Introducción a Matlab.

Ejercicios básicos de manipulación de imágenes.

Departamento de Ingeniería electrónica, Telecomunicación y Automática.
Área de Ingeniería de Sistemas y Automática

OBJETIVOS:

- Iniciación en la utilización de la herramienta Matlab. Algunos ejemplos de la utilización de la toolbox de procesado de imagen.

RESUMEN:

Se realizará una breve introducción del programa Matlab. Es imprescindible familiarizarse con esta herramienta a la hora de utilizar una de sus "Toolboxes", en concreto se trabajará con la toolbox de procesado de imagen.

1. INTRODUCCIÓN:

Matlab es la abreviatura de Matrix Laboratory (laboratorio de matrices). Creado en 1984 por The MathWorks, es un software de cálculo muy usado en universidades, centros de investigación y por ingenieros. En los últimos años ha incluido muchas más capacidades, como la de programar directamente procesadores digitales de señal, crear código, etc.

A continuación se muestran una serie de ejercicios "guiados" para permitir al alumno conocer y manejar la interfaz de este programa.

1.1 Interfaz:

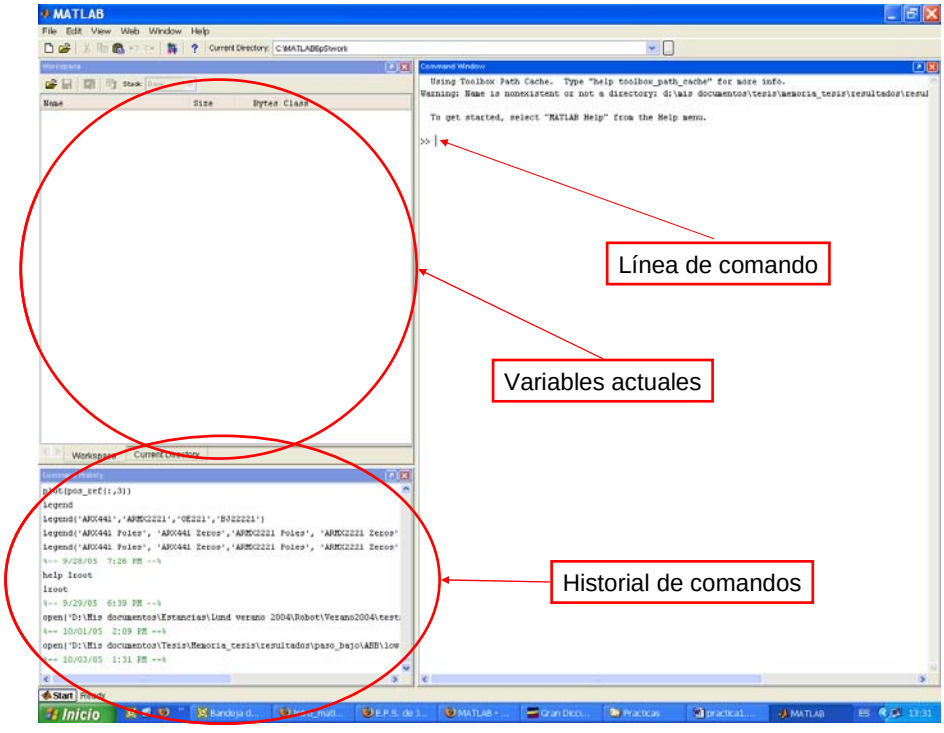


Figura 1. Interfaz de Matlab.

1.2 Ayuda de Matlab.

Básicamente, existen dos formas de utilizar la ayuda de Matlab: a través de la ayuda en línea; o bien, a través del navegador de ayuda.

Para acceder a la ayuda en línea basta con teclear en la línea de comandos:

```
>> help funcion
```

donde "funcion" sería el nombre de la función sobre la que necesitamos la ayuda.

Por otro lado, para acceder a la ayuda a través del navegador, es necesario seleccionar la opción "Matlab help" (Figura 2). Este segundo modo de ayuda resulta bastate más potente y eficaz que la primera añadiendo en muchos casos ejemplos de utilización.

