Señales y Sistemas – Otoño 2005
Prof. Verónica Vilaplana

Tutorial de MATLAB

1. Introducción: qué es MATLAB?

Matlab es un lenguaje de alto nivel para la resolución de cálculos numéricos en aplicaciones científicas y de ingeniería. Integra el cálculo, la visualización y la programación en un entorno fácil de usar.

Matlab es un sistema interactivo, cuyo elemento de datos básico es un arreglo o matriz, que no requiere ser dimensionado. Esto hace que se puedan resolver problemas técnicos de cálculo, especialmente los formulados en términos de matrices y vectores, en mucho menos tiempo que si se utilizara un lenguaje de programación no interactivo como C o Fortran.

El nombre Matlab viene de MATrix LABoratory.

2. Primeros pasos

Cómo entrar:
Hacer doble click en el ícono de Matlab

Cómo es el entorno de trabajo:
Por defecto, la configuración muestra en la pantalla tres zonas de trabajo, y una barra de menú en la parte superior. La mitad derecha de la pantalla muestra la ventana de comandos ("*command window*"), y es donde se escriben los comandos (en la línea indicada con ">>") y se ven los resultados de las operaciones. La mitad izquierda se divide en dos ventanas, la superior permite consultar la documentación, herramientas y demostraciones instaladas ("l*aunch pad*"), y las variables definidas hasta el momento ("*workspace*"), y la inferior muestra los comandos ejecutados hasta el momento ("*command history*")
Además, hay un editor de textos que permite editar ficheros con funciones o instrucciones que se ejecutan al ser invocados desde Matlab (ficheros con extensión .m, o "M-files"), y un *debugger*.

Cómo salir:
Escribir 'exit' o 'quit' en la línea de comandos.

```
>> exit
```

Caja de herramientas o "Toolbox"
Además de los comandos básicos de Matlab, también existen las "*toolboxes*" o cajas de herramientas, que son conjuntos de programas o funciones con propósitos específicos, que agregan funcionalidades específicas a las brindadas por el núcleo de Matlab. Para consultar qué *toolboxes* se encuentran instaladas en el sistema, se puede escribir

```
>> help
```

en la ventana de comandos, y aparecerá una lista de todas las cajas de herramientas disponibles, por ejemplo:

```
matlab\general       -  General purpose commands.
matlab\ops           -  Operators and special characters.
matlab\lang          -  Programming language constructs.
matlab\elmat         -  Elementary matrices and matrix manipulation.
matlab\elfun         -  Elementary math functions.
matlab\specfun       -  Specialized math functions.
matlab\matfun        -  Matrix functions - numerical linear algebra.
matlab\datafun       -  Data analysis and Fourier transforms.
matlab\polyfun       -  Interpolation and polynomials.
matlab\graph2d       -  Two dimensional graphs.
matlab\graph3d       -  Three dimensional graphs.
matlab\specgraph     -  Specialized graphs.
matlab\graphics      -  Handle Graphics.
matlab\uitools       -  Graphical user interface tools.
matlab\strfun        -  Character strings.
matlab\iofun         -  File input/output.
matlab\timefun       -  Time and dates.
matlab\datatypes     -  Data types and structures.
images\images        -  Image Processing Toolbox.
toolbox\optim        -  Optimization Toolbox.
signal\signal        -  Signal Processing Toolbox.
toolbox\splines      -  Spline Toolbox.
toolbox\stats        -  Statistics Toolbox.
```

Escribiendo
`>> help [nombre_herramienta]`
aparece una lista con los comandos que ofrece el toolbox.

Ayuda

En caso de tener problemas al utilizar algún comando de Matlab, el primer recurso a utilizar es la ayuda que brinda el mismo programa.

`>> help [nombre_herramienta]`
aparece una lista con los comandos que ofrece la caja de herramientas mencionada.

`>> help [nombre_comando]`
aparece la explicación sobre el funcionamiento del comando.

