Universidad Rey Juan Carlos

Práctica número 2. Curso 2011-2012

Las prácticas propuestas aquí están realizadas con la ayuda del programa de simulación Matlab. Las prácticas también se podrán realizar con el programa libre Octave disponible para los sistemas operativos Linux y Windows.

Las prácticas consisten en la elaboración de ficheros de comando Matlab para la resolución de problemas. Los "scripts" tienen que ir acompañados por una memoria explicativa y por las figuras correspondientes a cada problema.

En esta práctica se van a construir algunos fractales.

1. Representación del conjunto de Mandelbrot

En este apartado vamos a representar el conjunto de Mandelbrot para un barrido de condiciones iniciales. El conjunto de Mandelbrot consiste en el conjunto de órbitas acotadas de la aplicación $z_{n+1} = z_n + c$ con la condición inicial $z_0 = 0$. Se representa entonces en un plano complejo las órbitas que permanecen acotadas con un color y las órbitas que divergen con otro color en función del parámetro c. Este problema acepta particularmente bien una formulación matricial. A continuación explicamos el procedimiento para crear estas matrices.

Primero se define la parte del plano complejo que usamos para definir el valor de $c = x + iy$ formando una matriz de valores iniciales gracias a la función `meshgrid`. Funciona de la forma siguiente, dado dos rangos de valores de longitud n y m la función `meshgrid` devuelve dos matrices de tamaño $n \times m$ cuyas filas o columnas son todas identicas. Cojamos por ejemplo los rangos $0 : 0{,}5 : 1$ y $-1 : 1 : 1$ y llamamos la función `meshgrid`.

Las matrices x e y tendrán el siguiente aspecto:

m \ n	1	2	3
1	0	0.5	1
2	0	0.5	1
3	0	0.5	1

x

m \ n	1	2	3
1	-1	-1	-1
2	0	0	0
3	1	1	1

y

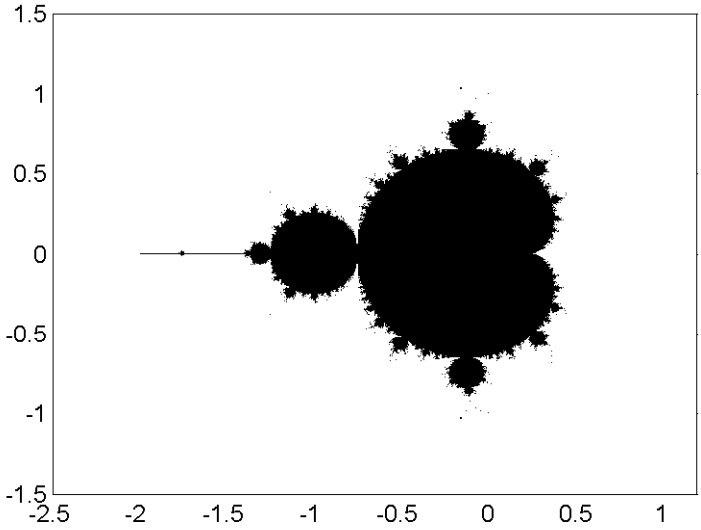

Figura 1: Representación del conjunto de Mandelbrot con escala 0.005 y 50 pasos, se ha elegido colormap(gray) para la visualización.

Se forman así todas las combinaciones posibles de las coordenadas de los dos rangos $0 : 0{,}5 : 1$ y $-1 : 1 : 1$. Creamos una matriz compleja $c = x + iy$ que contendrá todas las combinaciones de los dos rangos. Podemos ahora iterar la aplicación cuadrática con esta matriz:

```
% Script conjunto de Mandelbrot

% Definición de parámetros y reserva de memoria

número de iteraciones=50
mínimo valor de x=-2.5
máximo valor de x=1.2
mínimo valor de y=-1.5
máximo valor de y=1.5
escala=0.005
c=matriz compleja con todos los puntos entre los mínimos y máximos de x e y
 en pasos igual a la escala (utilizar función meshgrid)
w=matriz vacía de las mismas dimensiones que c

% Iteración de la matriz e imposición de condición de pertenencia al conjunto

Para k=1 hasta número de iteraciones
```

```
w=w*w+c
Fin

índices= valores de w con valor absoluto < 4 (usar función find)
resultado=matriz vacía de las mismas dimensiones que c
resultado(índices)=1

% Representación del conjunto de Mandelbrot

Usar funciones imagesc o pcolor (ver opción shading flat)
```

