Lab 3: Iterated Function Systems

Math 112 - Prof. Radunskaya

Go through the lab with a partner. (Each person should have their own computer if possible). TURN IN THE ANSWERS TO NUMBER 6 - include your partner's name. Do number 7 if you have time.

Iterated Function Systems are dynamical systems that are often used to generate fractals: sets with non-integer dimension. In this Lab we will explore some of these systems in the plane. Thus, our dynamical system will act on two-dimensional vectors, which we will denote by

$$p = \begin{bmatrix} x \\ y \end{bmatrix}$$

1. Let $p_0 = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$ be a fixed point in the plane, and define the function

$$A \begin{bmatrix} x \\ y \end{bmatrix} = \beta \begin{bmatrix} x - x_0 \\ y - y_0 \end{bmatrix} + \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$$

or, in vector notation:

$$A(p) = \beta(p - p_0) + p_0$$

Show that, if $0 < \beta < 1$ then p_0 is the unique attracting fixed point of A.
Hint: First show that A *contracts* distances, i.e. that $||A(p) - A(p_0)|| = \beta||p - p_0||$ where $||\cdot||$ denotes the *norm* or length of the vector in $\mathbb{R}^2$: $||\begin{bmatrix} x \\ y \end{bmatrix}|| = \sqrt{x^2 + y^2}$.

Definition 1 *Let $0 < \beta < 1$, and let $\{p_1, \ldots, p_n\}$ be a set of points in the plane. Let $A_i(p) = \beta(p - p_i) + p_i$ for $i = 1 \ldots n$. The set $\mathcal{A} = \{A_1 \ldots A_n\}$ is an* **iterated function system**.

An iterated function system (IFS) is a dynamical system of a special sort. The orbit of a point p_0 under the IFS is the sequence $p_0, p_1 = A_{X_1}(p_0), p_2 = A_{X_2}(p_1), \ldots$, where each A_{X_i} is a *random* choice from $\mathcal{A}$. (Note that by "random" choice, we mean that each A_i has equal probability of begin chosen.

Definition 2 *The set of points in the plane to which an arbitrary orbit of an IFS converges is the* **attractor** *for the system.*

Note: Since the A_i's are chosen randomly, we can only say that orbits converge *with a certain probability*. Therefore we should say that an arbitrary orbit converges *with probability 1* to the attractor.

2. **The Cantor Set as an Attractor**
We will use Matlab to generate the Cantor Middle Thirds set as the attractor of an IFS.

(a) Open Matlab and select your 112 folder as the Current Directory.

(b) **Matlab basics:**
The command "rand" generates a random number uniformly distributed between 0 and 1. Try it, type:

```
>> rand
```

Note: Don't type the command prompt ">>", and follow each line with "enter" or "return".

(c) If we want to generate a random integer between 1 and 47, say, we can multiply the rand number by 47 and "round up" using the "ceiling" function:

```
>> ceil(47*rand)
```

To generate a vector, v, containing 7 of these numbers:

```
>> v=ceil(47*rand(1,7))
```

Try this:

```
>> w = rand(3,4)
```

To generate a vector, say $p = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$, type:

```
>> p = [1;2]
```

TO DO: Try the same command with a comma instead of the semi-colon. Explain to your partner what difference this makes. Now try the same command with a semi-colon at the *end*. Explain to your partner what difference this makes.
Vectors can be added or subtracted, as long as they are the same size. They can also be multiplied by a scalar. Type the following to see what it does:

```
>> q = [3;4];
>> p+q
>> p-q
>> 47*(p-.5*q)
```

TO DO: Explain to your partner why the following will give you an error:

```
>> w = [-1, 7];
>> p + w
```

To switch from a column to a row vector, you can take the transpose:

```
>> ptrans = p'
```

We'll also want to store a bunch of vectors in a matrix, and access them one by one. To create a matrix, say, P, with two rows and three columns:

```
P = [1 2 3; 4 5 6]
```

Notice that the semicolon (";") means "start a new row".
To access the (i, j)th entry of P, type $P(i, j)$. So, to get the entry in the first row, third column, you type:

```
P(1,3)
```

To get the first column, use the shortcut ":" to mean "all", so "all rows, first column" gives the vector in the first column of P:

```
P(:,1)
```

(d) We will now create a file with the collection of A_i's. In the "File" menu, select "New M-file". This will open the Editor Window. Type the following:

```
function nextp = CantorIFS(p)


i=ceil(2*rand);
pset = [0 0; 1 0];
nextp = 1/3*p+2/3*pset(:,i);
```