Comandos de propósito general

Los siguientes son algunos comandos básicos y muy útiles, de propósito general:
`>> help general`

```
  General purpose commands.
  MATLAB Toolbox   Version 6.0 (R12)  06-Oct-2000

  General information
    help        - On-line help, display text at command line.
    demo        - Run demonstrations.
    java        - Use of Java from MATLAB.

  Managing the workspace.
    who         - List current variables.
    clear       - Clear variables and functions from memory.
    load        - Load workspace variables from disk.
    save        - Save workspace variables to disk.
    quit        - Quit MATLAB session.
```

```
  Managing commands and functions.
    edit        - Edit M-file.
    open        - Open files by extension.

  Managing the search path
    path        - Get/set search path.

  Operating system commands
    cd          - Change current working directory.
    copyfile    - Copy a file.
    pwd         - Show (print) current working directory.
    dir         - List directory.
    delete      - Delete file.
    getenv      - Get environment variable.
    mkdir       - Make directory.
    !           - Execute operating system command (see PUNCT).
    dos         - Execute DOS command and return result.
```

Escribiendo:
`>> help [nombre_comando]`
aparece la explicación sobre el funcionamiento del comando.

Ejemplos:

```
>> help cd

 CD     Change current working directory.
    CD directory-spec sets the current directory to the one specified.
    CD .. moves to the directory above the current one.
    CD, by itself, prints out the current directory.


>> help clear

 CLEAR  Clear variables and functions from memory.
    CLEAR removes all variables from the workspace.

    CLEAR ALL removes all variables, globals, functions and MEX links.

    CLEAR VAR1 VAR2 ... clears the variables specified.  The wildcard
    character '*' can be used to clear variables that match a pattern.
    For instance, CLEAR X* clears all the variables in the current
    workspace that start with X.
```

3. Matrices

El elemento básico de trabajo en Matlab es la matriz. Una matriz de dimensión nxm es un arreglo rectangular de números (n filas y m columnas). Los vectores y los escalares son casos particulares de matrices. Un vector fila es una matriz de 1xn, un vector columna es una matriz de nx1, y un escalar es una matriz de 1x1.

Cómo se genera una matriz:
(a) Escribiendo la lista de sus elementos
(b) Utilizando funciones predefinidas
(c) Generándolas a partir de funciones definidas por el usuario
(d) Cargándolas desde ficheros externos de datos.

(a) Escribiendo la lista de sus elementos

```
>> A=[ 1 5 8 ; 4 3 20 ; 0 0 1 ]

A =

     1     5     8
     4     3    20
     0     0     1
```

Se escriben los elementos que forman la matriz entre corchetes [ ], separando con espacios los elementos de cada fila, y separando las filas con ";".
De esta manera, se crea y dimensiona una variable A como una matriz de tres filas y tres columnas con los valores especificados.
Escribiendo

```
>> who

Your variables are:

A
```

Aparece la lista de variables en memoria, por el momento solo la variable A.

(b) Utilizando funciones predefinidas

Existen diferentes funciones que permiten crear ciertas matrices, por ejemplo, los comandos "zeros", "ones", etc.

```
>> B = ones(5)
```

Genera una matriz de unos, de dimensión 5x5.

```
>> C = ones(2,3)
```

Genera una matriz de dimensión 2x3, de unos.

```
>> C = zeros(4,1)
```

Genera una matriz de dimensión 4x1, de ceros (un vector columna de 4 elementos iguales a 0).

(c) Generándolas a partir de funciones definidas por el usuario

(d) Cargándolas desde ficheros externos de datos.

Observaciones:

- Para las variables de 1 dimensión (valores constantes) no es necesario usar [ ]

```
>> E=8
E =
     8
```

- Si se escribe el nombre de alguna de las variables ya definidas y la tecla <enter>, Matlab devuelve el valor de la variable.

```
>> E

E =
     8
```

- Para conocer las dimensiones de una variable, se utiliza el comando `size`

```
>> size(A)

ans =
```

```
     3     3
```

- Cuando no se especifica la variable de salida, Matlab usa la variable `ans` (de answer) para almacenar el resultado el cálculo o el valor devuelto por la función utilizada. En el ejemplo previo, `ans` contiene el tamaño de la matriz A.

