# Notas de clase 30/03

```
clear; clc; %Limpieza de command window y workspace
m=4:10  %Creación de matrices por extensión
```

```
m = 1×7
     4     5     6     7     8     9    10
```

MATLAB reconoce la creción de matrices por extensión. La manera M=a:b:c (con a,b,c variables con números reales) creará una matriz con números desde a hasta c, con intervalos de b entre los números. En el caso que c>a y b>0 creará un vector vacío, pues no habrá números que cumplan la condición.

```
m1=1:1:10
```

```
m1 = 1×10
     1     2     3     4     5     6     7     8     9    10
```

Ahora bien, es sencillo entrar a las componentes individuales de la matriz, como también realizar una reasignación (o asignación) de esta:

```
t=m1(2)*m1(3);
disp(t)  %Comando para mostrar cosas por pantalla
```

```
     6
```

```
m1(2)=99;
disp(m1)
```

```
     1    99     3     4     5     6     7     8     9    10
```

```
m1(3,2)=21;
disp(m1)
```

```
     1    99     3     4     5     6     7     8     9    10
     0     0     0     0     0     0     0     0     0     0
     0    21     0     0     0     0     0     0     0     0
```

```
y=m1(:,2)  %Capturar la columna 2 de la matriz m1
```

```
y = 3×1
    99
     0
    21
```

```
z=m1(end,:)  %Utiliza el comando 'end' para indicar la última fila de la matriz m1
```

```
z = 1×10
     0    21     0     0     0     0     0     0     0     0
```

```
cols=[1 3 2];
z=m1(:,cols);
```

```
disp(z)
```

```
     1     3    99
     0     0     0
     0     0    21
```

```
ones(4)    %Crea una matriz cuadrada de unos.
```

```
ans = 4×4
     1     1     1     1
     1     1     1     1
     1     1     1     1
     1     1     1     1
```

```
zeros(2,5) %Crea una matriz de 2x5 con ceros.
```

```
ans = 2×5
     0     0     0     0     0
     0     0     0     0     0
```

```
y=magic(4)   %Crea un cuadrado mágico de 4x4
```

```
y = 4×4
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
```

```
sum(y)    %Sumatoria de una matriz
```

```
ans = 1×4
    34    34    34    34
```

```
sum(y')' %Transpuesta de la suma de la transpuesta de y
```

```
ans = 4×1
    34
    34
    34
    34
```

```
eye(5)  %Matriz identidad de 5x5
```

```
ans = 5×5
     1     0     0     0     0
     0     1     0     0     0
     0     0     1     0     0
     0     0     0     1     0
     0     0     0     0     1
```

Ahora para la creación de matrices con números aleatorios exsiten varios códigos: rand, randi, rng... y sus argumentos, en cada caso sería func([a b],c,d) donde [a b] es el intervalo donde estarán los enteros, y estarán en una matriz de cxd

```
r=randi([-6 6],1,7) %Creación de números aleatorios enteros
```

```
r = 1×7
```

```
     5    -1    -1    -4    -5    -2     3
```

```
tmp=r==0;  %El doble igual indica un operador lógico de equivalencia
disp(tmp)
```

```
   0   0   0   0   0   0   0
```

```
cant_ceros=sum(sum(r==0));    %Contador de ceros en la matriz aleatoria 0
disp(cant_ceros)
```

```
     0
```

```
todos=[sum(sum(r<0)) sum(sum(r==0)) sum(sum(r>0))];   %Contador de los números menores, iguales
disp(todos)
```

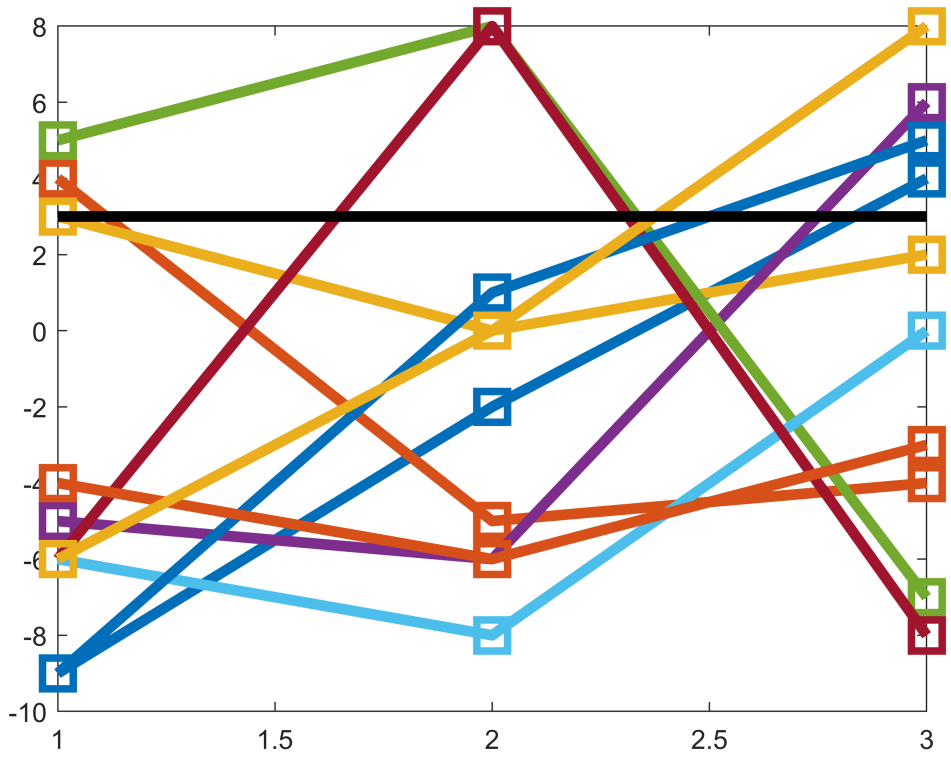
```
     5     0     2
```

MATLAB también soporta matrices en varias dimensiones:

