other actions, such as a shear, but we'll only be using the ones listed.

3. Putting the actions together: "Function Composition = Matrix Multiplication"

   To compose two actions, we have to define matrix multiplication: Let

   $$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, \quad B = \begin{bmatrix} e & f \\ g & h \end{bmatrix}$$

   then:

   $$AB = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} ae+bg & af+bh \\ ce+dg & cf+dh \end{bmatrix}$$

   **EXAMPLE:** Let $f$ rotate points by $\frac{\pi}{4}$, and let $g$ expand horizontally by $\sqrt{2}$, and contract vertically by $\frac{1}{3}$. Then:

   $$f(g(\mathbf{x})) = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \sqrt{2} & 0 \\ 0 & \frac{1}{3} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x - \frac{1}{3\sqrt{2}}y \\ x + \frac{1}{3\sqrt{2}}y \end{bmatrix}$$

   which corresponds to expansion/contraction first, then the rotation. The opposite order (rotation first, then expansion/contraction) gives a different action:

   $$g(f(\mathbf{x})) = \begin{bmatrix} \sqrt{2} & 0 \\ 0 & \frac{1}{3} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x - y \\ \frac{1}{3\sqrt{2}}(x+y) \end{bmatrix}$$

   **EXAMPLE:** Matlab has matrix multiplication built-in. If I want a matrix that does the action of $F = f \circ g$, I could type the following set of commands. The first 4 lines constructs the matrix, the next two lines applies the matrix to a random point in the plane, $x$, to get the result, $y$.

   ```
   a=cos(pi/4); b=sin(pi/4);
   A=[a, -b; b, a];
   B=[sqrt(2), 0; 0, 1/3];
   F=A*B;
   x=rand(2,1);
   y=F*x;
   ```

   The difference in Matlab between `A*B` and `A.*B` is that in the first case, we have standard matrix multiplication, and in the second case, we have (the non-standard) element-by-element multiplication.

# 2 Fractals

1. Definition 1: An object $A$ is said to be *self-similar* if its picture is a finite union of smaller copies of itself.

2. By "small copy", we mean that the set of points defining $A$ have been sent through a function that contracts space. In this case, a small copy is written as

$$f(A) = \{f(x) \mid x \in A\}$$

or, $f(A)$ is the image of $A$ under $f$.

3. Definition 2: An object $A$ is said to be *self-similar* if

$$A = \bigcup_{j=1}^{k} f_k(A)$$

where we will assume that each $f_i$ is an affine contraction map.

4. From Definition 2, a self-similar object $A$ is then defined by $k$ sets of 6 constants. It is convenient to write them in tabular form. For example, the following set of functions:

$$f_1(\mathbf{x}) = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}, \quad f_2(\mathbf{x}) = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0.5 \\ 0 \end{bmatrix},$$

$$f_3(\mathbf{x}) = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 0.5 \end{bmatrix},$$

can be written more compactly as:

| | $a$ | $b$ | $c$ | $d$ | $e$ | $f$ |
|---|---|---|---|---|---|---|
| 1 | 0.5 | 0 | 0 | 0.5 | 0 | 0 |
| 2 | 0.5 | 0 | 0 | 0.5 | 0.5 | 0 |
| 3 | 0.5 | 0 | 0 | 0.5 | 0 | 0.5 |

The set of functions above comes from an object known as "The Serpinski Gasket", which is shown in Figure 4 below. Verify that the gasket is made up of these three smaller copies of itself.

Another way of constructing the gasket is the following: Begin with a square. Cut the square into 4 equal squares, and remove the upper right square. With each remaining square, cut each of those into 4 equal squares, and remove the upper right from each. For each of these, cut into 4, and remove the upper right square... Continue ad infinitum.

   (a) How much area has been removed all together from this last construction?

   At the first step, we remove one square, its area is (1/2)(1/2)=1/4 the total.

   At the second step, we remove 3 squares, each area is (1/4)(1/4)=1/16 the total.

   At the third step, we remove ? squares, each area is ?

   Finally, form an infinite sum of all that has been removed...

