Introducción a MATLAB®

MATLAB® es un sistema interactivo y un lenguaje de programación que utiliza matrices que no requieren dimensionamiento como elementos de datos básicos. Esto permite la solución de muchos problemas numéricos en un tiempo menor al requerido por otros lenguajes de programación. Al igual que en estos lenguajes, se pueden construir herramientas propias en código MATLAB® (archivos m) que pueden ser utilizados en la resolución de otros problemas.

En los siguientes apartados se muestra un pequeño resumen de las funciones que se utilizarán con más frecuencia en estas prácticas. No se pretende desarrollar un completo manual de utilización del programa. Tan sólo se trata de una referencia rápida que debería completarse con la utilización de la ayuda online del propio programa. De hecho, puede que en el desarrollo de alguna práctica sea necesaria la utilización de alguna función no recogida en este resumen.

Las órdenes y funciones de MATLAB® se muestran agrupadas por temas dependiendo de la tarea a realizar en cada momento con objeto de facilitar su consulta. Algunas órdenes pueden incluir también ejemplos sencillos de utilización. En general, a lo largo de esta memoria, las sentencias y órdenes propias de MATLAB® se escribirán con `este tipo de letra`.

1. Órdenes de propósito general

- `help` — Ayuda en línea. El tema puede ser un nombre de función propia de MATLAB® pero se pueden crear esta ayuda en línea para los archivos propios del usuario. En cualquier momento se puede interrogar al programa sobre el funcionamiento de una determinada función o comando.

- `lookfor` — Búsqueda de caracteres en todas las entradas de ayuda.
`lookfor Fourier` $\Rightarrow$ busca el string "Fourier" en la primera línea de comentario del todos los ficheros m y proporciona como salida el contenido de esta línea cuando se ha encontrado este string.

- `quit` — Termina la sesión de MATLAB® sin almacenar el espacio de trabajo. Para almacenar las variables utilizadas durante la sesión se debe usar el comando `save` antes de finalizar el trabajo. Las variables se almacenan en un archivo binario definido por el usuario. Con el comando `load` se recuperan de nuevo esas variables.

2. Manejo de archivos y espacio de trabajo

- `clear` — Elimina elementos de la memoria. Sin argumentos borra todas las variables del espacio de trabajo. El argumento puede ser el nombre de una función o una variable. La orden `clear all` borra todas las variables y funciones dejando el espacio de trabajo como si la sesión de MATLAB® comenzase en este mismo momento.

- `who` / `whos` — Lista el directorio de variables en memoria. `whos` lista las variables actuales, su tamaño y si tienen parte imaginaria distinta de cero.

- `cd` Cambia el directorio de trabajo por defecto.
- `delete` Borra archivos y objetos gráficos.
- `dir` / `ls` Listado de un directorio. Sin argumentos produce el listado del directorio actual.
- `pwd` Muestra el nombre del directorio de trabajo actual.
- `unix` / `!` Ejecuta un comando UNIX/DOS desde MATLAB®.
- `format` Controla el formato de salida en la visualización. Utilizar `help format` para una descripción más detallada.

3. Operadores y caracteres especiales

- `+ - * / ^` Suma, resta, multiplicación de matrices, división de matrices, potencia de matrices. Las dimensiones de las matrices deben ser las adecuadas en cada caso.
  `Z=A*B` ⇒ si `A` y `B` son matrices `n×n`, proporciona una nueva matriz `Z` `n×n` cuyos elementos se obtienen mediante producto matricial, por ejemplo:
  $z_{11}=a_{11}*b_{11}+a_{12}*b_{21}+a_{13}*b_{31}+...+a_{1n}*b_{n1}$.
- `.* ./ .^` Multiplicación, división y potencia de arrays elemento a elemento.
  `Z=A.*B` ⇒ si `A` y `B` son matrices `n×n`, proporciona una nueva matriz `Z` `n×n` cuyos elementos se obtienen mediante producto elemento a elemento, por ejemplo:
  $z_{11}=a_{11}*b_{11}$.
  $z_{12}=a_{12}*b_{12}$.
- `;` Se usa después de una expresión para suprimir su impresión en pantalla o para separar órdenes. Dentro de los corchetes indica el final de una línea.
  `x=sin(5.0*pi*t);` ⇒ evalúa la expresión sin imprimir en pantalla el resultado.
- `%` Comentarios. Se ignora cualquier texto que se encuentre a continuación.
- `'` (comilla) Matriz traspuesta complejo conjugada.
- `.'` Matriz traspuesta pero no conjugada.
- `== < >` Operadores relacionales para comparaciones entre dos matrices elemento a elemento.
- `& | ~` Operadores lógicos AND, OR y NOT.