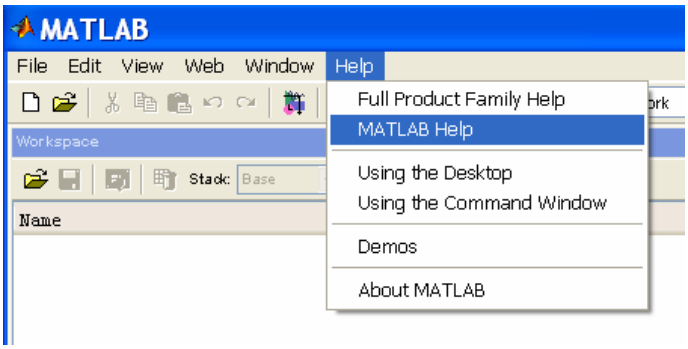


Figura 2. Menú de ayuda

1.3 Variables y matrices.

Matlab no requiere ningún tipo de declaración de variables sino que, una vez que se utiliza una variable, Matlab crea la respectiva variable reservando el espacio de memoria necesario. Por tanto, si la variable ya existe, Matlab únicamente cambia su contenido.

En lo que se refiere a la nomenclatura de las variables. Matlab distingue entre mayúsculas y minúsculas ("Variable" es distinto de "variable") permitiendo nombres de variables que contengan al menos una letra.

En Matlab, una variable consiste en una matriz de las dimensiones correspondientes. En cuanto al tipo de variables a utilizar puede ser: entero, real, complejo, carácter, etc., y al igual que en la definición, Matlab lo asigna de forma automática.

Los operadores aritméticos básicos son:

Símbolo	Operación
+	Suma
-	Resta
*	Multiplicación
/	División (2/1 = 2)
\	División en sentido contrario (2\1 = 0.5)
^	Potencia
‘	Transpuesta
()	Paréntesis

Ejemplo de utilización de una variable:

```
>> a = 5
a =
    5
>> a = a*a
a =
   25
```

1.4 Operar con variables.

- Definir una variable:

```
>> A = 7
A =
    7
```

- Definir una matriz:

```
>> B = [1 2 3; 4 5 6; 7 8 9]
B =
    1    2    3
    4    5    6
    7    8    9
```

- Comprobar el estado de una variable: para comprobar el valor de una variable se puede, bien mirar en la parte superior izquierda dedicada a las variables activas (Fig. 1), bien introduciendo su nombre.

```
>> B
B =
    1    2    3
    4    5    6
    7    8    9
```

- Eliminar una variable de memoria:

```
>> clear B
>> clear all  %elimina todas las variables
```

- Acceder a un elemento de una matriz: tomando la matriz B, queremos acceder al valor de la posición (1,2).

```
>> B(1,2)

ans =

    2
```

Nótese que “ans” equivale a la respuesta (del inglés “answer”).

- Acceder a todos los elementos de una fila o columna:

>> **B(:,1)** % Para obtener todos los elementos de la columna 1.

ans =
1
4
7

>> **B(1,:)** % Para obtener todos los elementos de la fila 1.

ans =
1 2 3

- Mostrar componentes consecutivos (p.ej.: del 1 al 2 de la columna 1):

>> **B(1:2,1)**

ans =
1
4

- Añadir elementos a una matriz:

>> **B(4,1) = -1**

B =
1 2 3
4 5 6
7 8 9
-1 0 0

- Producto escalar:

>> **A = [1 2;3 4]**
A =
1 2
3 4

>> **B = [4 3; 2 1]**
B =
4 3
2 1

>> **A(1,1)*B(1,1)**

ans =

4

- Producto matricial:

>> **A*B**

```
ans =
     8     5
    20    13
```

1.5 Funciones especiales.

Matlab proporciona una serie de funciones matemáticas básicas además de funciones más complejas. Como ejemplo de funciones aritméticas básicas tenemos:

- abs() % proporciona el valor absoluto de un numero.
- cos() % coseno.
- sin() % seno.
- sqrt() % cálculo de la raíz cuadrada.
- inv () % calcula la inversa de una matriz.
- …

Y como ejemplo de otras funciones tenemos:

- clock %Muestra, en un vector de seis componentes, la fecha y hora completa.
- display (‘ ‘) %Muestra el texto introducido por pantalla.

Destacar que el nombre de las funciones definidas en matlab no puede ser utilizadas como nombres de variables.

