

## Introducción

Una razón por la que Octave es un lenguaje muy potente es que se caracteriza por ser un programa funcional. Esto quiere decir que tiene múltiples funciones que pueden ser aplicadas a objetos más complejos que un número, por ejemplo una matriz. Una matriz es una herramienta matemáticas que permite tratar mucha información de una manera muy eficiente y flexible. Por ejemplo, una matriz de píxeles puede ser una imagen, o una película; o una matriz de fluctuaciones puede ser un sonido o una voz humana. De este modo vamos a ver en este número como se tratan las matrices en Octave y algunas funciones asociadas a ellas.

## 1 Matrices

Una matriz de números es un objeto matemático que contiene números colocados por filas y columnas. En octave estas matrices irán entre corchetes([ ]) y dentro escribiremos las filas separados por espacio o comas y las columnas por punto y coma. Veamos algunos ejemplos,

```
octave:1> A=[1 2 3;4 5 6]
A =

     1     2     3
     4     5     6
```

```
octave:2> b=[1 10 100]
b =

     1    10   100
```

De este modo hemos creado dos matrices:  $A$  con 2 filas y 3 columnas, y la matriz  $b$  con una fila y 3 columnas (esta matriz más propiamente es llamada *vector fila*). Basado en la posición en la que se encuentran los elementos de las matrices podemos extraer sus elementos o bien trozos de la matriz, lo que se denominan submatrices.

```
octave:3> A(2,3)
ans = 6
octave:4> A(:, [1 3])
ans =

     1     3
     4     6
```

Cuando escribimos dos puntos(esto es, :) significa que tomamos o bien todas las filas o bien todas las columnas según corresponda.

## 1.1 Operaciones con matrices

Pues las matrices si son del mismo tamaño se pueden sumar, restar o multiplicar por un escalar.

```
octave:5> A+A
ans =

     2     4     6
     8    10    12
```

```
octave:6> 2*A
ans =

     2     4     6
     8    10    12
```

Lo que no podemos es multiplicar dos matrices cualesquiera, sino que el número de columnas de la de la izquierda debe coincidir con el número de filas de la de la derecha. ¿Recuerdas eso?

```
octave:7> A*b
error: operator *: nonconformant arguments (op1 is 2x3, op2 is 1x3)
octave:7> A*b'
ans =

    321
    654

octave:8> A'*A
ans =

    17    22    27
    22    29    36
    27    36    45
```

Observa que la prima ' lo que hace es transponer y conjugar la matriz, esto es cambiar las filas por columnas y viceversa y si los números de las matrices fuesen complejos los pasaría a sus conjugados.

Como hemos dicho antes *Octave* es funcional. Fíjate que de una vez podemos calcular el cuadrado, o el cubo, o el seno trigonométrico, o la tangente, o la raíz cuadrada de todos los elementos de la matriz  $A$ , por ejemplo:

```
octave:9> A.^2
ans =

    1     4     9
   16    25    36

octave:10> A.^3
ans =

    1     8    27
   64   125   216

octave:11> sin(A)
ans =

    0.84147    0.90930    0.14112
   -0.75680   -0.95892   -0.27942

octave:12> tan(A)
ans =

    1.55741   -2.18504   -0.14255
    1.15782   -3.38052   -0.29101

octave:13> sqrt(A)
ans =

    1.0000    1.4142    1.7321
    2.0000    2.2361    2.4495.
```

La última operación usual en matrices es la inversa de una matriz. No todas las matrices tienen inversa; estas tienen que ser cuadradas (mismo número de filas que de columnas) y además su determinante debe ser distinto de cero. En este caso podemos calcular la inversa y se calcula con el comando *inv*.

```
octave:15> C=[1 2;2 5]
C =

     1     2
     2     5

octave:16> det(A)
error: det: argument must be a square matrix
octave:16> C=[1 2;2 5]
C =

     1     2
     2     5

octave:17> det(C)
ans = 1
octave:18> inv(C)
ans =

     5    -2
    -2     1

octave:19> ans*C
ans =

     1     0
     0     1
```

## 1.2 Generando algunas matrices

Tenemos muchos tipos de matrices:

- Caracterizadas por su forma: cuadradas, rectangulares, triangulares superiores, triangulares inferiores, escalonadas, etc.
- Caracterizadas por su contenido: diagonales, tridiagonales, huecas o sparse, aleatorias, etc.
- Caracterizadas por propiedades intrínsecas: Pascal, Vandermonde, Hilbert, Toeplitz, Hankel, etc.