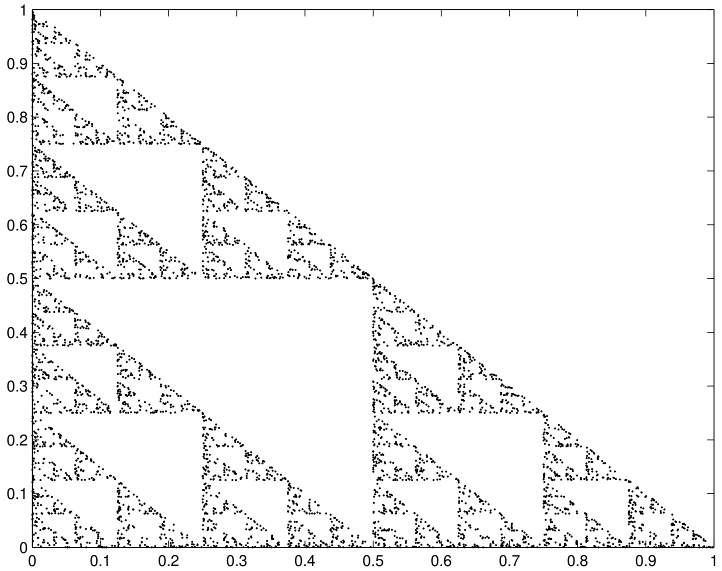


Figure 1: The Serpinski Gasket

(b) What is the "length" of all the line segments in the gasket?

We have the two legs and hypotenuse of the original triangle.

We have one main triangle inside that.

We have 3 triangles that are next biggest.

We have 9 triangles that are next biggest.

We have 27 triangles that are next biggest....

(c) (Remark) The previous two questions show that the gasket has zero area (a 2-dimensional measure), and infinite length boundary (a one-dimensional measure).

(d) In Figures 4g, 4g, 4g, we show the sequence of images that we get by applying functions $f_1, f_2, f_3$ to the unit square. We interpret the sequence of numbers shown as which functions we applied to get that square. For example, the square labelled 221 was the result of the composition

$$f_2(f_2(f_1(A)))$$

Some of the squares in Figure 4g are unlabelled- Place the appropriate label on them.

(e) On Figure 4g, also show where the following three squares would be:

- 1231
- 3333
- 2323

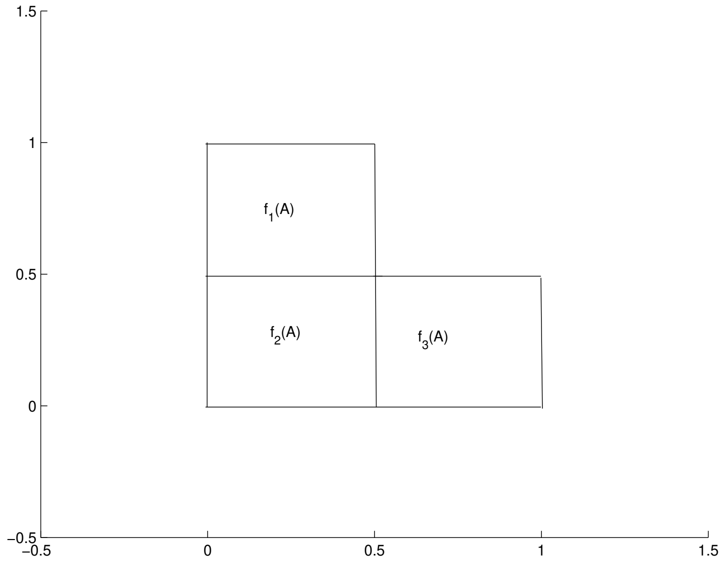


Figure 2: The three images of a unit square under $f_1, f_2, f_3$ for the Serpinski Gasket.

(f) Where are the points ...11111, ...22222, ...33333?

(g) If we were to associate each point of the gasket with an infinite string of 1's, 2's, and 3's, would each point have a unique representation?

Hint: Where would the points ...33331 and ...11113 be?

(h) We can build a 1-1 correspondence between the points of the gasket and $\Sigma_3$, the space of symbols taken from $0, 1, 2$. On this set, the shift map $\sigma$ as defined previously, gives a chaotic dynamical system.

We also have a 1-1 correspondence between the points of the gasket and the interval $[0, 1]$. How would we do that?

(i) (Remark) We can think of the gasket as a visualization of $\Sigma_3$, where $\sigma$ bounces around the points.

(j) We can build a single orbit of $\sigma$ that is dense on the gasket. We do this by taking *any* initial point, $\mathbf{x}_0$, and begin randomly applying one of the three functions to $\mathbf{x}_0$.

Give an argument showing this orbit is dense. That is, given $s = s_0 s_1 s_2 s_3 \ldots \in \Sigma_3$, show that at some point, the random orbit of $x_0$ must get arbitrarily close to $s$.