4. Control de flujo

- `if` Ejecución condicionada de sentencias dependiendo del valor que tome una determinada expresión.

```
if x=3
   ⋮
  conjunto de sentencias
   ⋮
end
```

  En este ejemplo, el conjunto de sentencias solo se ejecutan si el valor de `x` es igual a 3. En caso contrario se continúa con el código que se encuentre a continuación del `end`.

- `else/elseif` Añade nuevas condiciones para la expresión.
```
if x=3
  ⋮
  conjunto de sentencias 1
  ⋮
else
  ⋮
  conjunto de sentencias 2
  ⋮
end
```
En este ejemplo, el primer conjunto de sentencias solo se ejecutan si el valor de `x` es igual a 3. En caso contrario, `x` distinto de 3, se ejecuta el segundo grupo de sentencias. A continuación se ejecuta el código escrito a continuación del `end`.
Se puede utilizar `elseif` para añadir más condiciones y construir bloques `if` anidados.

- `for` Repite sentencias un número fijo de veces.
```
for k=1:2:n
  ⋮
  conjunto de sentencias
  ⋮
end
```
En este ejemplo, el conjunto de sentencias se ejecuta una vez haciendo `k=1`. A continuación se incrementa el índice `k` (en el ejemplo `k=3` ya que se ha definido un paso de `2`). Si el nuevo `k` es menor que `n`, se vuelve a ejecutar el grupo de sentencias y así sucesivamente hasta que `k` sea mayor que `n`. En este caso se ejecuta el código que se encuentre a continuación del `end`.
Aunque este parece ser el modo indicado de operar con vectores, en MATLAB® existe una forma mucho más eficiente, por lo que los bucles `for` **deben ser evitados** siempre que sea posible.

- `while` Repite sentencias un número indefinido de veces hasta que se verifique una determinada expresión. Se puede caer en un bucle infinito si la expresión no se verifica nunca.

- `end` Finaliza las sentencias `for`, `while` e `if`.

- `pause` Detiene la ejecución temporalmente.

- `input` Requiere una entrada desde el teclado.

5. Manipulación de matrices y constantes

- `linspace` Genera vectores linealmente espaciados. Es similar al operador "`:`" pero proporciona un control directo sobre el número de puntos del vector.
`t=linspace(1,2,100)` ⇒ genera un vector de 100 puntos en el intervalo [1,2].
`t=1:0.01:2` ⇒ es equivalente al ejemplo anterior y no es necesario especificar el número de puntos del vector. Este es un ejemplo claro de que MATLAB® trabaja con matrices que no requieren ser dimensionadas con antelación a su utilización..

- `logspace` Genera vectores logarítmicamente espaciados.

- `ones / zeros` Crean vectores o matrices cuyos elementos sean todos unos o todos ceros respectivamente.
`x=ones(3,3)` ⇒ crea una matriz `3×3` con todos sus elementos iguales a 1.

- `rand` Crea números y matrices aleatorios uniformemente distribuidos en el intervalo (0,1).
`rand(m,n)` ⇒ crea una matriz formada por `m×n` números aleatorios.

- `ans` — Es la respuesta que se crea automáticamente cuando no se especifica ningún argumento de entrada.
  `5+8` ⇒ crea una variable de nombre `ans` y de valor 13.

- `i` / `j` — Unidad imaginaria.
  `z=x+i*y` ⇒ crea el número complejo `(x,y)`.

- `pi` — Función que devuelve el valor 3.141592654...

- `:` — Es uno de los operadores más útiles de MATLAB®. Puede servir para definir vectores.
  `1:5` ⇒ es el vector `(1, 2, 3, 4, 5)`.
  `1:0.2:2` ⇒ es el vector `(1, 1.2, 1.4, 1.6, 1.8, 2)`.
  Puede servir también para referirse a elementos de una matriz o vector.
  `A(i,:)` ⇒ es la fila i–ésima de la matriz `A`.
  `B(j:k)` ⇒ son los elementos `B(j), B(j+1), ..., B(k)` del vector `B`.