Os animamos a que investiguéis los siguientes instrucciones con nuevos comandos en octave: *rand(2,3)*, *diag([1 2 3 4])*, *tril(A)*, *triu(A)*, *ones(2,3)*, *zeros(2,3)*, *eye(3)*, *vander([1 2 3 4])* y *pascal(3)*. ALguna pista de lo que va a salir puedes mirarlo en lo que sigue.

```
octave:21> A1=rand(2,3);A2=diag([100 200]);A3=tril(A);
A4=triu(A);A5=ones(2,3);A6=zeros(2,3);A7=eye(2);
octave:22> %¿Las ponemos todas en una sola?
octave:22> [A1 A2 A3 A4 A5 A6 A7]
ans =

Columns 1 through 7:

    0.66373    0.80437    0.18190   100.00000    0.00000    1.00000    0.00000
    0.08936    0.00524    0.91602    0.00000   200.00000    4.00000    5.00000

Columns 8 through 14:

    0.00000    1.00000    2.00000    3.00000    1.00000    1.00000    1.00000
    0.00000    0.00000    5.00000    6.00000    1.00000    1.00000    1.00000

Columns 15 through 19:

    0.00000    0.00000    0.00000    1.00000    0.00000
    0.00000    0.00000    0.00000    0.00000    1.00000
```

## 1.3 Algunos comandos interesantes

La aplicabilidad de las matrices es enorme y en la ingeniería una matriz aparece en múltiples aplicaciones, desde la resolución de sistemas de ecuaciones, la resolución de ecuaciones diferenciales, el procesamiento de señales, simulaciones, etc. En futuros números estudiaremos algunos problemas en ingeniería donde podamos ver la importancia de las matrices y el uso de *Octave* en ella. Por el momento, os animamos a que vayáis investigando algunos comandos como son *eig* que calcula los autovalores de una matriz, *svd* que calcula la descomposición en valores singulares o *lu* que calcula la factorización de una matriz basada en la eliminación gaussiana.



¿Quieres ir al **supermercado** a comprar gratis algunas matrices?

Esta es la dirección:  
[math.nist.gov/MatrixMarket](http://math.nist.gov/MatrixMarket)

## 2 Procesando una foto

¿Qué es una foto? Es una colección ordenada de cuadraditos del mismo tamaño, cada uno de ellos coloreado de un color determinado. Cuanto más pequeño es el cuadrado mayor resolución de la foto y cuanto más grande, menor resolución y la calidad de la foto no será tan buena. Entonces podríamos pensar en una foto como un rectángulo lleno de cuadraditos del mismo tamaño y que cada cuadrado tiene la información del color del cuadrado, ¿verdad? Es decir

una foto es una matriz de colores. Vamos a tomar una foto, ver esa matriz que está escondida en ella y jugaremos un poco con la imagen. Para ello en vez de trabajar como hasta ahora, que hemos ido escribiendo en la pantalla de ejecución de octave cada instrucción por separado, ahora vamos a juntar muchas instrucciones juntas en un archivo, que se llama *script*, que tiene extensión *.m* y llamaremos en la pantalla de ejecución de octave a este archivo por su nombre. En windows te recomendamos que utilices como editor *notepad++* (¡Cuidado no se pueden usar editores estilo word ya que estos guardan formatos además del texto!); y en linux, por ejemplo os recomendamos utilizar el editor *emacs*. EL programa que hemos generado es el siguiente:

```
% El % lo usamos para hacer comentarios, esto no afecta a octave, no
% lee ni interpretalo que haya en las líneas que comienzan con %

clear all
% Vamos a crear la matriz A que representa la foto "fotoNena.jpg"
A=imread("fotoNena.jpg");

%El ; lo colocamos para que no saque por pantalla la matriz A

%Veamos cuantas filas y columnas tiene la matriz A
disp("La foto está presentada en la matriz A de tamaño:")
size(A)
% Vamos a crear una figura con una fila y tres columnas
subplot(1,3,1)
% en la primera ponemos la foto "fotoNena.jpg"
imshow(A)

% en la segunda vamos a mostrarla rotada 45º

b=imrotate(A,45,'bilinear','crop');
subplot(1,3,2)
imshow(b)

% la pasamos a blanco y negro

c=rgb2gray(A);
subplot(1,3,3)
imshow(c)
```

Ahora mostramos en la Figura 1 la salida que genera el programa anterior.