- Para suprimir la visualización del resultado se puede escribir ';' al final de la expresión.

```
>> zeros(100)
```

```
>> zeros(100);
```

Subídices:

El elemento fila i, columna j de la matriz A se referencia como A(i,j)

```
>> A(2,1)

ans =

     4
```

Una referencia a un elemento fuera de la matriz, es un error

```
>> A(4,5)
???  Index exceeds matrix dimensions.
```

Pero si lo que se intenta hacer es almacenar un valor en un elemento fuera de la matriz, Matlab aumenta el tamaño de la matriz para acomodarlo. Agrega la cantidad de filas y/o columnas necesarias y las inicializa con el valor 0.

```
>> A(4,1)=7

A =
     1     5     8
     4     3    20
     0     0     1
     7     0     0

>> A(2,:)

ans =

     4     3    20
```

El operador ':'

El operador : se utiliza para hacer referencia a un rango ordenado de elementos.

Ejemplos:

Un vector fila que contiene los números enteros entre 1 y 10

```
>> 1:10
ans =
     1     2     3     4     5     6     7     8     9    10
```

Si queremos guardar estos números en una matriz (en un vector fila):

```
>> a=1:10
a =
     1     2     3     4     5     6     7     8     9    10

>> size(a)
ans =
     1    10
```

Es posible definir incrementos distintos de 1, positivos o negativos:
Un vector con números enteros entre el 100 y el 50, en incrementos de -7:

```
>> 100:-7:50
ans =
   100    93    86    79    72    65    58    51
```

Un vector con números pares entre 0 y 10:

```
>> 0:2:10
ans =
     0     2     4     6     8    10
```

El operador : también se utiliza para indicar rangos o partes de una matriz
A(1:k,j) se refiere a los primeros k elementos de la columna j de la matriz A

Por ejemplo A(1:3,1) se refiere a los primeros 3 elementos (filas 1, 2 y 3) de la columna 1 de A:

```
>> A(1:3,1)
ans =
     1
     4
     0
```

El operador : solo, se refiere a todos los elementos de una fila o columna. Por ejemplo A(:,j) devuelve todos los elementos de la columna j. A(i,:) devuelve todos los elementos de la fila i.

```
>> A(:,1)
ans =
     1
     4
     0
     7
```

Concatenación de matrices
La concatenación de matrices se hace con el operador [ ]

Ejemplos:
Una matriz al lado de la otra

```
>> B=[A , A]

B =

     1     5     8     1     5     8
     4     3    20     4     3    20
     0     0     1     0     0     1
     7     0     0     7     0     0
```

Una matriz debajo de la otra

```
>> C=[A;A]

C =

     1     5     8
     4     3    20
     0     0     1
     7     0     0
     1     5     8
     4     3    20
     0     0     1
```

```
     7     0     0
```

4. Expresiones

Variables

A diferencia de otros lenguajes de programación de alto nivel como C o Java, Matlab no requiere declaraciones de tipo de variable ni de dimensión. Cuando el intérprete de comandos encuentra un nuevo nombre de variable, la crea y reserva espacio en memoria.

Los nombres de variables son conjuntos de caracteres: letras, números y el carácter '_'. El primer carácter de un nombre de variable siempre debe ser una letra. Se distinguen mayúsculas de minúsculas.

Números

Ejemplos

```
3
-99
9.6397
0.0001
1.6021e-20
6.0225e23
1.2i  ó     1.2j  ó     0+1.2i
4+3j  ó     4+3i
```

Observación: se usan indistintamente `i` o `j` para indicar la unidad imaginaria.

Internamente, el formato utilizado por Matlab es *long*, con precisión de 16 dígitos decimales., y trabaja en el rango $[10^{-308}, 10^{308}]$.