1.6 Sentencias de Control.

La sintaxis de las sentencias de control utilizadas dentro del entorno de Matlab es la siguiente:

Bucles:

- **FOR**

```
for variable = valor_inicial:valor_final
        sentencias
        …
end
```

Ejemplo:

```
>> for i=1:3
   display('hola mundo')
end

ans =
hola mundo

ans =
hola mundo
```

```
ans =
hola mundo
```

- **WHILE**

```
while variable  expresion
        sentencias
        …
end
```

Ejemplo:

```
>>  i = 1;
while i < 3
   display('hola mundo')
   i = i+1;
end

ans =
hola mundo

ans =
hola mundo
```

- **IF**

```
if  expresion
        sentencias
end
```

Ejemplo:

```
>> a
a =
    1

>> b
b =
    1

>> if (a==b)
   display('hola mundo')
    end

ans =
hola mundo
```

1.6 Algunas sentencias especiales.

1.6.1 Definición de un vector de términos crecientes o decrecientes.

Para definir un vector de términos crecientes o decrecientes se utiliza la siguiente nomenclatura:

vector = [inicio_vector:incremento:fin_vector]

Ejemplo:

```
>> t = [0:0.1:10]    % De esta forma definimos un vector t que va desde 0 hasta 10 con un
                     %  incremento  de 0.1
t =

  Columns 1 through 8

    0    0.1000    0.2000    0.3000    0.4000    0.5000    0.6000    0.7000
```

1.6.2. Representación gráfica.

Para obtener la representación gráfica de un dato normalmente se utiliza la función "plot". Esta función tiene la siguiente nomenclatura:

- plot (x,y) % dibuja el vector y (abcisas) frente al vector x (coordenadas).

Ejemplo: Si introducimos las siguientes instrucciones aparecerá la siguiente gráfica:

```
>> a = 0:0.1:4*pi
>> plot(a,sin(a))
```

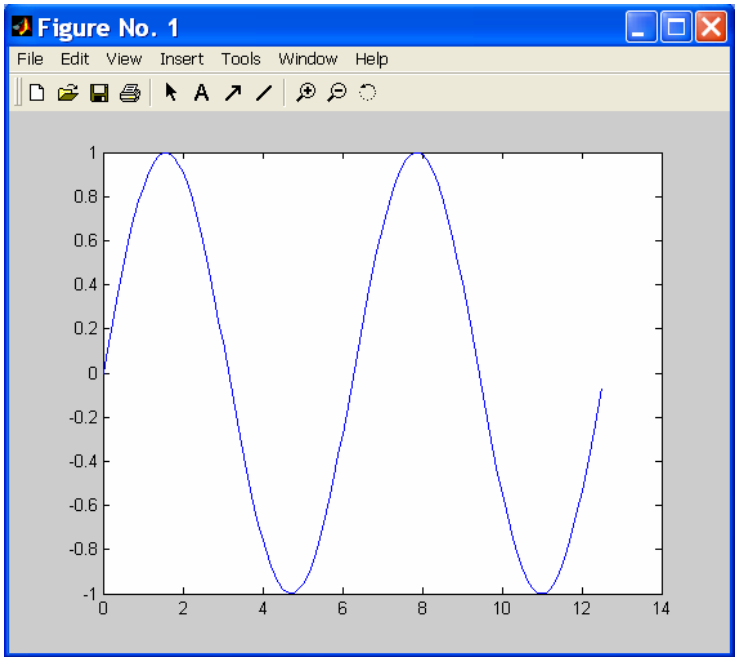


Figura 3. Representación de una señal senoidal

- plot(y) % dibuja el vector y en la abcisas mientra en las coordenadas representa el % índice del vector.

Ejemplo: Si introducimos las siguientes instrucciones aparecerá la siguiente gráfica:

```
>> a = 0:0.1:4*pi
>> plot(sin(a))
```

