Detalles de Matlab

Formatos de Salida

Respecto a los formatos numéricos con que MATLAB muestra los resultados (recuérdese que siempre calcula y almacena con doble precisión, es decir con unas 16 cifras decimales equivalentes), las posibilidades existentes se muestran en la lista desplegable de la Figura 21 y son las siguientes:

short	coma fija con 4 decimales (***defecto***)
long	coma fija con 15 decimales
hex	cifras hexadecimales
bank	números con dos cifras decimales
short e	notación científica con 4 decimales
short g	notación científica o decimal, dependiendo del valor
long e	notación científica con 15 decimales
long g	notación científica o decimal, dependiendo del valor
rat	expresa los números racionales como cocientes de enteros

Estos formatos se pueden cambiar también desde la línea de comandos anteponiendo la palabra ***format***. Por ejemplo, para ver las matrices en formato ***long*** habrá que ejecutar el comando:

```
>> format long
```

MATLAB mantiene una forma especial para los *números muy grandes* (más grandes que los que es capaz de representar), que son considerados como ***infinito***. Por ejemplo, obsérvese cómo responde el programa al ejecutar el siguiente comando:

```
>> 1.0/0.0
Warning: Divide by zero
ans =
      Inf
```

Así pues, para MATLAB el ***infinito*** se representa como ***inf*** ó ***Inf***. MATLAB tiene también una representación especial para los resultados que no están definidos como números. Por ejemplo, ejecútense los siguientes comandos y obsérvense las respuestas obtenidas:

```
>> 0/0
Warning: Divide by zero
ans =
      NaN
>> inf/inf
ans =
      NaN
```

En ambos casos la respuesta es ***NaN***, que es la abreviatura de ***Not a Number***. Este tipo de respuesta, así como la de ***Inf***, son enormemente importantes en MATLAB, pues permiten controlar la fiabilidad de los resultados de los cálculos matriciales. Los ***NaN*** se propagan al realizar con ellos cualquier operación aritmética, en el sentido de que, por ejemplo,

cualquier número sumado a un ***NaN*** da otro ***NaN***. MATLAB tiene esto en cuenta. Algo parecido sucede con los ***Inf***.

MATLAB dispone de tres funciones útiles relacionadas con las operaciones de coma flotante. Estas funciones, que no tienen argumentos, son las siguientes:

eps	devuelve la diferencia entre 1.0 y el número de coma flotante inmediatamente superior. Da una idea de la precisión o número de cifras almacenadas. En un PC, ***eps*** vale 2.2204e-016.
realmin	devuelve el número más pequeño con que se puede trabajar (2.2251e-308)
realmax	devuelve el número más grande con que se puede trabajar (1.7977e+308)

En muchos cálculos matriciales los datos y/o los resultados no son reales sino ***complejos***, con ***parte real*** y ***parte imaginaria***. MATLAB trabaja sin ninguna dificultad con números complejos. Para ver como se representan por defecto los números complejos, ejecútense los siguientes comandos:

```
>> a=sqrt(-4)
a =
      0 + 2.0000i
>> 3 + 4j
ans =
      3.0000 + 4.0000i
```

En la entrada de datos de MATLAB se pueden utilizar indistintamente la **i** y la **j** para representar el *número imaginario unidad* (en la salida, sin embargo, puede verse que siempre aparece la **i**). Si la **i** o la **j** no están definidas como variables, puede intercalarse el signo (*). Esto no es posible en el caso de que sí estén definidas, porque entonces se utiliza el valor de la variable. En general, cuando se está trabajando con números complejos, conviene no utilizar la **i** como variable ordinaria, pues puede dar lugar a errores y confusiones. Por ejemplo, obsérvense los siguientes resultados:

```
>> i=2
      i =
            2
>> 2+3i
      ans =
            2.0000 + 3.0000i
>> 2+3*i
      ans =
            8
>> 2+3*j
      ans =
            2.0000 + 3.0000i
```

Cuando **i** y **j** son variables utilizadas para otras finalidades, como *unidad imaginaria* puede utilizarse también la función ***sqrt(-1)***, o una variable a la que se haya asignado el resultado de esta función.