5. The last question gives us a way of numerically constructing the gasket:

**The Random Iteration Algorithm.**

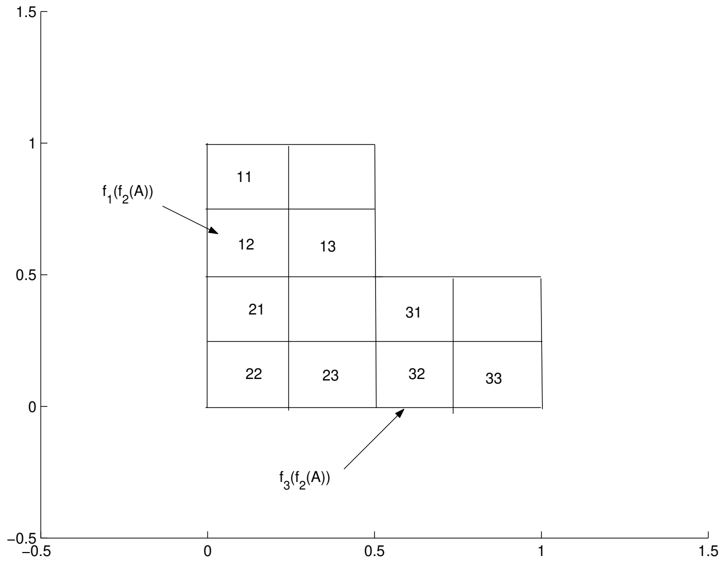


Figure 3: The nine images of a unit square under all possible combinations of two function compositions, taken from $f_1, f_2, f_3$ for the Serpinski Gasket.

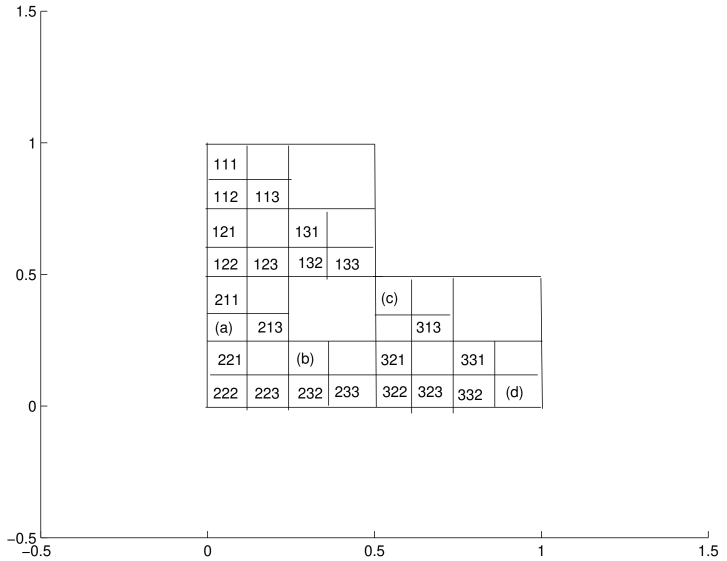


Figure 4: The 27 images of a unit square under all possible combinations of three function compositions, taken from $f_1, f_2, f_3$ for the Serpinski Gasket.

Given a set of contraction maps, $\{f_1, f_2, \ldots, f_k\}$, and initial value $\mathbf{x}$,

- Choose $j$ at random from $\{1, 2, \ldots, k\}$
- Apply function $j$ to $\mathbf{x}$ to get $\mathbf{y}$
- Plot $\mathbf{y}$
- Reset $\mathbf{x} = \mathbf{y}$
- Repeat from Step 1.

Matlab code that implements the algorithm for the gasket is given at the appendix.

6. The set of points in the fractal $A$ are invariant under both forward and backward iteration... How would we define the backwards iteration for the gasket? We need to be careful of the domains of each function...

7. The gasket is *attracting* under forward iteration, and *repelling* under backwards iteration. Another cool algorithm for the coloring the plane:

**Escape Time Algorithm**

Given: (1) $\left\{f_1^{-1}, \ldots, f_k^{-1}\right\}$, the inverse maps and domains from forwards iteration and (2) a set of points in the plane to serve as initial conditions.

Output: For each initial condition, color in how fast the point "escapes" the set. Numerically, we can say that a point has escaped if its size is bigger than some value (say, bigger than 2).

WARNING: With a lot of data, Matlab can take a very, very long time to compute this. I've attached a sample code in C that doesn't take very long at all to compute it. It outputs a data file that Matlab can read in and plot. If the C code outputs the file sample1.dat, in Matlab, we would type:

```
X=load('sample1.dat');

imagesc(X)
```

8. **Exercise:** For each of the four pictures of fractals in our text (Figure 14.19), construct the IFS for each, and verify your functions by constructing the associated attracting set using the Random Iteration Algorithm. You may take the attached m-file and modify it appropriately.

