Week 4 Matlab/Octave: Fractals

A concept that is fairly new (1970's) to geometry is something called a "fractal". Fractals typically share a phenomena known as "self similarity"- That is, the object is made up of smaller copies of itself.

In the natural world, we see this all the time. For example, if you break off a piece of cauliflower, you end up with a "little cauliflower". A cloud is made up of smaller copies of a cloud; mountains are made up of smaller mountains, and so on.

A great introduction to fractals is available on YouTube by the master himself, Benoit Mandelbrot. We are fortunate that he was able to give a "Ted" talk- he passed away shortly after that in 2010. I will provide a link to it on our class website, and I highly recommend it- It is very accessible to undergraduates.

Self Similarity

Mathematically, making a smaller copy of an object is something that a linear function can do. If we have several smaller copies created by functions $F_1, F_2, F_3, \cdots, F_k$, then an object A is said to be **self similar** if it is consists of the union of the smaller copies:

$$A = F_1(A) \cup F_2(A) \cup \cdots \cup F_k(A)$$

What would such an object look like? Below we'll create the object known as the Sierpinski Gasket, after the Polish mathematician Waclaw Sierpiński.

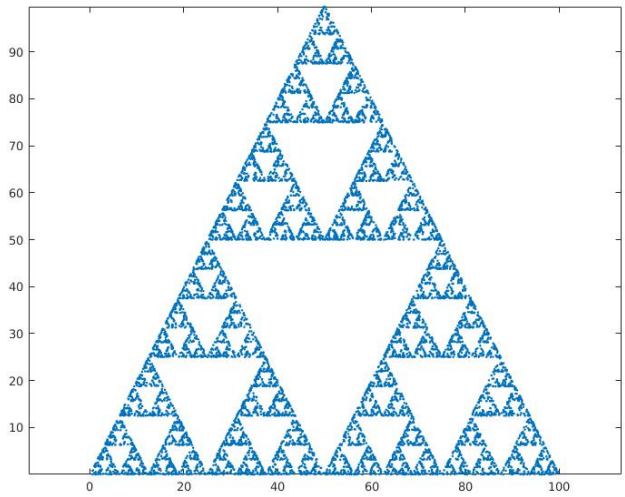


Figure 1: The Sierpinski Gasket. It can be made up of 3 smaller copies of itself.

Can we see the 3 copies for the Sierpinski Gasket? There is a lower left triangle, a lower right triangle and an upper triangle in the middle. Mathematically, if we put the origin at the leftmost lower point, and make the rightmost lower point be 100 (this is arbitrary), then:

- Function 1: Scale in x and y by $1/2$.
- Function 2: Scale in x, y by $1/2$, and then shift forward by 50.
- Function 3: Scale in x, y by $1/2$, and then shift forward by 25, up by 50.

In terms of linear algebra, each function is a mapping from $\mathbb{R}^2$ to $\mathbb{R}^2$. The function F_1 can be realized as:

$$F_1(x_1, x_2) = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Similarly,

$$F_2(x_1, x_2) = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 50 \\ 0 \end{bmatrix} \qquad F_3(x_1, x_2) = \begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 25 \\ 50 \end{bmatrix}$$

We might recall that F_2, F_3 are not strictly linear- They are "affine" maps, but they will be allowable for us. In any event, all affine maps from $\mathbb{R}^2$ to $\mathbb{R}^2$ can be written as:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix}$$

Therefore, every map is defined by those six values, so often we will just define the maps by listing those 6 numbers in a vector.

The way we'll construct the fractal is by using the "random iteration algorithm". That is, we'll select an initial point $\mathbf{x}$ at random from the plane, and we'll select a function at random to apply. That gives a new point in the plane- Select another function at random, apply it. Select another function, and so on. In this way, we construct what is called the orbit of $\mathbf{x}$, and we can plot it in Matlab. Here's a short script showing the code:

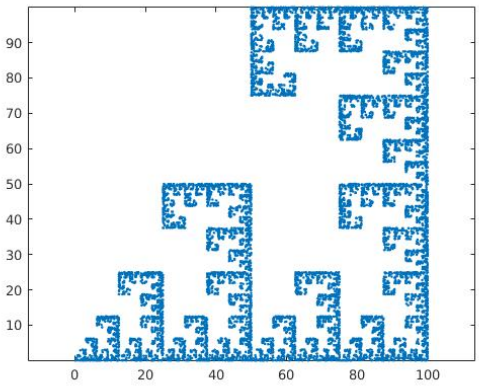
```
%% Plot the Sierpinski Gasket.

Numits=3001; % Specify the number of iterations (or number of points to graph)
NumFunc=3;   % Specify how many functions.

F=[0.5 0 0 0.5 0 0
   0.5 0 0 0.5 50 0
   0.5 0 0 0.5 25 50];

X=zeros(2,Numits);
% Initialize the first point randomly:
X(:,1)=randn(2,1);

for k=1:Numits-1
  idx=ceil(NumFunc*rand); % Select a function at random
% Build the matrix and vector for the function:
```

```
  A=[F(idx,1) F(idx,2); F(idx,3) F(idx,4)];  b=[F(idx,5); F(idx,6)];
  X(:,k+1)=A*X(:,k)+b;
end
plot(X(1,:),X(2,:),'k.');
```

How would we change the code for other self-similar objects? Be sure to consider `Numits` and especially be sure that *NumFunc* is the number of functions being used (in this case, we had three functions). Then, just change the 6 numbers for each function, and you're ready to input that in Matlab!

Another Example:

Consider the self-similar object shown below- Come up with the words that will describe how many functions we need, and the "action" of each function:



We'll need three functions again. One function will contract x and y by 1/2, just like the Sierpinski gasket, another will do the contraction, but then shift the result forward by 50 units. The third function (upper right) is a bit harder- We'll need to shrink by 1/2, but we'll need to then rotate by 90 degrees CCW, then shift forward by 100 and upward by 50. See if you can create the functions needed, and check with Matlab (see pg 72 in the text).

For homework, create the functions needed for the top 4 fractals on the next page, and turn in one (published) script for each. For fun, you might attempt the last two, although the fern can be a little tricky!