PROGRAMACIÓN EN MATLAB

PRÁCTICA 05
GRÁFICOS EN MATLAB 2D
GRÁFICOS EN MATLAB 3D
TRATAMIENTO DE IMÁGENES EN MATLAB
ALGORITMOS RECURSIVOS

En primer lugar crea un directorio llamado ***Practica05*** en tu directorio ***G:\Informatica1***. En este directorio deberás guardar todos los ficheros y todo el trabajo resultante de esta práctica. Después arranca el programa ***MATLAB*** y coloca como ***directorio de trabajo*** o ***directorio activo*** el directorio creado previamente.

EJERCICIO 1. GRÁFICOS EN MATLAB BIDIMENSIONALES

Matlab posee una serie de comandos para crear interfaces gráficas de manera muy sencilla. Un resumen de todos estos comandos se encuentran en los Capítulos 6, 7, 9 y 10 del ***manual "Aprenda Matlab como si estuviera en Primero***".

Matlab dispone de funciones gráficas de ***alto*** y ***bajo nivel***. Las funciones de alto nivel (***plot***, ***plot3***, ***surf***, ***mesh***, etc.) permiten representar gráficamente funciones o conjuntos de datos complejos (en forma de matrices de coordenadas ***x***, ***y*** y ***z***) con una sola llamada a una función. Las funciones de bajo nivel (***line***, ***patch***, etc.) permiten dibujar gráficos elementales (una línea, un polígono, etc.) o gráficos complicados a base de muchas llamadas para añadir cada uno de los gráficos elementales.

Crea un subdirectorio dentro de ***G:\Informatica1\Practica05*** llamado **Ejercicio01** y dentro de él, guarde los siguientes ejercicios de creación de gráficos.

EJERCICIO 1.1 Utilización del comando "line"

Construya un nuevo subdirectorio que se llame ***Ejercicio0101*** y sitúe, como siempre, el directorio de trabajo de Matlab dentro**.**

El comando ***line*** permite dibujar líneas. Los parámetros que espera esta función son dos vectores: el primero con las coordenadas ***x*** y en el segundo con las coordenadas ***y***. La forma general para dibujar una línea es la siguiente:

```
» line([xini xend],[yini yend]);
```

Así, la siguiente instrucción dibuja una línea que une el punto (0,0) con el (2,1):

```
» line([0 2],[0 1]);
```

Si se dan más de dos puntos ***Matlab*** dibuja una ***línea poligonal*** continua uniendo cada punto adicional con el anterior. Ejecútese el siguiente ejemplo:

```
» x = [1 2 3 4]'
» y = [1 3 2 4]'
» line(x,y);
```

Si se desea dibujar varias líneas pero independientes unas de otras (el comienzo de un nuevo segmento no es el final del anterior) se pueden también introducir las coordenadas de los puntos en forma de matriz: ***Matlab*** unirá los puntos por columnas. La siguiente instrucción une el punto (0,0) con el (1,1) y el punto (0,2) con el (2,0):

```
» X=[0 0
     1 2]
» Y=[0 2
     1 0]
» line(X, Y)
```

Cada columna de **X** contiene la *xini* y la *xend* de cada una de las líneas a dibujar, y lo mismo con las columnas de la matriz **Y**.

De forma análoga, también se pueden introducir los puntos con coordenadas 3D, añadiendo como argumento el vector o la matriz de coordenadas correspondiente a la componente ***z***:

```
» x = [0 1]
» y = [0 1]
» z = [0 1]
» line(x, y, z)
» view(-37.5, 80)
```

El comando ***line*** permite, asimismo, determinar el color de la línea, añadiendo pares de valores ***'color', 'w'*** (con el código de colores que se indica en la página 57 del Manual). Así, para que la primera línea apareciera en rojo, habría que teclear:

```
» line([0 2], [0 1], 'color', 'r');
```

De forma análoga se pueden añadir o modificar otras propiedades (grosor de línea, tipo de trazo, etc.), según se explica en el Capítulo 9 del Manual.