Save this function as "CantorIFS.m" in your directory. (Note: you need the ".m" extension, or Matlab won't recognize it as a valid routine.)
TO DO: Explain to your partner what each line in this file does. There are two functions in this IFS (called A_1 and A_2). What are they?
Note that the *contraction factor*, β, is equal to $1/3$ in this IFS.
Let's check that the function does what we want it to. Define a point $p = [.5, .5]$. Calculate by hand what A_0 and A_1 would to to p, and see what your function CantorIFS does to it. Try it several times since it will randomly choose either A_1 or A_2. Recall: to repeat a previous command, use the "up" arrow.

```
>> p=[.5;.5];
>> CantorIFS(p)
```

Do the results agree with your predictions?

(e) We will now write an "outer" function that will iterate the CantorIFS as many times as you want and then plot the results. Open a new M-file and type the following. The lines beginning with a per-cent sign ("%") are comments. You don't need to type these in: they are supposed to serve a pedagogical purpose.

```
function Output=IterateIFS(IFS,n)
% Iterate the Iterated Function System stored in the file IFS
% n times and put the results in the
% array "Output".
% This will be a 2 by n matrix, where each column is a point on the orbit.

% Generate a random initial point, p.
```

```
p = rand(2,1);
% Store it in Output
Output = p;
% clear the figure and plot the initial point,
% set "hold" to "on" and set axis to 'square'
clf
plot(p(1),p(2),'r*')
hold on
axis('square')
% Iterate n times
for k = 1:n
    nextp = feval(IFS,p);
    Output = [Output, nextp];
    p = nextp;
    plot(p(1),p(2),'.')
    drawnow
end;
```

To run, make sure that the Figure window is visible, and type:

```
O = IterateIFS('CantorIFS',100);
```

The output is now stored in the matrix "O" (check it out! To see what O looks like, type its name followed by a carriage return.). The single quotes around 'CantorIFS' tells Matlab to interpret this as a string of letters, not the name of a variable. I have decided to plot the initial point with a red star, but you can change this (try it!).

The Cantor set is the *attractor* of the system, so the first few points along the orbit are not very close to it. If we want to get a cleaner picture of the attractor, we might want to ignore the first few points. So, suppose we want to ignore the first 50 points. We could do this by adding 50 "throwaway" iterates. Add these four lines to the "IterateIFS" file just after the one that says "hold on":

```
for k = 1:50
    nextp = feval(IFS,p);
    p = nextp
end;
```

Zoom in on the graph to see if it looks self-similar, as the Cantor Set should be!

In your text (pages 192–194) it is shown why the attractor of this IFS is the Cantor Middle Thirds Set. For now, we explore a few other systems.

3. Create an M-file (modify "CantorIFS" and save it as a new m-file) that describes the iterated function system $\mathcal{A} = \{A_1, A_2, \ldots A_5\}$ where $A_i = \beta(p - p_i) + p_i$, with contraction factor, $\beta = 1/3$ and the p_i's:

$$p_1 = [0; 0],\ p_2 = [1; 0],\ p_3 = [0; 1],\ p_4 = [1; 1],\ p_5 = [1/2; 1/2]\ .$$

If you get stuck, there is an example on Sakai, called "BoxFractalIFS.m" .

Look at the set of attracting points. Can you guess what the attractor of the IFS looks like? Use "IterateIFS" to validate your guess. You will need to use many more iterates (try 1000) to start to get a good picture of the attractor. **TIMESAVING TIP:** Comment out the "drawnow" line when doing many iterates or it will take forever. 5000 iterates gave me a pretty nice picture. (You should get something like Figure 14.7 in your textbook).

4. **Including rotations in an IFS** Allowing the linear functions in the IFS to include rotations as well as contractions produces variations on the attractors. A function that contracts by β and rotates by an angle θ can be written:

$$A\begin{bmatrix} x \\ y \end{bmatrix} = \beta \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x - x_0 \\ y - y_0 \end{bmatrix} + \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$$

Question: What is the fixed point of A?

Suppose the initial value is $p = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$. Describe the orbit of the function A described above with $\beta = .9$, $\theta = \pi/2$, $[x_0, y_0] = [1, 1]$. Check that you are correct by plotting some iterates one by one in Matlab. You can do this by first defining the rotation matrix, R:

```
theta = pi/2;
R = [cos(theta), -sin(theta); sin(theta), cos(theta)];
p = [2;2];
p0 = [1;1];
beta = .9;
p = beta*R*(p - p0) + p0;
plot(p(1),p(2),'.')
hold on
```

These commands will plot one iterate of the function. and set "hold" to "on". (Be sure that you clear any previous figures). Use the up arrow to plot a few iterates to see if your guess was correct.