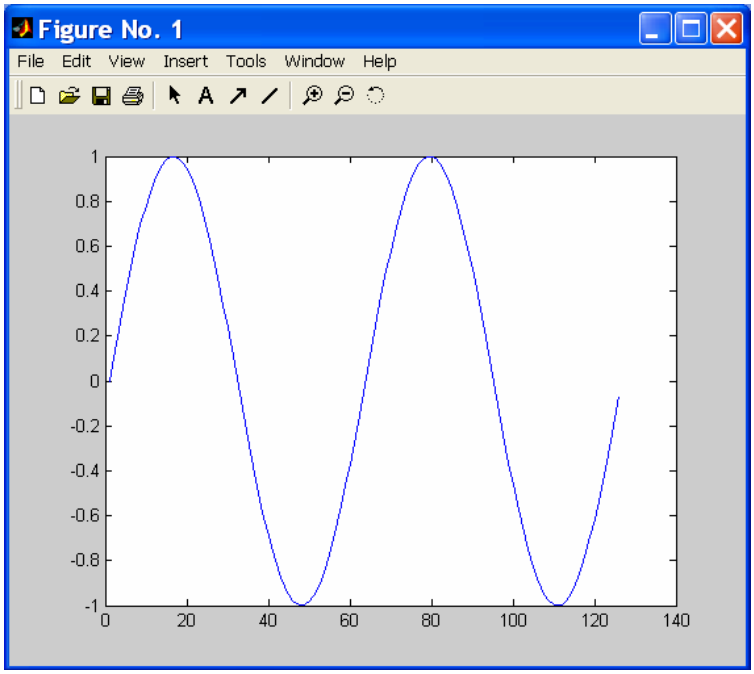


Figura 4. Representación de una señal senoidal

Nótese que en este caso la señal senoidal viene referenciada según las posiciones del vector.

Si además queremos mostrar varias señales en una misma gráfica se utilizará la función **"hold on",** que haría que a partir de entonces todos los dibujos que se realicen aparezcan en la gráfica activa. Este comando sólo funciona para la gráfica que se ha ejecutado.

Por último, para abrir una nueva gráfica basta con escribir la función **"figure".**

1.6.3. Mostrar por pantalla

Como se vio anteriormente, la función "display" puede ser utilizada para mostrar un texto por pantalla. Sin embargo, si se quiere mostrar un texto y además el valor de una variable, se puede utilizar la función "sprintf". Esta función utiliza el estándar ANSI C.

Ejemplo:

```
>> a = 1
a =
    1
>> sprintf('Valor de a: %d',a)
```

ans =
Valor de a: 1

1.7 Creación de una función a través de un fichero .m

Con Matlab también es posible crear nuestras propias funciones. Para ello se puede utilizar bien el editor de texto de Matlab (Fig. 5) o bien cualquier otro editor de texto (bloc de notas, Wordpad…).

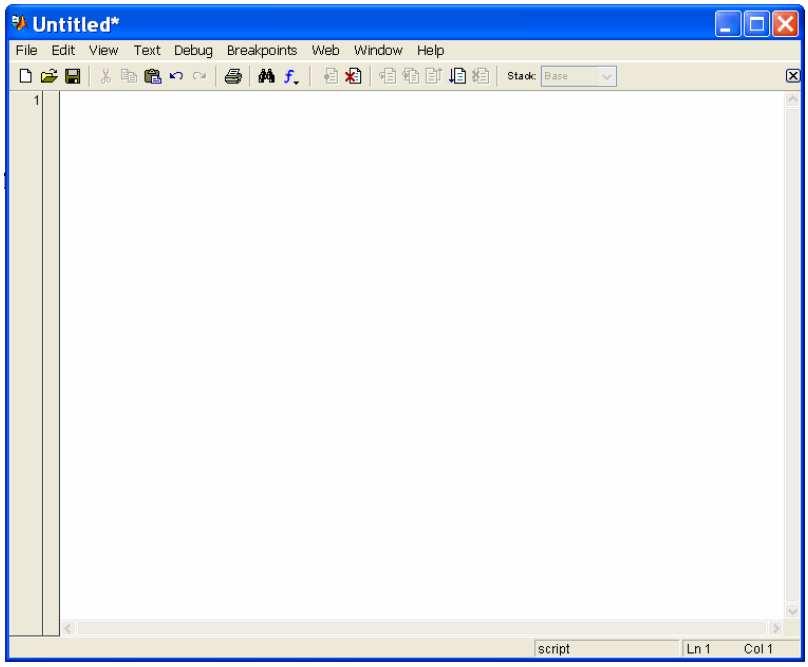


Figura 5. Interfaz del editor de texto de Matlab.