EJERCICIO 1.2 Utilización del comando "patch"

Construya un nuevo subdirectorio que se llame ***Ejercicio0102*** y sitúe, como siempre, el directorio de trabajo de Matlab dentro**.**

El comando ***patch*** se emplea para dibujar polígonos en dos o tres dimensiones, que pueden estar rellenos de un color constante o variable (interpolado entre los colores de los vértices). Se

distingue entre el color de las aristas y el color de relleno. Teclea y ejecuta el fichero ***patch1.m*** que se indica a continuación:

```
% fichero patch1.m
% ejemplo de la función patch en 2D
clear all
close all
x1=[0 1 0];
y1=[0 0 2];
patch(x1,y1,'b');
axis([0,3,0,3]);
```

EJERCICIO 1.3 Dibujo de una función con Plot y Subplot

Construya un nuevo subdirectorio que se llame ***Ejercicio0103*** y sitúe, como siempre, el directorio de trabajo de Matlab dentro**.**

En este ejercicio vamos a aprender a utilizar la función Subplot y Plot. La función Subplot(m,n,p) permite dividir el área de dibujo en una matriz de $m_x n$ rectángulos. Con el parámetro ***p*** seleccionamos cuál de ellas vamos a utilizar para dibujar, lo cual haremos con la función ***Plot***.

A continuación se muestra el ejemplo a seguir para obtener dos gráficas, una con una señal senoidal y la otra con una señal coseno, del doble de frecuencia que la primera. Para el desarrollo de este ejercicio debes copiar el código en un fichero ***Ejercicio0103.m***, usando el editor de ***Matlab***. Luego podrás ejecutarlo desde ***Matlab***, como ya lo sabes hacer.

```
%gráfica de dos funciones Sin y Cos
amp1=10;
amp2=5;
num=100;
ciclos=2;
f1 =100;      % Frecuencia de 100Hz
f2 =2*f1;     %el doble
t=0:ciclos/(f1*num):ciclos/f1;
%calculo de funciones
y1=amp1*sin(2*pi*f1*t);
y2=amp2*cos(2*pi*f2*t);

%dibujar funciones
subplot(2,1,1);     %divide en 2 filas y una columna la pantalla y selecciona
la primera de las dos figuras
plot(t,y1,'Color','r');
grid;
title('Funcion Seno de 100Hz','Color','b');
subplot(2,1,2);          %Selecciona la segunda de las dos figuras
plot(t,y2,'Color','r');
grid;
title('Funcion Coseno de 2*100Hz','Color','b');
xlabel('Tiempo en Segs.','Color','r');
```

Como ejercicio adicional, prueba obtener una *tercera gráfica* que sea el producto de las anteriores. A ver qué obtienes...

EJERCICIO 1.4 Dibujar un cuadrilátero inscrito en una circunferencia de radio 1

Construya un nuevo subdirectorio que se llame ***Ejercicio0104*** y sitúe, como siempre, el directorio de trabajo de Matlab dentro**.**

Se pide crear un fichero llamado ***circunferencia.m*** que dibuje un cuadrado inscrito en una circunferencia de radio unidad, según se muestra en la Figura 1. Para realizar este ejercicio se deberá emplear la función ***line***.

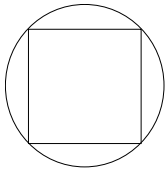


Figura 1. Cuadrado inscrito en una circunferencia.