Si embargo, si bien trabaja con precisión de 16 dígitos, sólo muestra 4 dígitos en pantalla. Para controlar el formato con que se ven los números se utiliza el comando *format*

```
FORMAT Set output format.
    All computations in MATLAB are done in double precision.
    FORMAT may be used to switch between different output
    display formats as follows:
      FORMAT         Default. Same as SHORT.
      FORMAT SHORT   Scaled fixed point format with 5 digits.
      FORMAT LONG    Scaled fixed point format with 15 digits.
      FORMAT SHORT E Floating point format with 5 digits.
      FORMAT LONG E  Floating point format with 15 digits.
      FORMAT SHORT G Best of fixed or floating point format with 5
digits.
      FORMAT LONG G  Best of fixed or floating point format with 15
digits.
      FORMAT HEX     Hexadecimal format.
      FORMAT +       The symbols +, - and blank are printed
                     for positive, negative and zero elements.
                     Imaginary parts are ignored.
      FORMAT BANK    Fixed format for dollars and cents.
      FORMAT RAT     Approximation by ratio of small integers.
```

Operadores

Los más comunes:

```
      +     suma
      -     resta
      *     multiplicación (de matrices!)
```

```
/     división
^     potencia
'     conjugar y transponer
( )   paréntesis para indicar un orden de evaluación particular
.*    producto escalar de dos vectores
.'    transponer
```

Ejemplos:

```
>> a= 3+5i;
a =

   3.0000 + 5.0000i


>> a'

ans =

   3.0000 - 5.0000i


>> F=[3+5j 2-4j]
F =
   3.0000 + 5.0000i   2.0000 - 4.0000i
```

Obs: F es un vector fila de 2 elementos (una matriz de 1x2)

```
>> F'
ans =
   3.0000 - 5.0000i
   2.0000 + 4.0000i

>> 3*A
ans =
     3    15    24
    12     9    60
     0     0     3
    21     0     0

>> -a
ans =
  -3.0000 - 5.0000i
```

Observación:

* es producto de matrices: hay que tener cuidado con las dimensiones. Para que el producto de dos matrices A y B esté bien definido, debe ocurrir que $A \in \mathbb{R}^{N\times M}$, $B \in \mathbb{R}^{M\times K}$, y entonces el producto es $A*B \in \mathbb{R}^{N\times K}$. Si el número de columnas de A no coincide con el número de filas de B, Matlab devolverá un mensaje de error.

Otros operadores (solo algunos, ver 'help ops'):

```
  Operators and special characters.

  Arithmetic operators.
    plus       - Plus                                +
    uplus      - Unary plus                          +
    minus      - Minus                               -
    uminus     - Unary minus                         -
    mtimes     - Matrix multiply                     *
    times      - Array multiply                     .*
    mpower     - Matrix power                        ^
```

```
    power      - Array power                        .^
    mrdivide   - Slash or right matrix divide       /
    rdivide    - Right array divide                 ./

  Relational operators.
    eq         - Equal                              ==
    ne         - Not equal                          ~=
    lt         - Less than                          <
    gt         - Greater than                       >
    le         - Less than or equal                 <=
    ge         - Greater than or equal              >=

  Logical operators.
    and        - Logical AND                        &
    or         - Logical OR                         |
    not        - Logical NOT                        ~
    xor        - Logical EXCLUSIVE OR

  Special characters.
    colon      - Colon                              :
    paren      - Parentheses and subscripting       ( )
    paren      - Brackets                           [ ]
    paren      - Braces and subscripting            { }
    punct      - Decimal point                      .
    punct      - Assignment                         =
    transpose  - Transpose                          .'
    ctranspose - Complex conjugate transpose        '
```

Funciones

Matlab provee gran cantidad de funciones matemáticas estándards. Es posible ver un listado de estas funciones, agrupadas por categorías:

```
>> help elfun     Elementary math functions.
>> help specfun  Specialized math functions.
>> help elmat  Elementary matrices and matrix manipulation.
```