9. **Exercise:** We can construct $\Lambda$, the invariant set for $x^2 + c$ for $c < -2$ in the same way. $\Lambda$ was repelling under forward iteration... What are the two functions that will do backwards iteration? If we construct them correctly, $\Lambda$ becomes an attracting set! Use the Random Iteration Algorithm on these to visualize $\Lambda$.

10. **Barnsley's Fern** Consider the following IFS with probabilities:

| $f$ | $a$ | $b$ | $c$ | $d$ | $e$ | $f$ | $p$ |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0.16 | 0 | 0 | 0.1 |
| 2 | 0.85 | 0.04 | $-0.04$ | 0.85 | 0 | 1.6 | 0.85 |
| 3 | 0.2 | $-0.26$ | 0.23 | 0.22 | 0 | 1.6 | 0.07 |
| 4 | $-0.15$ | 0.28 | 0.26 | 0.24 | 0 | 0.44 | 0.07 |

Try modifying the Random Iteration Algorithm to produce the attracting set! Rather than choosing the functions totally at random, choose function $i$ using probability $p_i$.

# 3 Appendix: Matlab Programs

**Code to create the word "Math"**

```
numits=80000;
a(1)=0.0;b(1)=-.08;c(1)=.62;d(1)=0.0;e(1)=-7.2;f(1)=5.0;
a(2)=-.07;b(2)=.05;c(2)=.39;d(2)=.005;e(2)=-7.3;f(2)=5.0;
a(3)=.07;b(3)=-.05;c(3)=.39;d(3)=.005;e(3)=-5.3;f(3)=5.0;
a(4)=0.0;b(4)=-.08;c(4)=.62;d(4)=0.0;e(4)=-4.4;f(4)=5.0;
a(5)=.10;b(5)=-.08;c(5)=.59;d(5)=.01;e(5)=-2.1;f(5)=4.7;
a(6)=.2;b(6)=0.0;c(6)=0.0;d(6)=.08;e(6)=-1.5;f(6)=3.9;
a(7)=-.10;b(7)=.08;c(7)=.59;d(7)=.01;e(7)=-1.1;f(7)=4.7;
a(8)=0.0;b(8)=-.08;c(8)=.59;d(8)=0.0;e(8)=2.4;f(8)=4.5;
a(9)=.215;b(9)=0.0;c(9)=0.0;d(9)=.08 ;e(9)=2.0;f(9)=9.1;
a(10)=0.0;b(10)=-.08;c(10)=.62;d(10)=0.0;e(10)=5.3;f(10)=5.0;
a(11)=.215;b(11)=0.0;c(11)=0.0;d(11)=.08;e(11)=6.4;f(11)=4.5;
a(12)=0.0;b(12)=-.08;c(12)=.625;d(12)=0.0;e(12)=8.0;f(12)=5.0;

y=zeros(2,numits);
x=rand(2,1);
z=zeros(2,1);
for i=1:100  %We'll be throwing away the first few points...
   k=ceil(12*rand);
      z(1)=a(k)*x(1)+b(k)*x(2)+e(k);
      z(2)=c(k)*x(1)+d(k)*x(2)+f(k);
      x=z;
end
%Keep these for plotting
for i=1:numits
   k=ceil(12*rand);
   z(1)=a(k)*x(1)+b(k)*x(2)+e(k);
   z(2)=c(k)*x(1)+d(k)*x(2)+f(k);
   y(:,i)=z;
   x=z;
end
```

**Code to create Barnsley's Fern**

```
numits=80000;
a=[0,0.85,0.2,-0.15];
b=[0,0.04,-0.26,0.28];
c=[0,-0.04,0.23,0.26];
d=[0.16,0.85,0.22,0.24];
e=zeros(1,4);
f=[0,1.6,1.6,0.44];
p=[2,86,93];

y=zeros(2,numits);
x=rand(2,1);
z=zeros(2,1);

for i=1:100  %We'll be throwing away the first few points...
      t=ceil(100*rand);
      if t<p(1), k=1;
      elseif t<p(2), k=2;
      elseif t<p(3), k=3;
      else, k=4;
      end

      z(1)=a(k)*x(1)+b(k)*x(2)+e(k);
      z(2)=c(k)*x(1)+d(k)*x(2)+f(k);
      x=z;
end
%Keep these for plotting
for i=1:numits
      t=ceil(100*rand);
      if t<p(1), k=1;
      elseif t<p(2), k=2;
      elseif t<p(3), k=3;
      else, k=4;
      end
   z(1)=a(k)*x(1)+b(k)*x(2)+e(k);
   z(2)=c(k)*x(1)+d(k)*x(2)+f(k);
   y(:,i)=z;
   x=z;
end
plot(y(1,:),y(2,:),'k.','Markersize',3);
axis([-5,5,0,10]);
```