Debe ser consciente de que Matlab trabaja de manera numérica, por lo que para dibujar la circunferencia deberá discretizarla en una serie de líneas que la bordeen. Suponiendo que se dividan los 2π radianes que componen la circunferencia en trocitos de θ radianes cada uno, las coordenadas {x, y} de una línea de las que bordean a la circunferencia de radio 1 entre dos partes consecutivas $\{\theta_k, \theta_{k+1}\}$ serían:

$$x_k=\cos(\theta_k) \quad y_k=\sin(\theta_k)$$

$$x_{k+1}=\cos(\theta_{k+1}) \quad y_{k+1}=\sin(\theta_{k+1})$$

De manera que el dibujo de la línea a la que nos referimos será:

$$\text{line}([x_k\ x_{k+1}],[y_k\ y_{k+1}])$$

La circunferencia debe quedar discretizada en **100 puntos.**

EJERCICIO 2. GRÁFICOS EN MATLAB TRIDIMENSIONALES

Crea un subdirectorio dentro de ***G:\Informatica1\Practica05*** llamado **Ejercicio02** y dentro de él, guarde los siguientes ejercicios.

EJERCICIO 2.1 Dibujar una pirámide de base cuadrada

Construya un nuevo subdirectorio que se llame ***Ejercicio0201*** y sitúe, como siempre, el directorio de trabajo de Matlab dentro**.**

Se pide crear una función de Matlab que, utilizando la función ***patch***, dibuje una pirámide de base cuadrada en función de la altura y la longitud de los lados de la base que le marque el usuario. Su encabezado deberá ser:

```
function Piramide(L,h)
```

Donde **L** debe corresponderse con la longitud de los lados de la pirámide y **h** con la altura.

EJERCICIO 2.2 Dibujo de la función z=sin(x)*cos(y)

Construya un nuevo subdirectorio que se llame ***Ejercicio0202*** y sitúe, como siempre, el directorio de trabajo de Matlab dentro**.**

El programa que debe desarrollar a continuación debe guardarlo dentro de un fichero de instrucciones llamado **SinXCos.m.**

Utilizando el comando ***Surf***, debe dibujar la función tridimensional Z=sin(x)*cos(y). Para ello deberá seguir los siguientes pasos:

1º Crear un vector ***x,*** cuyo valor inicial y final sea -3 y 3 respectivamente, y con un paso de 0.1.

2º Crear un vector ***y,*** cuyo valor inicial y final sea -3 y 3 respectivamente, y con un paso de 0.1.

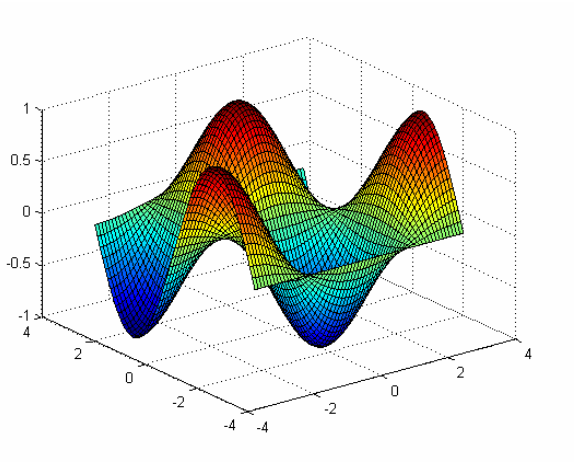


Figura 2.1. Función tridimensional dibujada en Matlab con la función SURF.

3º Crear la matriz ***Z***, cuyos valores se correspondan con los valores de la función matemática concretados para los valores definidos en los vectores **x** e **y**.

4º Pasarle ***x***, ***y*** y ***Z*** a la función ***Surf***.

EJERCICIO 3. TRATAMIENTO DE IMÁGENES EN MATLAB

La digitalización de una imagen permite realizar operaciones numéricas sobre ella. Esto es lo que realizan las distintas funciones implementadas en los programas para tratamiento de imágenes. Mediante las distintas operaciones se pueden realizar efectos especiales sobre una imagen, eliminar ruido, darle más o menos luz a una fotografía, etc.

En el directorio **Q:\Informatica1\Practica05** se dispone de una serie de archivos gráficos del tipo **JPG** con los que puede probar los programas que se piden a continuación. Crea un subdirectorio dentro de ***G:\Informatica1\Practica05*** llamado **Ejercicio03** y dentro de él, guarde los siguientes ejercicios.