Variables

Una ***variable*** es un nombre que se da a una entidad numérica, que puede ser una matriz, un vector o un escalar. El valor de esa variable, e incluso el tipo de entidad numérica que representa, puede cambiar a lo largo de una sesión de MATLAB o a lo largo de la ejecución de un programa.

La forma más normal de cambiar el valor de una variable es colocándola a la izquierda del ***operador de asignación*** (=).

Una expresión de MATLAB puede tener las dos formas siguientes: primero, asignando su resultado a una variable,

```
variable = expresión
```

y segundo evaluando simplemente el resultado del siguiente modo,

```
expresión
```

en cuyo caso el resultado se asigna automáticamente a una variable interna de MATLAB llamada ***ans*** (de *answer*) que almacena el último resultado obtenido. Se considera por defecto que una expresión termina cuando se pulsa ***intro***. Si se desea que una expresión continúe en la línea siguiente, hay que introducir ***tres puntos*** (...) antes de pulsar ***intro***. También se pueden incluir varias expresiones en una misma línea separándolas por *comas* (,) o *puntos y comas* (;).

Si una expresión ***termina en punto y coma*** (;) su resultado se calcula, pero no se escribe en pantalla. Esta posibilidad es muy interesante, tanto para evitar la escritura de resultados intermedios, como para evitar la impresión de grandes cantidades de números cuando se trabaja con matrices de gran tamaño.

MATLAB puede definir variables que contengan cadenas de caracteres. En MATLAB las cadenas de texto van entre apóstrofos o comillas simples (Nótese que en lenguaje C van entre comillas dobles: "cadena"). Por ejemplo, en MATLAB:

```
s = 'cadena de caracteres'
```

Las cadenas de texto tienen su más clara utilidad en temas que se verán más adelante y por eso se difiere hasta entonces una explicación más detallada.

A semejanza de C, ***MATLAB distingue entre mayúsculas y minúsculas*** en los nombres de variables. Los ***nombres de variables*** deben empezar siempre por una letra y pueden constar de hasta 31 letras y números. El carácter guión bajo (_) se considera como una letra. A diferencia del lenguaje C, no hace falta declarar las variables que se vayan a utilizar. Esto hace que se deba tener especial cuidado con no utilizar nombres erróneos en las variables, porque no se recibirá ningún aviso del ordenador.

Cuando se quiere tener una *relación de las variables* que se han utilizado en una sesión de

trabajo se puede utilizar el comando ***who***. Existe otro comando llamado ***whos*** que proporciona además información sobre el tamaño, la cantidad de memoria ocupada y el carácter real o complejo de cada variable. Se sugiere utilizar de vez en cuando estos comandos en la sesión de MATLAB que se tiene abierta. Esta misma información se puede obtener gráficamente con el ***Workspace Browser***, que aparece con el comando ***View/Workspace*** o activando la ventana correspondiente si estaba ya abierto.
El comando ***clear*** tiene varias formas posibles:

clear	sin argumentos, elimina todas las variables creadas previamente (excepto las variables globales).
clear A, b	borra las variables indicadas.
clear global	borra las variables globales.
clear functions	borra las funciones.
clear all	borra todas las variables, incluyendo las globales, y las funciones.

Guardar variables y estados de una sesión: Comandos save y load

En muchas ocasiones puede resultar interesante interrumpir el trabajo con MATLAB y poderlo recuperar más tarde en el mismo punto en el que se dejó (con las mismas variables definidas, con los mismos resultados intermedios, etc.). Hay que tener en cuenta que al salir del programa todo el contenido de la memoria se borra automáticamente.

