3.1 Funciones de biblioteca

- **Concepto de función. Parámetros o argumentos**

Una función es un módulo independiente programado para realizar una tarea específica. Internamente está constituida por un conjunto de instrucciones y puede utilizarse desde cualquier sentencia escrita en la ventana de comandos, desde cualquier programa realizado por el usuario, así como desde otra función.

MATLAB y Octave disponen de gran cantidad de funciones propias (intrínsecas o que vienen con el propio software) que facilitan la realización de cálculos y programas.

Una función puede considerarse como una entidad que en general, recibe unos datos, (parámetros o argumentos de entrada), realiza operaciones con éstos, generando unos resultados (parámetros o argumentos de salida).
Para llamar a una función de biblioteca, en primer lugar se debe conocer los argumentos de entrada que necesita y los resultados que va a devolver. Para ello es útil utilizar el comando help referido a la función.

Supongamos una función de nombre `nombrefuncion`, la llamada a esta función se realizaría de la forma:

```
[s_1,s_2,…]=nombrefuncion( arg_1,  arg_2, ...)
```

donde:

`arg_i` son los datos que queremos enviar a la función. Se denominan argumentos reales o actuales. Se transmiten pasando una copia de su valor a los argumentos de entrada de la función (argumentos formales).

`s_i` son las variables que reciben los datos de salida de la función.

Cuando una función es aplicable a escalares, si se aplica a una matriz realizará la operación elemento a elemento: $f([A]) = [f(A_{ij})]$

En los módulos anteriores se han estudiado algunas funciones de biblioteca, se analizan a continuación otras, divididas en diferentes categorías, que pueden ser de utilidad en el uso del lenguaje M. Estas funciones sólo son una pequeña parte de la totalidad de las funciones existentes. El alumno puede investigar otras fácilmente a través de la ayuda del programa.

- **Funciones matemáticas**

– **Funciones para cálculos básicos**

`abs(x)` obtiene el valor absoluto de `x`

`rem(x,y)` obtiene el resto de la división de `x` entre `y`

`sqrt(x)` obtiene la raíz cuadrada de `x`

`log(x)` obtiene el logaritmo neperiano de `x`

`log2(x)` obtiene el logaritmo base 2 de `x`

`log10(x)` obtiene el logaritmo base 10 de `x`

`exp(x)` obtiene la exponencial de `x` (e^x)

`sign(x)` retorna 1 si `x`>0, 0 si `x`=0, y -1 si `x`<0

– **Funciones para cálculos básicos estadísticos**

`mean(x)` calcula la media de los valores del vector `x`

`std(x)` calcula la desviación típica de los valores del vector `x`

– **Funciones trigonométricas**

Obtienen las siguientes razones trigonométricas de `x`:

`sin(x)` `sinh(x)` seno y seno hiperbólico

`cos(x)` `cosh(x)` coseno y coseno hiperbólico

`tan(x)` `tanh(x)` tangente y tangente hiperbólica

`cot(x)` `coth(x)` cotangente y cotangente hiperbólica

`csc(x)` `csch(x)` cosecante y cosecante hiperbólica

`sec(x)` `sech(x)` secante y secante hiperbólica

`asin(x)` `asinh(x)` arcoseno y arcoseno hiperbólico

`acos(x)` `acosh(x)` arcocoseno y arcocoseno hiperbólico

`atan(x)` `atanh(x)` arcotangente y arcotangente hiperbólica

`acot(x)` `acoth(x)` arcocotangente y arcocotangente hiperbólica

`acsc(x)` `acsch(x)` arcocosecante y arcocosecante hiperbólica

`asec(x)` `asech(x)` arcosecante y arcosecante hiperbólica

- **Funciones de redondeo**

`fix(x)` elimina la parte decimal del dato `x`

`floor(x)` obtiene el mayor entero menor o igual a `x`

`ceil(x)` obtiene el menor entero por encima de `x`

`round(x)` redondea `x` al entero más cercano

- **Funciones para trabajar con polinomios**

El lenguaje M dispone de funciones para realizar operaciones estándar con polinomios: búsqueda de raíces, evaluación, interpolación, ….

`polyval(p,x)` evalúa el polinomio de coeficientes `p` en `x`. Si x es una matriz evalúa el polinomio en cada elemento de `x`. Para una evaluación matricial se debe usar la función `polyvalm(p,x)`.

`roots(p)` obtiene un vector columna con las raíces del polinomio de coeficientes `p`.

`poly(r)` obtiene un vector con los coeficientes del polinomio de raíces `r`.

`polyder(p)` obtiene un vector con los coeficientes del polinomio resultado de derivar `p`.

`conv(u,v)` obtiene los coeficientes del polinomio resultado de multiplicar los polinomios de coeficientes `u` y `v`.

`[p,q]=deconv(u,v)` devuelve los polinomios cociente y resto de la división de los polinomios `u` entre `v`.

`polyfit(x,y,n)` devuelve el polinomio de grado `n` que ajusta los puntos (`x`,`y`) en el sentido de mínimos cuadrados.

- **Funciones para búsqueda de condiciones lógicas**

`any(condición)` si en la condición interviene un vector, la función devuelve 1 si esta condición se cumple al menos en un elemento del vector, cero en caso contrario. Si en la condición interviene una matriz, se aplica la función a cada columna de ésta dando como resultado un vector de 0 y 1.

`all(condición)` similar a la anterior, pero en este caso se chequea si todos los elementos cumplen la condición, en ese caso devuelve cierto.

`find(condición)` realiza la búsqueda de los elementos que cumplen la condición. El resultado es un vector columna con los índices de los elementos que cumplan la condición. Si en la condición interviene una matriz, el resultado también es un vector columna, al considerar la matriz como una columna detrás de otra.

Veamos a continuación unos ejemplos de utilización:

```
>> x=[1 4 7]


x =

     1     4     7


>> any(x<0)


ans =

     0


>> A=[1 2 3;-1 7 2;6 1 3]


A =

     1     2     3

    -1     7     2

     6     1     3


>> any(A<0)


ans =

     1     0     0


>> all(x>0)
```

```
ans =

     1


>> all(A>0)


ans =

     0     1     1


>> find(x>0)


ans =

     1     2     3


>> find(A<0)


ans =

     2
```