EJERCICIO 3.1 Obtención del negativo de una imagen

Construya un nuevo subdirectorio que se llame ***Ejercicio0301*** y sitúe, como siempre, el directorio de trabajo de Matlab dentro**.**

Realice una función cuyo encabezado sea:

function negativo(archivo)

Donde ***archivo*** es una cadena de caracteres con el nombre completo del archivo (incluida su extensión). Esta función debe abrir dos figuras, en la primera figura dibujará la imagen contenida en el archivo y en la segunda figura el negativo de la misma imagen.

Solución: El código podría ser el siguiente:

```
function negativo(archivo)
n=length(archivo); % n=numero de letras de la cadena 'archivo'
formato=archivo(n-2:n); %formato guarda las tres ultimas letras. Sirven BMP, JPG y TIF.
```

```
Colores=imread(archivo,formato) ; %Lectura del archivo
Colores=double(Colores); %Paso a double
Colores=Colores/255; %Escalado
figure;image(Colores) %Dibujo del JPG
axis equal %Escalado horizontal y vertical
x=size(Colores,2); % x=numero de pixeles horizontales
y=size(Colores,1); % y=numero de pixeles verticales
axis([1 x 1 y]) %Encuadre de la imagen
drawnow %Se obliga al programa a dibujar lo que se ha pedido
for c=1:x %Bucle para recorrer las columnas de la imagen
   for f=1:y %Bucle para recorrer las filas de la imagen
      for k=1:3 %Bucle para recorrer los 3 colores RGB de cada pixel
         Colores(f,c,k)=1-Colores(f,c,k); %Operacion
      end
   end
end
figure;image(Colores) %Dibujo de la nueva imagen obtenida
axis equal %Escalado horizontal y vertical
axis([1 x 1 y]) %Encuadre de la imagen
```

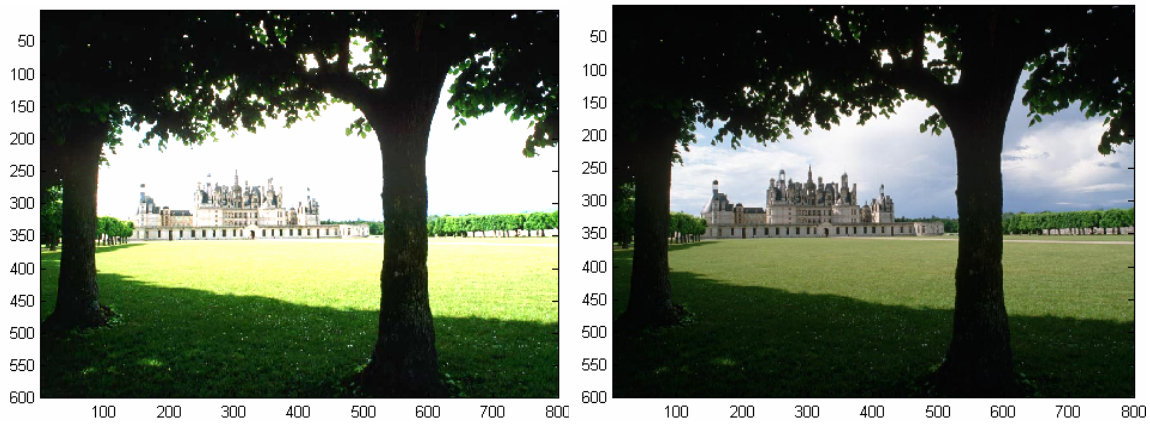
EJERCICIO 3.2 Modificación del brillo de una imagen

Construya un nuevo subdirectorio que se llame ***Ejercicio0302*** y sitúe, como siempre, el directorio de trabajo de Matlab dentro**.**

Realice una función cuyo encabezado sea:

function brillo(archivo, porcentaje)

Donde ***archivo*** es, de nuevo, una cadena de caracteres con el nombre completo del archivo y ***porcentaje*** es el % que se quiere incrementar la intensidad de los colores de la imagen. Esta función debe abrir dos figuras, en la primera dibujará la imagen del archivo y en la segunda dibujará la misma imagen con el incremento en la intensidad pedida por el usuario.