Ejemplos (algunas funciones de 'elfun'):

```
>> help elfun     Elementary math functions.
  Trigonometric.
    sin          - Sine.
    asin         - Inverse sine.
    cos          - Cosine.
    acos         - Inverse cosine.
    tan          - Tangent.
    atan         - Inverse tangent.

  Exponential.
    exp          - Exponential.
    log          - Natural logarithm.
    log10        - Common (base 10) logarithm.
    log2         - Base 2 logarithm and dissect floating point number.
    pow2         - Base 2 power and scale floating point number.
    sqrt         - Square root.

  Complex.
    abs          - Absolute value.
    angle        - Phase angle.
    complex      - Construct complex data from real and imaginary
parts.
    conj         - Complex conjugate.
    imag         - Complex imaginary part.
    real         - Complex real part.
```

```
    isreal      - True for real array.
```

Constantes

Algunos nombres de variables están asignados por defecto a ciertas constantes. Estos nombres no se pueden utilizar como nombres de variables definidas por un usuario.

Ejemplos:

```
>> pi
ans =
    3.1416

>> i
ans =
        0 + 1.0000i

>> j
ans =
        0 + 1.0000i
```

El menor número real representable con MATLAB se llama `realmin` y su valor es:

```
>> realmin
ans =
  2.2251e-308
```

El mayor número real representable con MATLAB se llama `realmax` y su valor es:

```
>> realmax
ans =
  1.7977e+308
```

Infinito:

```
>> Inf
```

Ejemplo:

```
>> a = 0;
>> b=5/a
Warning: Divide by zero.
b =
   Inf
```

Not_a_number: para operaciones con resultado numérico indefinido

```
>> NaN

Ejemplo:
>> b=0;
>> c = a/b
c =
   NaN
```

Observaciones:

- Las funciones, constantes, operadores se combinan para formar expresiones complejas:
  `z=sqrt(exp(log(pi*3.5)));`
- Si una expresión no cabe en una línea se puede separar en dos líneas, colocando al final de la primera el símbolo '...' que indica continuación.
- *Comentarios:* Se pueden escribir comentarios comenzando la línea con el símbolo '%'.

5. Gráficos

Una característica importante de Matlab es la facilidad con la que permite analizar datos de manera gráfica. Matlab tiene la capacidad de mostrar vectores y matrices como gráficos 2D o 3D, permite agregar información a estos gráficos e imprimirlos o guardarlos como ficheros.

Plot
Es la función que se utiliza para crear un gráfico lineal, trabaja de diferentes formas según sean los argumentos de entrada.

Ejemplos:
Si el argumento de entrada es un vector, genera un gráfico lineal por partes, mostrando los elementos del vector en función de su índice.

```
>> y=[2 3 6 1 7 8];
>> plot(y)
```

Si el argumento son dos vectores plot(x,y), genera un gráfico del vector y con respecto al vector x.

Por ejemplo, para hacer un gráfico de una parábola para valores enteros de x entre -10 y 10, creamos un vector x, calculamos x^2 y dibujamos el resultado

```
>>  x = -10:10;
>>  y = x.*x;
>>  plot(x,y);
```

Si sólo queremos graficar la paraboloa en los puntos en que la hemos calculado, usamos el comando `stem`:

```
>>  stem(x,y);
```

Otro ejemplo: para hacer un gráfico de la función sin(x) para ciertos valores de x: creamos un vector x de valores entre 0 y 2π, calculamos el seno de estos valores y graficamos el resultado.

```
>> x=0:pi/100:2*pi;  %nros reales entre 0 y 2pi en increm. de pi/100
>> y=sin(x);
>> plot(x,y)
```

Es posible agregar títulos al gráfico y nombre a los ejes:

```
>> title('Grafico de la funcion seno');
>> xlabel('x=0:2\pi');
>> ylabel('y=sin(x)');
```

Es posible crear varios gráficos juntos, en la misma llamada a la función plot.
Siguiendo con el ejemplo anterior, para dibujar diferentes senos desplazados:

```
>> y2=sin(x-0.25);
>> y3=sin(x-0.5);
>> plot(x,y,x,y2,x,y3);
```