Para guardar el estado de una sesión de trabajo existe el comando ***save***. Si se teclea:

```
>> save
```

antes de abandonar el programa, se crea en el ***directorio actual*** un fichero binario llamado ***matlab.mat*** (o ***matlab***) con el estado de la sesión (excepto los gráficos, que por ocupar mucha memoria hay que guardar aparte). Dicho estado puede recuperarse la siguiente vez que se arranque el programa con el comando:

```
>> load
```

Esta es la forma más básica de los comandos ***save*** y ***load***. Se pueden guardar también matrices y vectores de forma selectiva y en ficheros con nombre especificado por el usuario. Por ejemplo, el comando (sin comas entre los nombres de variables):

```
>> save filename A x y
```

guarda las variables **A**, **x** e **y** en un fichero binario llamado ***filename.mat*** (o ***filename***). Para recuperarlas en otra sesión basta teclear:

```
>> load filename
```

Operadores relacionales

El lenguaje de programación de MATLAB dispone de los siguientes operadores relacionales:

<	menor que
>	mayor que
<=	menor o igual que
>=	mayor o igual que
==	igual que
~=	distinto de

Obsérvese que, salvo el último de ellos, coinciden con los correspondientes operadores relacionales de C. Sin embargo, ésta es una coincidencia más bien formal. En MATLAB los operadores relacionales pueden aplicarse a vectores y matrices, y eso hace que tengan un significado especial.

Al igual que en C, si una comparación se cumple el resultado es 1 (***true***), mientras que si no se cumple es 0 (***false***). Recíprocamente, cualquier valor distinto de cero es considerado como ***true*** y el cero equivale a ***false***. La diferencia con C está en que cuando los operadores relacionales de MATLAB se aplican a dos matrices o vectores del mismo tamaño, *la comparación se realiza elemento a elemento*, y el resultado es otra matriz de unos y ceros del mismo tamaño, que recoge el resultado de cada comparación entre elementos.

Operadores lógicos (booleanos)

Los operadores lógicos de MATLAB son los siguientes:

&	and
|	or
~	negación lógica

Obsérvese que estos operadores lógicos tienen distinta notación que los correspondientes operadores de C (&&, || y !). Los operadores lógicos se combinan con los relacionales para poder comprobar el cumplimiento de condiciones múltiples. Más adelante se verán otros ejemplos y ciertas funciones de las que dispone MATLAB para facilitar la aplicación de estos operadores a vectores y matrices.

FUNCIONES

MATLAB tiene un gran número de funciones incorporadas. Algunas son ***funciones intrínsecas***, esto es, funciones incorporadas en el propio código ejecutable del programa. Estas funciones son particularmente rápidas y eficientes. Existen además funciones definidas en ficheros ****.m*** y ****.mex*** que vienen con el propio programa o que han sido aportadas por usuarios del mismo. Estas funciones extienden en gran manera las posibilidades del programa.

MATLAB dispone también de ficheros ****.p***, que son los ficheros ****.m*** pre-compilados con la función ***pcode***. Se verán más adelante.

Recuérdese que para que MATLAB encuentre una determinada función de usuario el correspondiente *fichero-M* debe estar en el ***directorio actual*** o en uno de los directorios del ***search path***.

Características generales de las funciones de MATLAB

El concepto de función en MATLAB es semejante al de C y al de otros lenguajes de programación, aunque con algunas diferencias importantes. Al igual que en C, una función tiene ***nombre***, ***valor de retorno*** y ***argumentos***. Una función *se llama* utilizando su nombre en una expresión o utilizándolo como un comando más. Las funciones se pueden definir en ficheros de texto ****.m*** en la forma que se verá más adelante. Considérense los siguientes ejemplos de llamada a funciones:

```
>> [maximo, posmax] = max(x);
>> r = sqrt(x^2+y^2) + eps;
>> a = cos(alfa) - sin(alfa);
```

donde se han utilizado algunas funciones matemáticas bien conocidas como el cálculo del valor máximo, el seno, el coseno y la raíz cuadrada. Los ***nombres*** de las funciones se han puesto en negrita. Los ***argumentos*** de cada función van a continuación del nombre entre paréntesis (y separados por comas si hay más de uno). Los ***valores de retorno*** son el resultado de la función y sustituyen a ésta en la expresión donde la función aparece.