La estructura a seguir a la hora de implementar una función es la siguiente:

```
   ----------------------------------- Identificador de “function”.
   |
   |          ------------------------ Argumento de salida.
   |          |
   |          |            ----------- Nombre de la función.
   |          |            |
   |          |            |       --- Argumento(s) de entrada.
   |          |            |       |
   V          V            V       V
function [o1,o2,...] = nombre_fun(i1,i2...) ← Definición
% Aquí se escribiría la ayuda que queremos que aparezca cuando
% el usuario escriba “help nombre_fun”
% ...
% ...

   Cuerpo de la función (Aquí estaría la parte del código).
```

```
% Comentarios si los hubiera.
```

Ejemplo:

A continuación se muestra una función que calcula la inversa de una matriz.

```
function a= inversa (b)
% Funcion = Calcula la inversa de una matriz
% Parametro de entrada = b;
% Parametro de salida  = a;
%

a = inv(b);
```

Si guardamos el texto anterior en un fichero .m (con el mismo nombre de la función, es decir, inversa.m) y lo ejecutamos para la matriz f tenemos:

```
>> f

f =

    1    2
    3    4

>> inversa(f)

ans =

   -2.0000    1.0000
    1.5000   -0.5000
```

1.7 Modificación del PATH de Matlab.

Por defecto, Matlab trabaja con la carpeta "C:\matlab\work" e inicialmente todos los archivos que no se encuentren en esa carpeta u otras que ya hayan sido configuradas no son reconocidas por el programa. De ahí que se recomiende al grupo de prácticas que se cree una carpeta dentro de "C:\matlab\work", que la active para que la reconozca Matlab, y que a partir de entonces todos los archivos sobre los que trabaje se encuentre dentro de esta carpeta.

El procedimiento para activar una carpeta en matlab sería el siguiente:

1. Crear una carpeta dentro de la carpeta C:\matlab\work.
2. Dentro de matlab, elegir la opción "Set Path…" (Fig. 6).
3. Aparecerá una ventana de configuración del path (Fig. 7), elegir la carpeta correspondiente y salvarla.
4. En caso de que pregunte si queremos guardar la configuración para sesiones futuras, decirle que si.

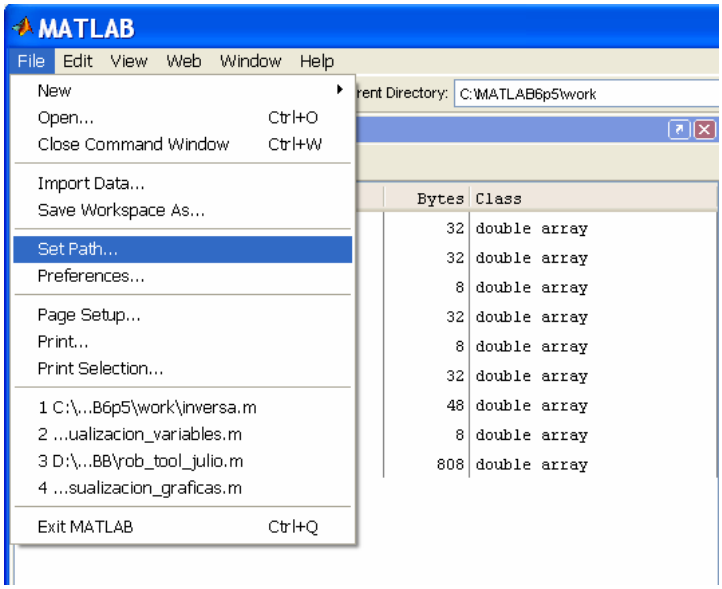


Figura 6. Opción de configuración de path.