6. Funciones matemáticas elementales

- `abs` — Valor absoluto de un número. Si este número es complejo, devuelve el módulo del número.
  `abs(X)` ⇒ devuelve el módulo de cada uno de los elementos del vector `X` complejo.

- `cos` / `acos` — Funciones coseno y arcocoseno. Para todas las funciones trigonométricas los ángulos se miden en radianes. Añadiendo una "h" se obtienen las correspondientes funciones hiperbólicas.

- `sin` / `asin` — Funciones seno y arcoseno.

- `tan` / `atan` — Funciones tangente y arcotangente.

- `cot` / `acot` — Funciones cotangente y arcocotangente.

- `angle` — Ángulo de fase medido en radianes de un número complejo. Los valores están comprendidos entre $-\pi$ y π.
  `p=angle(Z)` ⇒ devuelve un vector con la fase de cada uno de los elementos del vector complejo `Z`.

- `conj` — Devuelve el valor complejo conjugado de cada uno de los elementos de un vector o una matriz.

- `exp` — Función exponencial. Puede utilizarse también con números complejos.

- `real` / `imag` — Partes real e imaginaria de cada uno de los elementos de un vector o una matriz.

- `log` / `log10` — Logaritmo natural y en base 10 de los elementos de un vector o una matriz.

- `round` — Redondeo al número entero más próximo.

- `sign` — Aplicado a un número, devuelve un "1" si el número es mayor que cero y un "–1" si es menor que cero. Devuelve un cero cuando el número sea el cero.

- `sqrt` — Raíz cuadrada de un número. Puede aplicarse a números negativos creando como resultado un número complejo.

- `max` / `min` Encuentra el valor máximo/mínimo de los elementos de un vector. Si se aplica sobre una matriz se obtiene un vector fila con el máximo/mínimo de cada una de las columnas de la matriz.

- `sum` / `prod` Devuelve la suma o producto de los elementos de un vector. Si se aplica sobre una matriz se obtiene un vector fila con la suma o producto de los elementos de cada columna.

7. Gráficos bidimensionales

- `plot` Gráficos en dos dimensiones.
`plot(y)` ⇒ Representa los elementos de `y` frente a su índice. Si `y` es complejo se representa la parte real en función de la parte imaginaria.
`plot(x,y)` ⇒ Representa los elementos de `y` frente a los elementos de `x`.
`plot(x,y,'b')` ⇒ Representa los elementos de `y` frente a los elementos de `x` utilizando una línea de color azul. El resto de los colores se definen por la inicial de su nombre en inglés.
Ver también la descripción de los comandos siguientes de esta sección. Utilizar `help plot` para una descripción más detallada.

- `semilogx` Funciona igual que `plot` pero representa los resultados en un gráfico con escala logarítmica en el eje X.

- `semilogy` Funciona igual que `plot` pero representa los resultados en un gráfico con escala logarítmica en el eje Y.

- `loglog` Funciona igual que `plot` pero representa los resultados en un gráfico con escala logarítmica en ambos ejes.

- `bar` Gráfico de barras de los elementos de un vector.

- `polar` Gráfico en coordenadas polares del ángulo en radianes frente al radio.

- `stem` Representación de líneas para secuencias de datos discretos. Cada línea termina con un círculo cuya posición representa el valor del dato.

- `grid` Líneas de rejilla para diagramas bidimensionales y tridimensionales.

- `title` Título de un gráfico.
`title('Transformada Rápida de Fourier')` ⇒ escribe el texto `Transformada Rápida de Fourier` como un título en la parte superior del dibujo actual.

- `xlabel` Texto asociado al eje X de un gráfico.

- `ylabel` Texto asociado al eje Y de un gráfico.

- `axis` Modificación de la escala de los ejes.

- `figure` Abre una ventana gráfica nueva para la representación de un nuevo gráfico.

- `close` Cierra la ventana actual.

- `whitebg` Cambia el color de fondo de la figura actual entre el blanco y el negro. Se cambian también otras propiedades de la figura para asegurar que todos los objetos sean visibles con la nueva configuración.