EJERCICIO 3.3 Dibujo de la imagen en blanco y negro y escritura en un fichero

Construya un nuevo subdirectorio que se llame ***Ejercicio0303*** y sitúe, como siempre, el directorio de trabajo de Matlab dentro**.**

Realice una función cuyo encabezado sea:

function EscalaGrises(archivo_origen,archivo_destino)

Donde ***archivo_origen*** es, como siempre, una cadena de caracteres con el nombre completo del archivo y ***archivo_destino*** es el nombre completo del archivo en el que el usuario desea guardar la

imagen en escala de grises. Esta función debe abrir dos figuras, en la primera dibujará la imagen del archivo y en la segunda dibujará la misma imagen en escala de grises. Para dibujar en escala de grises cada píxel de la imagen debe tener la misma cantidad de Rojo, Verde y Azul. Así que lo que debe hacer es calcular la media de la composición RGB de cada píxel y asignar esta media a los tres valores RGB.

Para escribir el fichero con el nombre puesto por el usuario, habría que finalizar el programa con las instrucciones:

```
Colores=Colores(:,:,1);%Se recoge sólo un color ya que los tres coinciden al ser escala de grises
Colores=Colores*255; %Se reescala a 255
Colores=uint8(Colores); %Se convierten a enteros sin signo de 8 bits
imwrite(Colores,archivo_destino,formato) %Se guarda la imagen en el archivo de destino
```

EJERCICIO 4. ALGORITMOS RECURSIVOS

Para programar algoritmos recursivos es muy importante tener dominadas las **funciones.** Si no es así, debe repasarlas hasta que tengas claro cómo funcionan.

Además, es muy interesante seguir el programa con el DEBUGGER, observando como se suceden las distintas llamadas recursivas.

Crea un subdirectorio dentro de ***G:\Informatica1\Practica05*** llamado **Ejercicio04** y dentro de él, guarde los siguientes ejercicios.

EJERCICIO 4.1 Los números de Fibonacci

Construya un nuevo subdirectorio que se llame ***Ejercicio0401*** y sitúe, como siempre, el directorio de trabajo de Matlab dentro**.**

La serie de Fibonacci es la siguiente: 0 1 1 2 3 5 8 13 21...

La expresión general de la serie de Fibonacci para un número **n** se define de la siguiente manera:

$$fibonacci(n) = \begin{cases} 0 & si\ n = 0 \\ 1 & si\ n = 1 \\ fibonacci(n-1) + fibonacci(n-2) & en\ los\ otros\ casos \end{cases} \quad (1)$$

Lo que se corresponde con una función expresada de manera recursiva. Programe, siguiendo la regla recursiva de la ecuación (1), una función que devuelva un vector ***(Serie)*** con la serie de Fibonacci para cualquier número n que se pida. La programación se realizará en un fichero de Matlab llamado **Fibonacci.m** cuyo encabezado es:

function Serie=fibonacci(n)

EJERCICIO 4.2 Algoritmo de Euclides

Construya un nuevo subdirectorio que se llame ***Ejercicio0402*** y sitúe, como siempre, el directorio de trabajo de Matlab dentro**.**

El algoritmo que va a programar a continuación data del año 200 A.C. Sirve para obtener el máximo común divisor (MCD) de dos números. El máximo común divisor de dos números **a** y **b** es aquel número entero de mayor valor que divide simultáneamente de manera entera a **a** y a **b**. Supongamos que **a>=b>0.** En ese caso resulta que:

$$Para\ a \geq b \geq 0, el\ MCD\ de\ a\ y\ b\ es = \begin{cases} b & si\ b\ divide\ a\ a\ \text{sin}\ resto \\ MCD\left(a, resto\ de\ \dfrac{a}{b}\right) \end{cases} \quad (2)$$

