

# PRÁCTICA1. MATLAB


Principios básicos



# 1. INTRODUCCIÓN

Matlab proporciona un entorno potente y amigable para cálculo y simulación. El entorno de programación ofrece operaciones matemáticas básicas más una serie de procedimientos operacionales (conocidos como funciones). Las herramientas de programación abarcan operaciones matemáticas básicas y también un gran conjunto de procedimientos computacionales que se diseñan para tareas específicas. Así, el usuario tiene la opción de desarrollar un programa a medida o de llamar a cualquiera de las funciones de propósito especial que residen en los ficheros de MATLAB. Además, un potente procesador gráfico permite visualizaciones de alta calidad de las variables en diversos formatos. Programando en MATLAB, cada variable se supone que es una matriz y no existe ningún requisito para el dimensionamiento y declaración de variables. Las dimensiones de la matriz se definen mediante una lista explícita de elementos o por reglas que se aplican a las operaciones matemáticas.

## 1.1 Entorno

MATLAB dispone de un entorno gráfico de trabajo al cual se accede en Windows haciendo doble click sobre el icono  (en algunos casos tras la instalación es necesario crear un acceso directo al ejecutable de Matlab situado en la carpeta C:\Program Files\MATLAB\R2015a\bin\matlab.exe)

Este entorno gráfico se divide generalmente en varios paneles como se puede ver en la imagen. estacan de arriba hacia abajo, los siguientes elementos invariantes: **barra de herramientas**, formada por varios iconos de acceso rápido a las opciones más utilizadas del submenú de ficheros y de edición, muchos de ellos inactivos. Incluye, además, diferentes ventanas propias como la **ventana de comandos** (*Command Window*), donde teclearemos las diferentes instrucciones con que daremos órdenes al sistema, el **espacio de trabajo** (*Workspace*) donde se almacenarán las variables y resultados presentes en y el **directorio actual** (*Current Folder*) sobre el que trabajamos. También es posible disponer de un panel en el que aparezca un listado histórico de las últimas instrucciones ejecutadas (*Command History*).