Ejercicio 1 : Programar el script anterior y representar con él el conjunto de Mandelbrot tal y como aparece en la figura 1. Ver cómo varía su forma con el número de iteraciones y la resolución.

2. Representación de curvas fractales

En esta sección se propone realizar programas en Matlab para representar la construcción de fractales simples tales como el copo de Koch.

Para construir este fractal vamos a usar una vez más las propiedades de los números complejos. Construimos el patrón original como un vector de números complejos. Si representamos este vector con la función `plot(motivo)`, reconocemos el motivo básico del copo de Koch. Ahora para construir la curva, elegimos como vector inicial el segmento [0,1]. En la etapa k de la construcción de la curva, el algoritmo es el siguiente:

1. Calculamos el ángulo α que forma el segmento que estamos estudiando con el eje real.
2. Giramos el patrón elemental de un ángulo α, lo escalamos de un factor $1/3^{k-1}$ y lo trasladamos al primer punto del segmento que estamos tratando.
3. Repetimos para el siguiente segmento hasta llegar al punto (1,0).

Una vez llegado al punto (1,0) se han procesado todos los segmentos, podemos repetir el proceso en la etapa $k+1$ (ver figura 2).

```
% Programa para representar el copo de Koch

% Definición del motivo y del vector de traslación
motivo = [0   0.3333    0.5000+0.2887i   0.6667   1.0000]

v=[0 1] % segmento sobre el que dibujaremos el copo de Koch
```

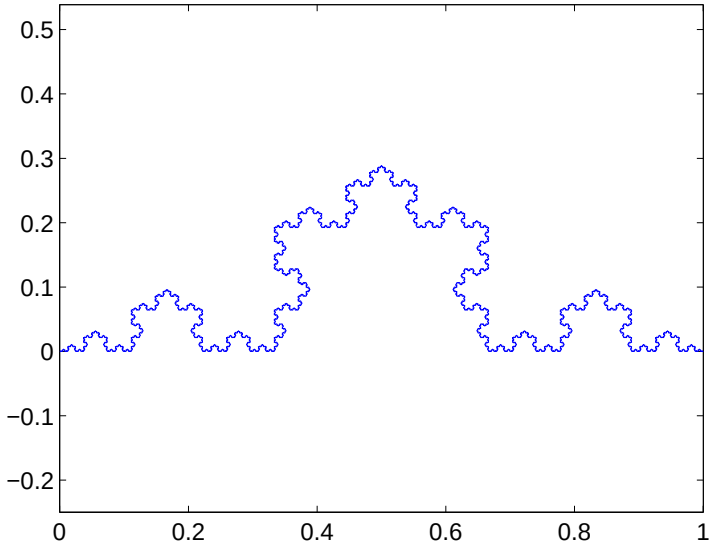

Figura 2: Copo de Koch sobre el segmento unidad, se han utilizado 6 iteraciones.

```
número de iteraciones=6

Para k=1 hasta número de iteraciones

    temporal=v(1);

    Para n=1 hasta longitud de v-1

        rotación=exponencial(i*ángulo que forman v(n+1) y v(n))
        traslación=v(n);
        escala=(1/3)^(k-1)
        temporal=[temporal, escala*rotación*motivo(2:end)+traslación];

    end

    v=temporal;

end

% Dibujar el vector v
Para obtener un mejor resultado es conveniente usar axis equal
```