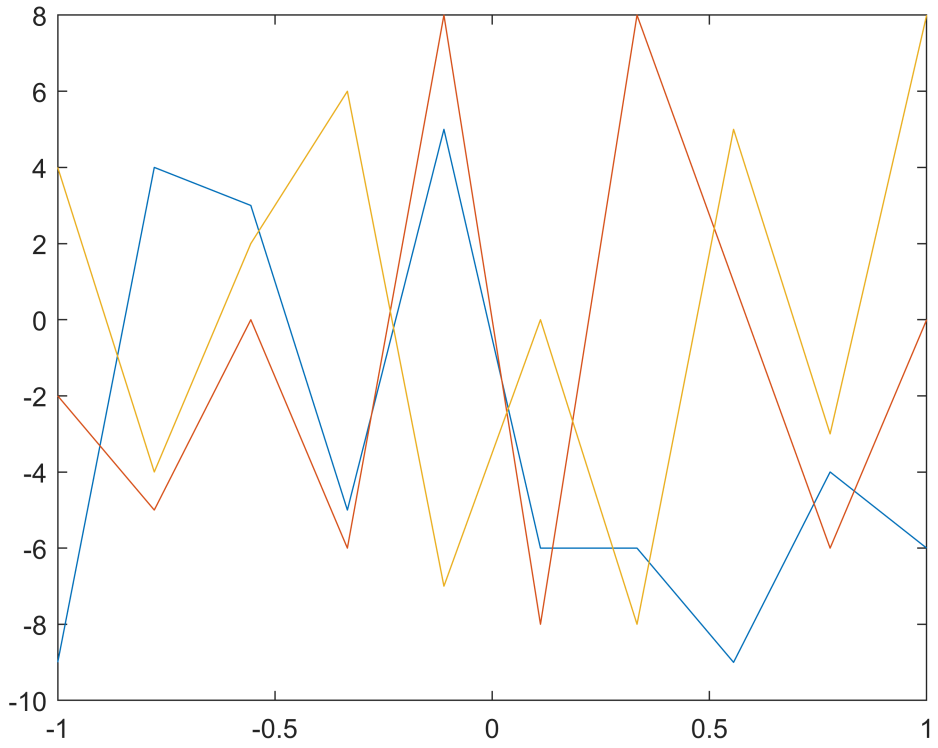
```
r3=randi([-5 5],3,1,2)  %Creación de una matriz de aleatorios de -5 a 5 de tres dimensiones(3x1
```

```
r3 =
r3(:,:,1) =

     3
     2
    -5


r3(:,:,2) =

     4
     5
     3
```

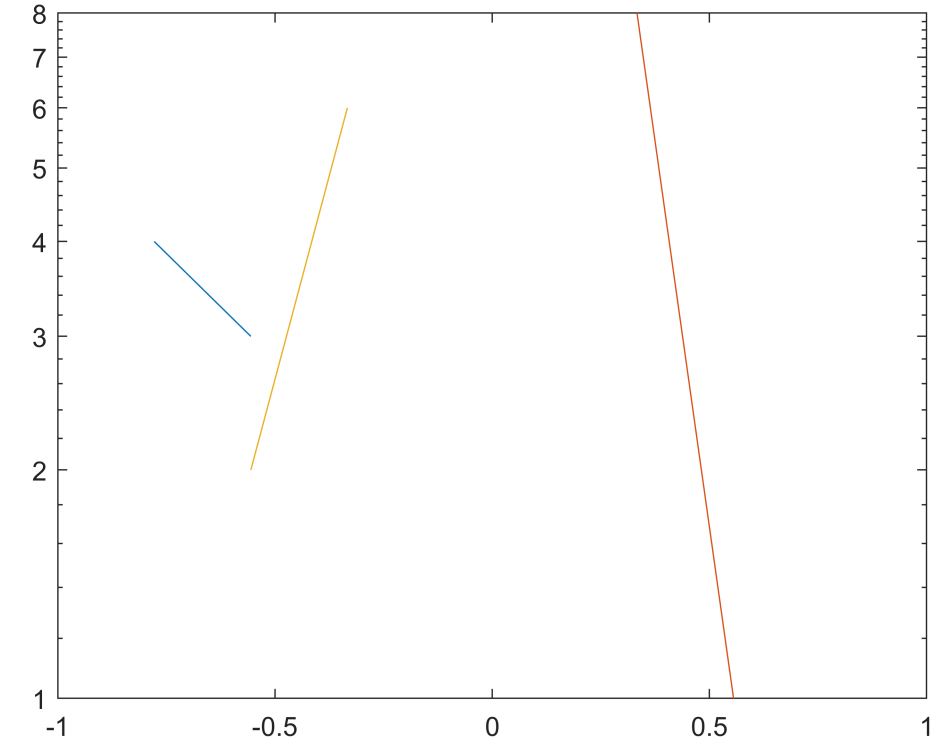
```
rr=randi([-9 9],3,10);
plot(rr,'Linewidth',4,'Marker','square','MarkerSize',15)  %Algunos argumentos del plot
line([1 3],[3 3],'color',[0 0 0], 'Linewidth',4) %Crea una línea en el plot
```




```
x=linspace(-1,1,length(rr));  %Crea un vector uniformemente distribuido de -1 a 1 con length(rr
plot(x,rr')
```




```
semilogy(x,rr')  %Gráfica algoritmica de los datos
```




```
Warning: Negative data ignored
```

```
Warning: Negative data ignored
```