El comando legend permite identificar los diferentes gráficos
```
>> legend('sin(x)','sin(x-o.25)','sin(x-0.5)');
```

Escribiendo 'help plot', es posible ver como crear gráficos con diferentes estilos de líneas y marcadores (círculos, cruces, etc.), diferentes colores, etc.
Ejemplo:
Para crear una línea de puntos de color rojo
```
>> plot(x,y,'r:')
```

Para marcar el signo '+' donde van los datos, y '-' en el resto de la línea:
```
>> z=[1 3 5 7 8 4];
>> plot(z,'b--+')
```

Figure
Los gráficos del *plot* se generan sobre ventanas especiales, llamadas *'figure windows'*. La primera vez que se invoca a la función *plot* se crea una de estas ventanas automáticamente; las siguientes llamas a la función *plot* realizan los gráficos sobre esta misma ventana. Para crear una nueva ventana se utiliza el comando *'figure'*. La última ventana utilizada es la que queda como ventana activa, y es sobre la que se generara el próximo grafico con la siguiente llamada a la función plot (o a otro comando que genere gráficos).
Para hacer que cierta *figure window* sea la ventana activa, se puede hacer click con el mouse mientras el puntero está sobre esta ventana, o bien se puede escribir
```
>> figure(n)
```
donde n es el número que aparece en la barra de título de la ventana. Los siguientes gráficos se dibujarán sobre esta ventana.

Axis
Este comando permite elegir una escala, rotación y relación de aspecto para los gráficos. Por ejemplo, una llamada con la siguiente forma:
```
>> axis([xmin xmax ymin ymax])
```
permite elegir los valores mínimo y máximo de cada uno de los ejes. Si no se utiliza esto, Matlab elige como límites de los ejes los valores máximos y mínimos de la función a dibujar.

Grid
El comando grid permite mostrar y ocultar una grilla sobre la gráfica
```
>> grid on

>> grid off
```

Subplot
El comando subplot permite mostrar múltiples plots en la misma ventana.
Escribir
```
>> subplot(m,n,p)
```
parte la figure window en una matriz de m por n partes, y selecciona la p-ésima parte para dibujar el próximo gráfico.
Las partes se numeran consecutivamente por filas, comenzando por la fila superior de la figure window, luego la segunda fila, etc.

Por ejemplo, para generar cuatro gráficos en la misma ventana

```
>> x = 0:pi/100:2*pi;
>> y = sin(x);
>> z = cos(x);
>> w = exp(-x*0.1).*y;
>> v= y.*z;
>>subplot(2,2,1)
>> plot(x,y)
>>subplot(2,2,2)
>> plot(x,z)
>>subplot(2,2,3)
>> plot(x,w)
>>subplot(2,2,4)
>> plot(x,v)
```

Guardar figuras en ficheros

Es posible guardar una figura en un fichero, en formato *fig* de Matlab, o exportarlo a otros formatos gráficos (*bmp, eps, jpg, tif*).

Para salvar la figura, seleccionar *Save* en el menú *File*. Para guardarla en otro formato, seleccionar *Export* en el menú *File*.

Para guardarla desde la línea de comandos se utiliza el comando *saveas*, incluyendo las opciones necesarias para guardar la figura en el formato deseado (ver 'help saveas').

6. Programando con MATLAB

Matlab tiene varias construcciones para el control de flujo:

if

IF expresión_lógica_1
  (comandos)
 ELSEIF expresión_lógica_2 *(opcional)*
  (comandos)
 ELSE *(opcional)*
  (comandos)
END

Recordemos cuáles son los operadores lógicos y relacionales que permiten construir expresiones lógicas:

```
  Relational operators.
    eq         - Equal                          ==
    ne         - Not equal                      ~=
    lt         - Less than                       <
    gt         - Greater than                    >
    le         - Less than or equal             <=
    ge         - Greater than or equal          >=

  Logical operators.
    and        - Logical AND                     &
    or         - Logical OR                      |
    not        - Logical NOT                     ~
    xor        - Logical EXCLUSIVE OR
```

for

FOR variable = expresión
  (comandos)