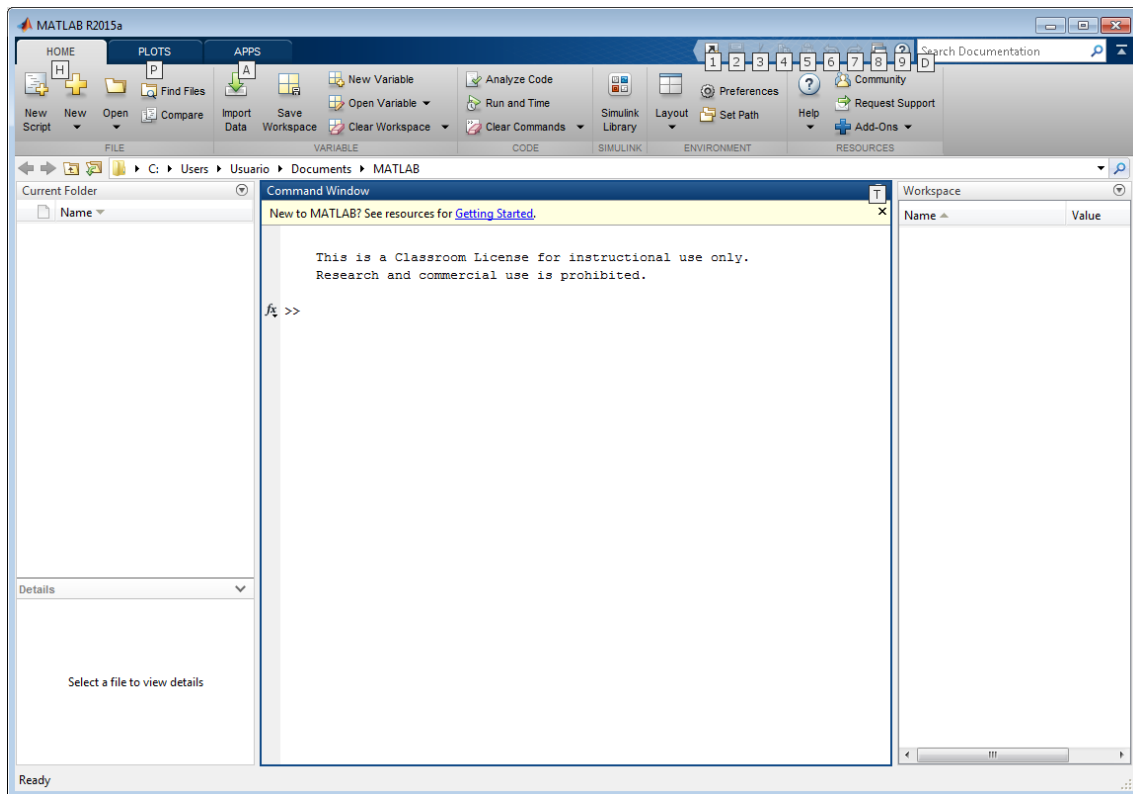


Figura1. Entorno de Matlab

## 1.2 Ayuda

Hay varios niveles de ayuda: desde **demos** hasta ayuda especializada para saber lo que hace una determinada función o comando. Si se quiere saber, por ejemplo, lo que es y hace el comando `lu` se teclea en la ventana de comandos `help lu` o `helpwin lu`. Matlab responde, en la propia ventana de comandos (en el primer caso) y en una ventana aparte (en el segundo), explicando todo lo relativo a dicho comando. Lo anterior implica que se sabe lo que se busca. Hay niveles más generales de ayuda. Por ejemplo, tecleando `helpwin` aparece una relación de las carpetas de MATLAB que contienen ayuda y una breve descripción del contenido de cada carpeta. Pinchando en cada una de ellas se obtiene un listado con los comandos o funciones que hay en dicha carpeta. Si se trabaja en modo gráfico hay, además, una tercera posibilidad de ayuda más general. Pinchando en **Help** en el menu principal de MATLAB aparecerá una nueva ventana con toda la ayuda de lo básico.