5. **A general routine** To make your life easier, we will now write an M-file that will iterate a general IFS a given number of times. Open the M-file IterateIFS and save it as something different (e.g. IterateIFSgen.m). We will send the values of β, θ and *p*set (the set of attracting points) as *arguments* or inputs to the function. The output of the function will be the same, but - instead of calling a function that contains the IFS, it will call an *internal* function. Adapt your file so that it looks like:

```
function Output=IterateIFSgen(n,beta,theta,pset)
% Iterate the Iterated Function System described by
% beta (the contraction factor), theta (the rotation) and
```

```
% pset (the set of atrracting p-values)
% n times (after discarding the initial 50 iterates
% and output the results in the array "Output".
% Output will be a 2 by n matrix, where each column is a point on the orbit.

% Generate a random initial point, p.
p = rand(2,1);
% Store it in Output
Output = p;
% Iterate 50 times to get close to the attractor
for k = 1:50
    nextp = feval(@IFS,p,beta,theta,pset);
    p = nextp;
end;
% clear the figure, plot the last point, set "hold" to "on" and axis
% to 'square'.
clf
plot(p(1),p(2),'.')
hold on
axis('square')
% Iterate n more times
for k = 1:n
    nextp = feval(@IFS,p,beta,theta,pset);
    Output = [Output, nextp];
    p = nextp;
    plot(p(1),p(2),'.')
end;

function nextp=IFS(p,beta,theta,pset)

L = length(pset(1,:));  % how many attracting points
i = ceil(L*rand);
nextp = beta*[cos(theta), -sin(theta); sin(theta),cos(theta)]*(p-pset(:,i)) ...
              +pset(:,i);
```

Discuss with your partner what each line in the code does (ask if you have questions!) Note that I have removed the "drawnow" command since it slows things down too much (you'll want to do thousands of iterations to get a good picture). I have also removed the plotting of the initial point in red since we are now interested only in the attractor of the IFS. Finally, notice how the arguments are passed to the function. You can do this with any function - remember to send the arguments in the same order in which they are listed on the first line of the function.

Use this function to examine the attractor of the IFS with a contraction factor $\beta = 1/2$,

a rotation factor $\theta = \pi/4$ and attracting points:

$$p_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, p_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, p_3 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad .$$

Use 5000 iterations. You should get something like Figure 14.17 in your text.

6. On the following page you will find graphs of seven attractors that were generated with an IFS. The first six use no rotation, and the last one contains a rotation. Your mission is to figure out what the contraction factor is, what the attracting points p_i are, and - for the seventh example - what the rotation factor is. Check your answer by using the function you just wrote (you can set $\theta = 0$ for no rotation). Write down and turn in the following

   (a) In each case the IFS consists of a set of linear contractions in the plane. Draw a picture of what the contraction does in the unit square (see Figure 14.15 in the text for an example).

   (b) What is the fractal (self-similar) dimension of the attractor in each case?

7. **More Variations** More variations on attractors can be generated by allowing the rotation and/or the contraction factor to vary for each function in the IFS, and/or to choose the functions with different probabilities.

   (a) For example, the Levy Dragon can be created using the IFS consisting of the two functions:

   $$A_1(p) = \frac{1}{\sqrt{2}} R_{\pi/4} p; \quad A_2(p) = \frac{1}{\sqrt{2}} R_{-\pi/4}(p - p_2) + p_2$$

   where $p_1 = 0$ (so you don't see it in the equation) and $p_2 = (.5; .5)$. The notation R_θ denotes *counterclockwise* rotation by θ. Write a routine that will generate the Levy Dragon. (*If you get stuck, see IterateIFSgen2.m on Sakai.*)

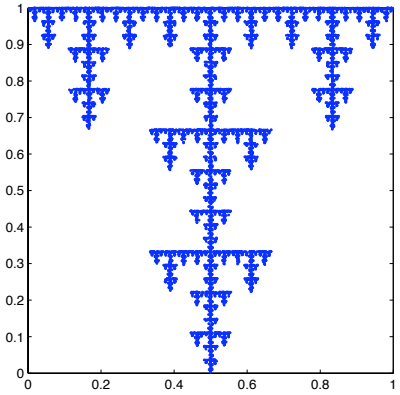
   (b) A fractal fern [1] combines different rotations and contractions with different probabilities. Write a routine that iterate the IFS with the following functions, where A_1 is picked with probability .85, A_2 with probability .07, A_3 with probability .07, and A_4 with probability .01.