Una diferencia importante con otros lenguajes es que en MATLAB las funciones pueden tener ***valores de retorno matriciales múltiples*** (ya se verá que pueden recogerse en variables *ad hoc* todos o sólo parte de estos valores de retorno), como en el primero de los ejemplos anteriores. En este caso se calcula el elemento de máximo valor en un vector, y se devuelven dos valores: el valor máximo y la posición que ocupa en el vector. Obsérvese que los 2 valores de retorno se recogen entre corchetes, separados por comas.

Una característica de MATLAB es que las funciones que no tienen argumentos no llevan paréntesis, por lo que a simple vista no siempre son fáciles de distinguir de las simples variables. En la segunda línea de los ejemplos anteriores, ***eps*** es una función sin argumentos, que devuelve la diferencia entre 1.0 y el número de coma flotante inmediatamente superior. En lo sucesivo el nombre de la función irá seguido de paréntesis si interesa resaltar que la función espera que se le pase uno o más argumentos.

Los nombres de las funciones de MATLAB no son *palabras reservadas* del lenguaje. Es posible crear una variable llamada ***sin*** o ***cos***, que ocultan las funciones correspondientes. Para poder acceder a las funciones hay que eliminar (***clear***) las variables del mismo nombre que las ocultan.

MATLAB permite que una función tenga un número de argumentos y valores de retorno variables, lo cual se determina sólo en tiempo de ejecución. Más adelante se verá cómo se hace esto.

Funciones matemáticas elementales que operan de modo escalar

Estas funciones, que comprenden las funciones matemáticas trascendentales y otras funciones básicas, actúan sobre cada elemento de la matriz como si se tratase de un escalar. Se aplican de la misma forma a escalares, vectores y matrices. Algunas de las funciones de este grupo son las siguientes:

sin(x)	seno
cos(x)	coseno
tan(x)	tangente
asin(x)	arco seno
acos(x)	arco coseno
atan(x)	arco tangente (devuelve un ángulo entre $-\pi/2$ y $+\pi/2$)
atan2(x)	arco tangente (devuelve un ángulo entre $-\pi$ y $+\pi$); se le pasan 2 argumentos, proporcionales al seno y al coseno
sinh(x)	seno hiperbólico
cosh(x)	coseno hiperbólico
tanh(x)	tangente hiperbólica
asinh(x)	arco seno hiperbólico
acosh(x)	arco coseno hiperbólico
atanh(x)	arco tangente hiperbólica
log(x)	logaritmo natural
log10(x)	logaritmo decimal
exp(x)	función exponencial
sqrt(x)	raíz cuadrada
sign(x)	devuelve -1 si <0, 0 si =0 y 1 si >0. Aplicada a un número complejo, devuelve un vector unitario en la misma dirección
rem(x,y)	resto de la división (2 argumentos que no tienen que ser enteros)
mod(x,y)	similar a ***rem***
round(x)	redondeo hacia el entero más próximo
fix(x)	redondea hacia el entero más próximo a 0
floor(x)	valor entero más próximo hacia $-\infty$
ceil(x)	valor entero más próximo hacia $+\infty$
gcd(x)	máximo común divisor
lcm(x)	mínimo común múltiplo
real(x)	partes reales
imag(x)	partes imaginarias
abs(x)	valores absolutos
angle(x)	ángulos de fase

FICHEROS DE COMANDOS (*SCRIPTS*)

Como ya se ha dicho, los ficheros de comandos o *scripts* son ficheros con un nombre tal como ***file1.m*** que contienen una sucesión de comandos análoga a la que se teclearía en el uso interactivo del programa. Dichos comandos se ejecutan sucesivamente cuando se teclea el nombre del fichero que los contiene (sin la extensión), es decir cuando se teclea ***file1*** con el ejemplo considerado.