Ejercicio 2 :

1. Representar el copo con 6 lados hexagonales como curva inicial. Para este ejercicio existen dos alternativas: escribir el vector v punto por punto de tal modo que sus puntos formen un hexágono regular, o bien escribir una pequeña función en Matlab que devuelva los vértices de cualquier polígono regular:

   ```
   Función: PolígonoRegular(número de lados)

   polígono=vector vacío de longitud igual al número de lados % reserva de me
   ángulo=2*pi/número de lados

   Para k=1 hasta número de lados
   polígono(k)=exponencial(i*k*ángulo)
   Fin

   polígono=polígono/longitud del lado del polígono % normalización
   ```

2. Imaginar otro patrón elemental y representarlo.

3. Sistemas de funciones iteradas (IFS)

Se puede formar un patrón auto similar con unas reglas de transformación lineal simple. Por ejemplo iterando la siguiente aplicación lineal:

$$\begin{aligned} x_{n+1} &= ax_n + by_n + c \\ y_{n+1} &= dx_n + ey_n + f \end{aligned} \tag{1}$$

Los parámetros $a \ldots f$ se eligen según un algoritmo que involucra aleatoriedad. Generamos un número r aleatoriamente entre [0,1], dependiendo del resultado elegimos el conjunto de parámetros:

$$\begin{array}{ll} \text{si } r < 0{,}05 & \rightarrow (a,b,c,d,e,f) = (0,0,0,0,0{,}2,0) \\ \text{si } r < 0{,}86 & \rightarrow (a,b,c,d,e,f) = (0{,}85,0{,}05,0,-0{,}04,0{,}85,1{,}6) \\ \text{si } r < 0{,}93 & \rightarrow (a,b,c,d,e,f) = (0{,}2,0{,}26,0,0{,}23,0{,}22,1{,}6) \\ \text{otro caso} & \rightarrow (a,b,c,d,e,f) = (-0{,}15,0{,}28,0,0{,}26,0{,}24,0{,}44) \end{array} \tag{2}$$

Iteramos el mapa y representamos el conjunto de puntos.

Ejercicio 3 :

1. Hacer una función para la aplicación lineal, que tenga como inputs los coeficientes a...f y el punto (x_n, y_n) y devuelva el punto (x_{n+1}, y_{n+1}). Escribir un programa que llame a dicha función, cambiando los parámetros en función de un número aleatorio (véase la función rand) e itere el mapa. Representar los puntos (x,y).

2. Cambiar parámetros de las transformaciones afines y representar el resultado.

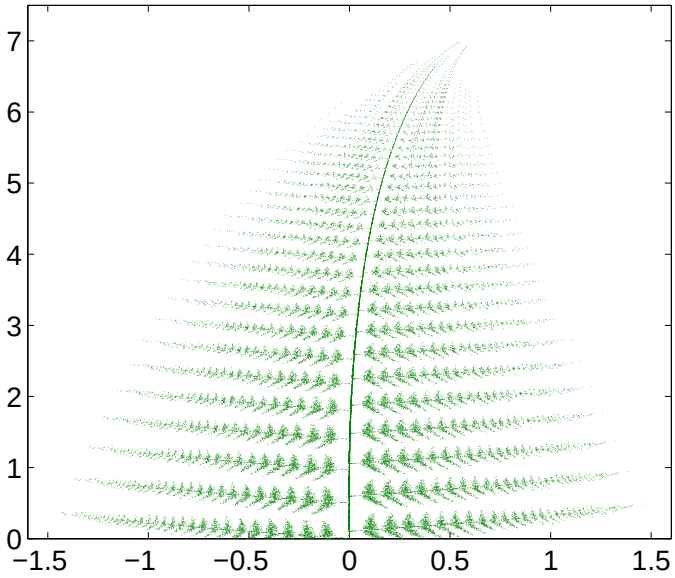

Figura 3: IFS con otros parámetros distintos a los indicados en el texto.