La afirmación expuesta en la ecuación 2 tal vez no resulte muy intuitiva. Sin embargo, no solamente es cierto, sino que además es verdad. La ecuación 2 vuelve a representar un sistema recursivo. Programe, siguiendo la regla recursiva de la ecuación (2), una función que devuelva el máximo común divisor entre dos número **a** y **b** siendo **a>=b>0.** Como hay que obtener el resto de la división, utilice la función **Resto** programada en el **ejercicio 3.5** de la **Práctica 5.** La programación se realizará en un fichero de Matlab llamado **MCD.m** cuyo encabezado es:

function d=MCD(a,b)

Siendo **d** el máximo común divisor de **a** y **b**.

EJERCICIO 4.3 El cuadrado de Sierpinski

Construya un nuevo subdirectorio que se llame ***Ejercicio0403*** y sitúe, como siempre, el directorio de trabajo de Matlab dentro**.**

NOTA: Este ejercicio sólo debe realizarse si se han terminado todos los ejercicios de las anteriores prácticas.

La construcción del cuadrado de Sierpinski es similar a la del triángulo, cuya explicación la encontramos en el libro **"Aprenda a programar como si estuviera en primero"**. En este caso se parte de un cuadrado de dimensión **h**x**h.** Este cuadrado simple, se correspondería con un **Cuadrado de Sierpinski de orden 0**. Para realizar un cuadrado de Sierpinski de orden 1, hay que dibujar otros 8 cuadrados de tamaño h/3 rodeando simétricamente el cuadrado inicial, según las medidas que se indican en la figura:

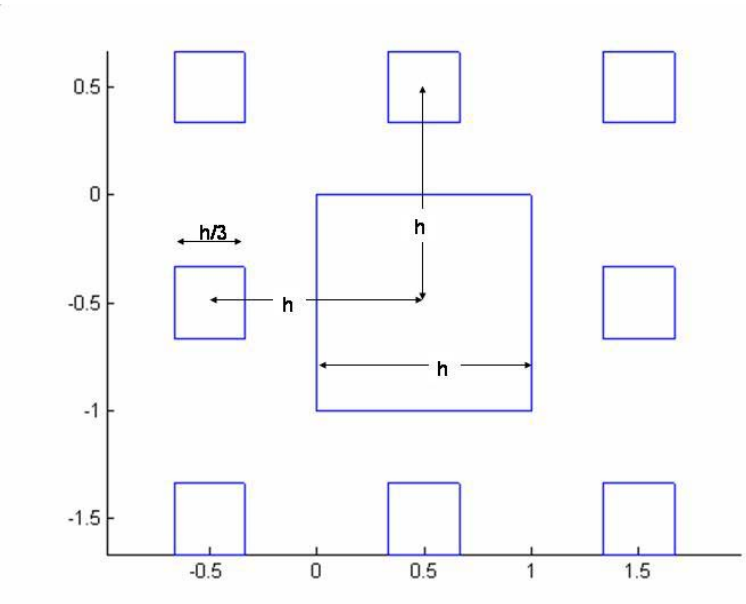


Figura 4.1 Cuadrado de Sierpinski de orden 1

Como puede observarse, el valor inicial asignado a **h** ha sido de 1.