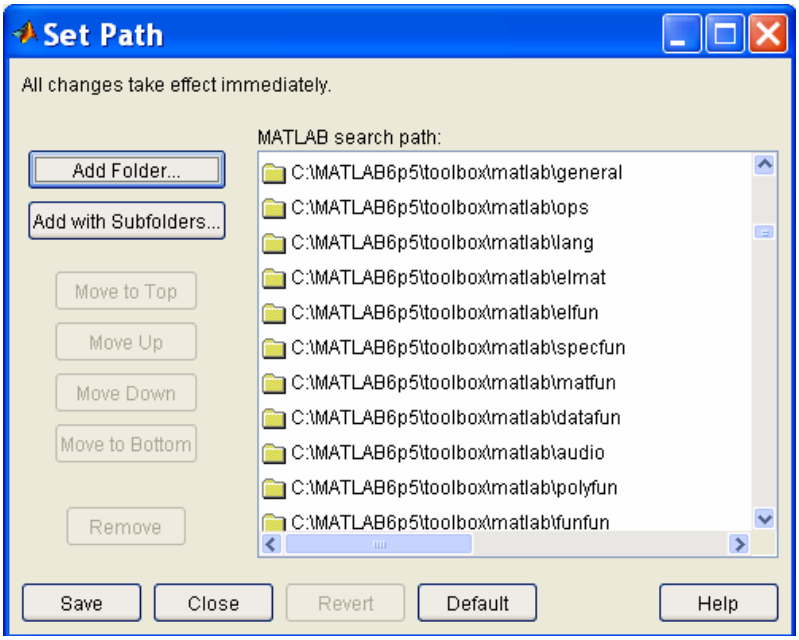


Figura 7. Ventana de configuración del path.

2. Toolbox de Image Processing.

2.1 Leer y mostrar una imagen.

El comando utilizado por Matlab para leer una imagen es **imread**.

>> I=imread('*.tif');

Esta instrucción lee una imagen (en este caso en formato tiff) y la almacena en una matriz llamada I.

Para mostrar la imagen se utiliza la instrucción **imshow(I).**

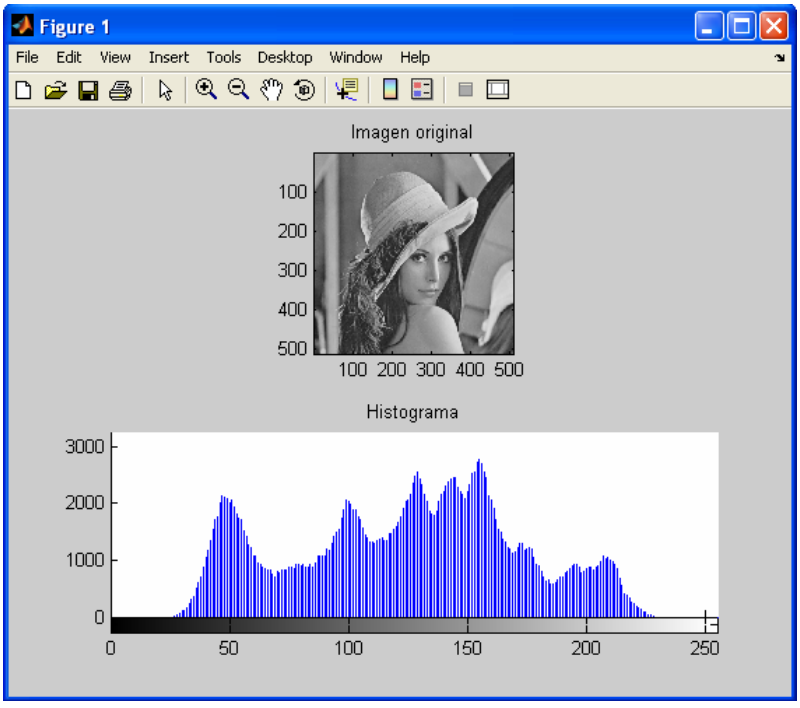
Si queremos comprobar el valor de la variable I (almacenada en memoria) bastará con teclear desde la ventana de comandos **whos.**

2.2 Histograma de una imagen.

Con un histograma es posible ver la distribución de intensidades en una imagen basta con utilizar la función **imhist.** Si nos interesa ecualizar el histograma de una imagen (para extender los valores de intensidad) bastaría con utilizar la función **histeq.**

El código matlab para visualizar en una misma pantalla una imagen y su histograma podría ser:

```
>>clear all;
>>close all;
>>LN=imread('C:\MATLAB7\work\imagenes\lena.jpg');
>>LN1=rgb2gray(LN); %pasar la imagen a escala de grises.
>>subplot(2,1,1),subimage(LN1),title('Imagen original');
>>subplot(2,1,2),imhist(LN1),title('Histograma');
```



2 Ejercicio Propuesto.

Nota: Para la realización de los ejercicios es necesario crear una carpeta de trabajo dentro de C:\matlab\work y activarla por parte de matlab. Las imágenes a utilizar se encuentran en el siguiente path: C:\matlab\work\imagenes.

2.1 Ejercicio 1.

Mostrar en una misma pantalla los siguientes gráficos e imágenes:

1. Imagen pout.tiff.
2. Histograma de la imagen.
3. Ecualización de la imagen.
4. Histograma de la imagen ecualizada.