END

Ejemplo:
```
>> for n=1:10
>>    H(n) = 3*n+2;
>> end
>> H
```

switch
SWITCH switch_expresion
  CASE case_expr
        (comandos)
  CASE {case_expr1, case_expr2, case_expr3,...}
        (comandos)
  OTHERWISE,
        (comandos)
END

while
WHILE condición_lógica
        (comandos)
END

7. Ficheros .m

Matlab es, además de un entorno de cálculo y visualización interactivo, un lenguaje de programación. Los ficheros que contienen código en lenguaje Matlab se llaman '*m-files*', son ficheros con extensión '.m'.
Es posible crear un fichero .m usando un editor de texto, y luego utilizarlo como se utiliza cualquier otra función o comando Matlab.

Hay dos tipos de ficheros .m: *scripts* y *funciones*.
Los *scripts* son ficheros que contienen una lista de comandos. No aceptan argumentos de entrada ni devuelven argumentos de salida. Operan sobre el espacio de variables de la sesión.
Las *funciones* pueden aceptar argumentos de entrada y devolver argumentos de salida. Operan sobre un espacio de variables propio. Las variables internas utilizadas dentro de la función son locales a la función.

Scripts
Cuando se invoca un script, Matlab simplemente ejecuta los comandos que aparecen en el fichero. Los scripts pueden operar sobre las variables existentes en el espacio de trabajo, o pueden crear otras nuevas. Todas las variables que se generan durante la ejecución del script permanecen en el espacio de variables de la sesión y pueden ser utilizadas posteriormente. Los scripts pueden generar funciones de salida gráfica usando funciones como el plot.

Funciones
Las funciones son ficheros .m que pueden aceptar argumentos de entrada y devolver argumentos de salida. *El nombre del fichero y el de la función deben coincidir*. Las funciones operan sobre un espacio de variables propio, separado del espacio de la sesión (al que se accede desde la ventana de comandos de Matlab).
La primera línea de un fichero .m de función comienza con la palabra *function*, seguida del nombre de la función y la lista de argumentos.

Ejemplo: en un fichero llamado 'modulo.m'

```
function r=modulo(a,d)
% aqui van los comentarios de ayuda
% r=modulo(a,d) devuelve el resto entero
% de la división de a por d

r = a-d.*floor(a./d);
```

En este ejemplo, la función tiene dos argumentos de entrada y uno de salida.
Las siguientes líneas, hasta la primera línea en blanco o la primera línea ejecutable, son líneas de comentarios que proveen un texto de ayuda. Estas líneas se imprimen al escribir 'help modulo'
El resto del fichero contiene el código que define la función.

Es muy aconsejable comentar el código que se escribe en los ficheros .m para poder entender su funcionamiento al cabo de un tiempo. Los comentarios se insertan comenzando las líneas con el símbolo '%'.

La función se ejecuta escribiendo el nombre de fichero en la línea de comandos.
En el ejemplo

```
>> res = modulo(10,3)

res =

     1
```

8. Ficheros .mat

El *workspace* actual, es decir, las variables definidas durante la sesión actual, no se mantienen entre sesiones sino que se borran al salir de Matlab. Es posible salvar todas o algunas de las variables del workspace a un fichero '*MAT-file*', que es un fichero binario específico de Matlab con extensión '.mat'. Luego, en la misma sesión o en otra, es posible cargar el MAT-file para volver a usar las variables.
Para salvar todas las variables del workspace, desde el menú *File*, seleccionar *Save Workspace As*; se abre entonces una ventana que permite especificar el nombre y la ubicación del fichero .mat. También es posible salvarlas desde la ventana de comandos con el comando *save.*
Para salvar solo algunas de las variables, se utiliza el comando *save*, especificando además del nombre del fichero, los nombres de las variables a salvar. (ver '*help save*')
Para cargar un fichero .mat y recuperar las variables guardadas en él, seleccionar el menú *File* y luego *Import Data* (se abre una ventana de diálogo), o bien utilizar el comando *load*.