Posteriormente, si se repite el proceso sobre cada uno de los 8 cuadrados recientemente dibujados, se obtiene el Cuadrado de Sierpinski de orden 2:

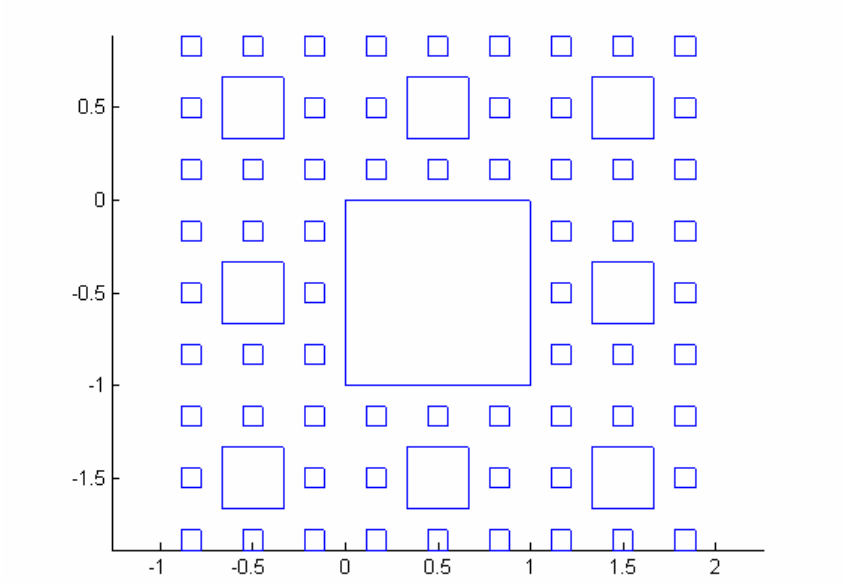


Figura 4.2 Cuadrado de Sierpinski de orden 2

y así sucesivamente:

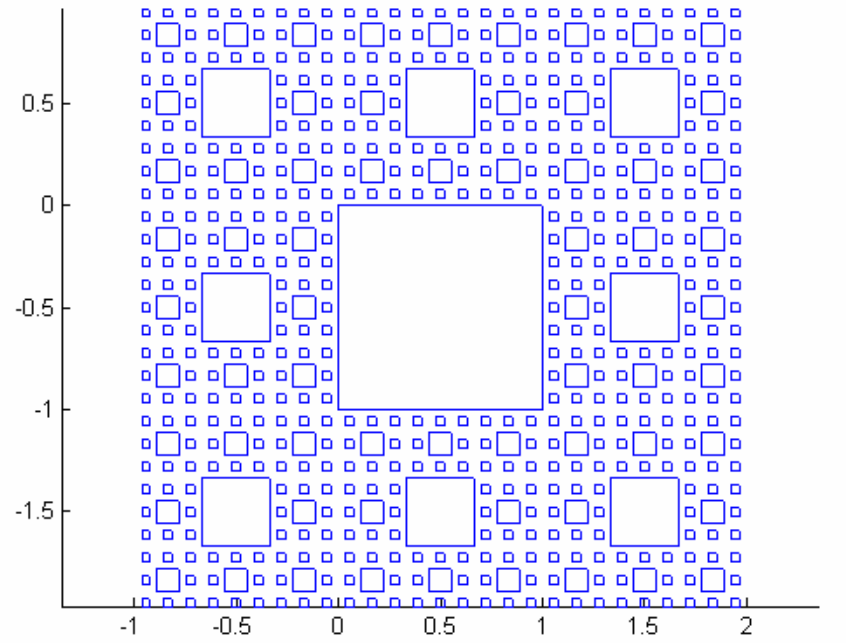


Figura 4.3 Cuadrado de Sierpinski de orden 3

De forma que el Cuadrado de Sierpinski puede ser dibujado, al igual que el triángulo, mediante un proceso recursivo en el que hay una medida clave que es **h** a la cual reducimos progresivamente su valor un tercio en cada llamada recursiva.

Establezca usted, basándose en el Triángulo de Sierpinski y en las explicaciones anteriores, el sistema de dibujo del Cuadrado de Sierpinski y establezca una función cuyo encabezado sea:

function CuadradoSierpinski(n)

La cual deberá dibujar un Cuadrado de Sierpinski de orden **n**.