2.2 Operaciones con matrices. Funciones específicas.

- **Operaciones con matrices mediante operadores**

M puede operar con matrices[1] por medio de *operadores* y por medio de *funciones*.

Sean `A` y `B` dos matrices y `c` un escalar. Los operadores matriciales del lenguaje M son los siguientes:

– Operadores suma (+) y resta (-)

Utilizado entre matrices (siempre con el mismo tamaño) obtiene la suma/resta matricial (elemento a elemento). Utilizado entre una matriz y un escalar, suma/resta el escalar a cada elemento de la matriz. No existen los operadores (.+) y (.-).

– Operador producto (*)

Utilizado entre matrices resuelve el producto matricial. Las dimensiones de las matrices deben ser congruentes. Utilizado entre una matriz y un escalar, multiplica el escalar por cada elemento de la matriz.

– Operador producto elemento a elemento (.*)

A.*B da como resultado una matriz cuyo elemento ij es Aij*Bij.

– Operador potenciación (^)

Para utilizarlo, al menos uno de los operandos debe ser un escalar y la matriz debe ser cuadrada.

`A^c` resuelve el producto matricial `A*A*A ..... *A`, `c` veces.

`c^A` se computa mediante autovalores y autofunciones.

– Operador potenciación elemento a elemento (.^)

A continuación aparecen todas las posibilidades de utilización:

`A.^B` da como resultado una matriz cuyo elemento ij es A_{ij}^B_{ij}.

`A.^c` da como resultado una matriz cuyo elemento ij es A_{ij}^`c`.

`c.^A` da como resultado una matriz cuyo elemento ij es `c`^A_{ij}.

– Operador división (/) (\)

La utilización entre una matriz y un escalar obtiene el cociente elemento a elemento. La utilización entre matrices se verá más adelante.

– Operador división elemento a elemento (./) (.\)

A continuación aparecen todas las posibilidades de utilización:

`A./B` da como resultado una matriz cuyo elemento `ij` es A_{ij} /B_{ij}.

`A.\B` da como resultado una matriz cuyo elemento `ij` es B_{ij} /A_{ij}.

[1] En esta sección, hablaremos de matrices de forma global, considerando también los vectores.

- Operador traspuesta (')

`A'` da como resultado la matriz traspuesta de `A`.

Algunos ejemplos:

```
>>a=[1 2 3];b=[4 5 6];c=3;


>>a+b

       ans=        5    7    9

>>a.*b

       ans=        4   10   18

>>a-b

       ans=        -3   -3   -3

>>a.^b

       ans=        1   32   729

>>a+c

       ans=        4    5    6

>>a*c

       ans=        3    6    9

>>a.^c

       ans=        1    8    27

>>c.^a

       ans=        3    9    27
```

- Operadores relacionales, lógicos y de igualdad

Son válidos los analizados en la sección 1.3. Aplicados entre matrices se emplean elemento a elemento, luego el tamaño de las matrices debe coincidir. El resultado es una matriz de tipo lógico.

```
>> A=1:9;

>>P=(A>2)&(A<6)

P=

     0  0  1  1  1  0  0  0  0
```

- **Operaciones con matrices mediante funciones**

Además de los operadores analizados en la sección anterior, existen funciones M que permiten realizar otro tipo de operaciones con vectores:

`dot(v,w)` producto escalar de los vectores `v` y `w`.

`cross(v,w)` producto vectorial de los vectores `v` y `w` (máximo tres componentes de cada uno): $[v_2w_3-v_3w_2 \quad v_3w_1-v_1w_3 \quad v_1w_2-v_2w_1]$.

`length(v)` calcula el número de componentes del vector `v`, o el máximo tamaño de las dimensiones si `v` es una matriz.

`[m,n]=size(A)` devuelve el número de filas y de columnas de la matriz `A`. Si la matriz es cuadrada basta recoger el primer valor de retorno.

`size(A,1)` devuelve el número de filas.

`size(A,2)` devuelve el número de columnas.

`max(v)` devuelve el valor de la mayor componente del vector `v`, o si `v` es una matriz genera un vector fila con el máximo de cada columna de la matriz.

`min(v)` devuelve el valor de la menor componente del vector `v`, o si `v` es una matriz genera un vector fila con el mínimo de cada columna de la matriz.