Cuando se ejecuta desde la línea de comandos, las variables creadas por ***file1*** pertenecen al espacio de trabajo base de MATLAB. Por el contrario, si se ejecuta desde una función, las variables que crea pertenecen al espacio de trabajo de la función. En los ficheros de comandos conviene poner los puntos y coma (;) al final de cada sentencia, para evitar una salida de resultados demasiado cuantiosa. Un fichero ****.m*** puede llamar a otros ficheros ****.m***, e incluso se puede llamar a sí mismo de modo recursivo. Sin embargo, no se puede hacer ***profile*** de un fichero de comandos: sólo se puede hacer de las funciones.

Las variables definidas por los ficheros de comandos son variables del espacio de trabajo desde el que se ejecuta el fichero, esto es variables con el mismo carácter que las que se crean interactivamente en MATLAB si el fichero se ha ejecutado desde la línea de comandos. Al terminar la ejecución del *script*, dichas variables permanecen en memoria.

Mención especial merece el fichero de comandos ***startup.m***. Este fichero se ejecuta cada vez que se entra en MATLAB. En él puede introducir todos aquellos comandos que le interesa se ejecuten siempre al iniciar la sesión, por ejemplo ***format compact*** y los comandos necesarios para modificar el ***path***.

DEFINICIÓN DE FUNCIONES

La ***primera línea*** de un fichero llamado ***name.m*** que define una función tiene la forma: **`function [lista de valores de retorno] = name(lista de argumentos)`** donde ***name*** es el nombre de la función. Entre corchetes y separados por comas van los ***valores de retorno*** (siempre que haya más de uno), y entre paréntesis también separados por comas los ***argumentos***. Puede haber funciones sin valor de retorno y también sin argumentos. Recuérdese que los ***argumentos*** son los ***datos*** de la función y los ***valores de retorno*** sus ***resultados***. Si no hay valores de retorno se omiten los corchetes y el signo igual (=); si sólo hay un valor de retorno no hace falta poner corchetes. Tampoco hace falta poner paréntesis si no hay argumentos.

Una diferencia importante con C/C++/Java es que en MATLAB una función no modifica nunca los argumentos que recibe. Los resultados de una función de MATLAB se obtienen siempre a través de los valores de retorno, que pueden ser múltiples y matriciales. Tanto el número de argumentos como el de valores de retorno no tienen que ser fijos, dependiendo de cómo el usuario llama a la función.

Las variables definidas dentro de una función son ***variables locales***, en el sentido de que son inaccesibles desde otras partes del programa y en el de que no interfieren con variables del mismo nombre definidas en otras funciones o partes del programa. Se puede decir que pertenecen al propio espacio de trabajo de la función y no son vistas desde otros espacios de trabajo. Para que la función tenga acceso a variables que no han sido pasadas como argumentos es necesario declarar dichas variables como ***variables globales***, tanto en el programa principal como en las distintas funciones que deben acceder a su valor. Es frecuente utilizar el convenio de usar para las variables globales nombres largos (más de 5 letras) y con mayúsculas.

Por razones de eficiencia, los argumentos que recibe una función de MATLAB no se copian a variables locales si no son modificados por dicha función (en términos de C/C++ se diría que se pasan ***por referencia***). Esto tiene importantes consecuencias en términos de eficiencia y ahorro de tiempo de cálculo. Sin embargo, si dentro de la función se realizan modificaciones sobre los argumentos recibidos, antes se sacan copias de dichos argumentos a variables locales y se modifican las copias (diríase que en este caso los argumentos se pasan ***por valor***).

Dentro de la función, los valores de retorno deben ser calculados en algún momento (no hay sentencia ***return***, como en C/C++/Java). De todas formas, no hace falta calcular siempre todos los posibles valores de retorno de la función, sino sólo los que el usuario *espera obtener* en la sentencia de llamada a la función. En cualquier función existen dos variables definidas de modo automático, llamadas ***nargin*** y ***nargout***, que representan respectivamente el número de argumentos y el número de valores de retorno con los que la función ha sido llamada. Dentro de la función, estas variables pueden ser utilizadas como el programador desee.