   $$A_1(p) = \begin{bmatrix} .85 & .04 \\ -.04 & .85 \end{bmatrix} p + \begin{bmatrix} 0 \\ 1.6 \end{bmatrix}, \quad A_2(p) = \begin{bmatrix} .2 & -.26 \\ .23 & .22 \end{bmatrix} p + \begin{bmatrix} 0 \\ 1.6 \end{bmatrix},$$

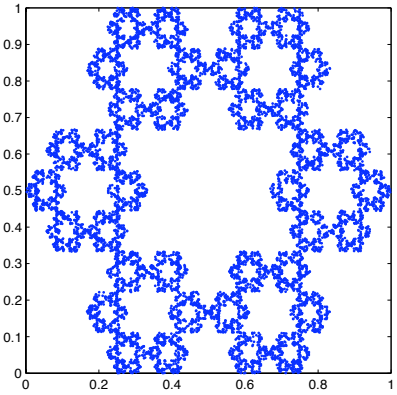
   $$A_3(p) = \begin{bmatrix} -.15 & .28 \\ .26 & .24 \end{bmatrix} p + \begin{bmatrix} 0 \\ .44 \end{bmatrix}, \qquad A_4(p) = \begin{bmatrix} 0 & 0 \\ 0 & .16 \end{bmatrix} p$$

   If you get stuck, see IterateIFSgen3.m on Sakai

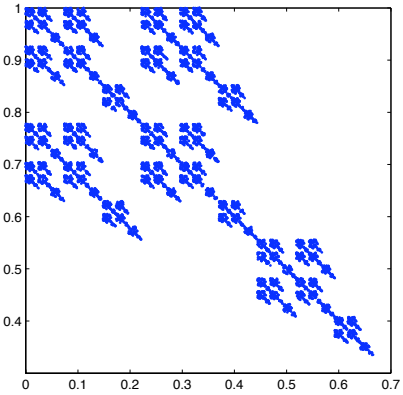
[1] See Barnsley's "Fractals Everywhere", (1993)



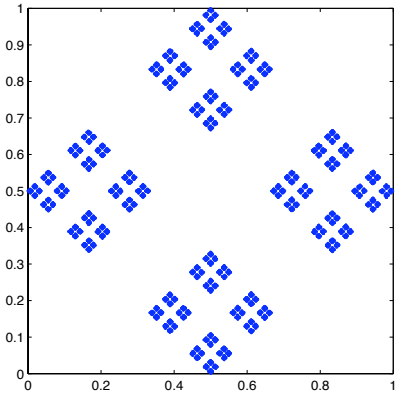
(I)



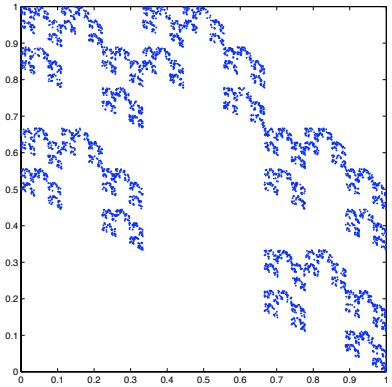
(II)



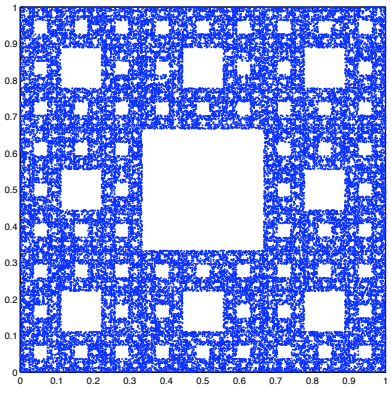
(III)



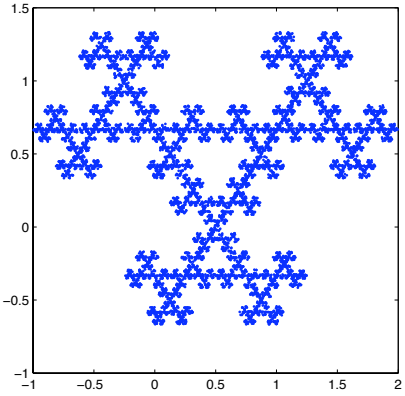
(IV)



(V)



(VI)



(VII)