`norm(v)` obtiene el módulo del vector `v`

`sum(v)` devuelve la suma de las componentes del vector `v`. Si `v` es matriz genera un vector fila, siendo cada elemento igual a la suma de la columna correspondiente de la matriz.

`prod(v)` devuelve el producto de las componentes del vector `v`. Si `v` es matriz genera un vector fila, siendo cada elemento igual al producto de la columna correspondiente de la matriz.

`sort(v)` ordena las componentes de `v` de menor a mayor. Si `v` es una matriz, ordena sus columnas de menor a mayor. Por ejemplo, para ordenar un vector v, bastaría con ejecutar la siguiente instrucción:

```
v=sort(v)
```

A continuación se detallan algunas funciones que operan con matrices:

`inv(A)` da como resultado la matriz inversa de A.

`det(A)` da como resultado el determinante de A.

`trace(A)` da como resultado la traza de A.

- **Funciones para definir matrices particulares**

Existen en el lenguaje M varias funciones orientadas a definir con gran facilidad matrices de tipos particulares. Algunas de estas funciones son las siguientes:

`eye(n)` forma la matriz *identidad* de tamaño (nxn)

`eye(m,n)` forma la matriz *identidad* de tamaño (mxn)

`zeros(m,n)` forma una matriz de *ceros* de tamaño (mxn)

`zeros(n)` forma una matriz de *ceros* de tamaño (nxn)

`ones(n)` forma una matriz de *unos* de tamaño (nxn)

`ones(m,n)` forma una matriz de *unos* de tamaño (mxn)

Ejemplos:

```
>>eye(2)

                  1   0

                  0   1

>>zeros(2,3)

                      0    0    0

                      0    0    0

>>ones(4)

                      1  1  1  1

                      1  1  1  1

                      1  1  1  1

                      1  1  1  1
```

- **Números y matrices aleatorios**

Para la generación de números y matrices pseudoaleatorios, M dispone de las siguientes funciones:

`rand` este comando genera números pseudoaleatorios distribuidos uniformemente entre 0 y 1. Cada llamada proporciona un nuevo número.

`rand(n)` genera una matriz de números pseudoaleatorios entre 0 y 1, con distribución uniforme, de tamaño (`nxn`).

`rand(m,n)` igual que en el caso anterior pero de tamaño (`mxn`).

Ejemplos:

```
>>rand

                        0.9501

>>rand(3)

                  0.2311    0.8913     0.0185

                  0.6068    0.7621     0.8214

                  0.4860    0.4565     0.4447
```

- **Borrado de elementos de matrices y vectores**

Se pueden crear nuevas matrices eliminando elementos de otras ya existentes. Para borrar elementos de una matriz o vector, se debe asignar a éstos el valor vacío entre corchetes **[ ]**. Ejemplo:

```
A(3,:)=[  ];
```

con esta orden se elimina la fila 3 de la matriz A.

- **Extracción de elementos de tablas mediante índices lógicos**

Conocemos que para extraer elementos de una tabla, se deben indicar los índices correspondientes a los elementos; sin embargo, si como índices de una tabla disponemos un vector de tipo lógico, estamos indicando que se extraigan los elementos situados en las posiciones de valor lógico 1. Ejemplo:

```
>>v=1:10

v =

     1     2     3     4     5     6     7     8     9     10

>> in=v>=5

in =

     0     0     0     0     1     1     1     1     1     1

>> v(in)

ans =

     5     6     7     8     9    10

% se extraen los elementos de v que cumplen la condición, es decir los
elementos de v mayores o iguales a 5
```

- **Reutilización del formato en escritura de matrices en pantalla**

Cuando en lugar de un escalar se quiere escribir una matriz, se imprimirán los elementos en orden columnas, necesitando por cada elemento un formato. Sin embargo, en lenguaje M, ya conocemos que no es necesaria la disposición explícita de un formato por cada dato ya que cuando se termina de usar el formato especificado se reutiliza éste al completo hasta conseguir escribir el resto de datos. A continuación se escriben ejemplos relativos a esta propiedad:

```
>> A=[1 2;3 4]; B=[3.56 7.89];

>> fprintf('%d %d\n',A,B);

1 3

2 4

3.560000e+000 7.890000e+000


>> fprintf('El dato es: %f\n',B);

El dato es: 3.560000

El dato es: 7.890000
```