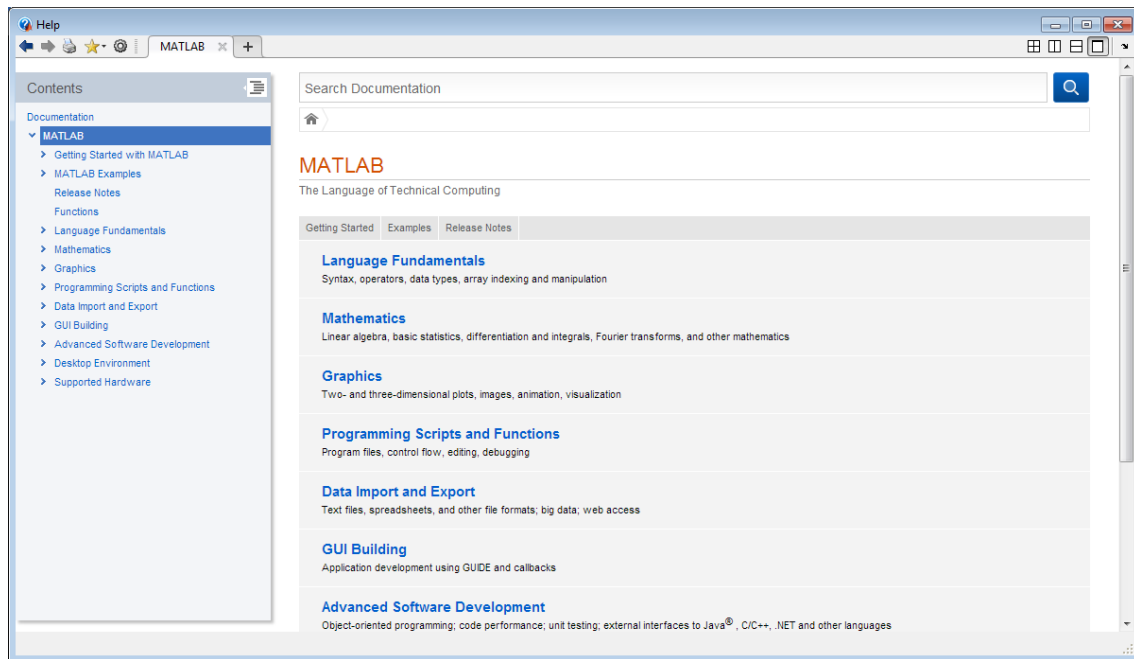


Figura2. Ventana de ayuda de Matlab

### 1.3 Edición en línea de comandos

↑	ctrl-p	Reescribe la línea anterior
↓	ctrl-n	Reescribe la línea siguiente
←	ctrl-b	Mueve el cursor un carácter hacia atrás
→	ctrl-f	Mueve el cursor un carácter hacia adelante
ctrl-→	ctrl-r	Mueve el cursor una palabra a la derecha
ctrl-←	ctrl-l	Mueve el cursor una palabra a la izquierda
Inicio	ctrl-a	Mueve el cursor al comienzo de la línea
Fin	ctrl-e	Mueve el cursor al final de la línea
Esc	ctrl-u	Borra la línea
Supr	ctrl-d	Borra el carácter sobre el cursor
Backspace	ctrl-h	Borra el carácter que está delante del cursor
	ctrl-k	Borra todo desde el cursor hasta el fin de la línea

## 2. OPERACIONES ELEMENTALES Y VARIABLES

Las sentencias de MATLAB están típicamente en el formato general de `variable=expresión` (o simplemente `expresión`), y se devuelve una variable como respuesta a una interpretación de MATLAB de la evaluación de la expresión.

Un ejemplo simple es:

```
y = 10*sin(pi/6)
```

El resultado devuelto es un escalar (matriz de 1 por 1) con un valor de 5,0. además el usuario podrá insertar la variable de salida y en cualquier sentencia que siga.

La forma de representar números y de operar de Matlab es la misma que la de las calculadoras. Por ejemplo:

```
3      -99    .001  9.63  1.62e-020
```

Se usa el punto como separador decimal, en lugar de la coma.

Las operaciones usuales se realizan con los mismos símbolos y en la misma secuencia que en las calculadoras.

+	adición
-	sustracción
*	multiplicación
/	división por la derecha
\	división por la izquierda
^	potenciación
'	transposición

Hay otras operaciones especiales para matrices:

.*	multiplicación término a término
./	división a la derecha término a término
.\	división a la izquierda término a término
potenciación término a término	

Para que matlab ejecute una orden, es preciso pulsar la tecla **Intro**. Por ejemplo, para calcular el valor de  $3 + 5 \times 2 + 1$ , se ejecuta la instrucción

```
>> 3 + 5*2 + 1
```

y se obtiene como respuesta

```
ans =  
14
```

Esto quiere decir que el resultado se ha almacenado en la variable `ans`. En cambio,

```
>> s = (3+5)*2 + 1
```

indica a Matlab que el resultado de esa operación ha de guardarse en la variable `s`.

## 2.1 Reglas para nombrar variables

- El nombre de una variable puede tener como máximo 63 caracteres (31 en versiones anteriores), que pueden ser letras, números y el guión de subrayar
- El primer carácter tiene que ser una letra. `lado2` es un nombre válido, pero no lo es `2lado`.
- Las mayúsculas y las minúsculas tienen valor distintivo. La variable `Base` es distinta de la variable `base`.
- Dentro de un nombre de variable no puede haber espacios en blanco. `lado1` es válido, pero no `lado 1`.
- Existen nombres que deben evitarse, porque tienen significado propio en Matlab: `ans`, `pi`, `Inf`, ...

## Ejercicios

1. Realizar las operaciones siguientes y guardar el resultado en la variable que se indica:

$$s = 3^2 - 5 - 6/3 \times 2. \quad \text{Sol } s = 0$$

$$t = 3^2 - 5 - 6/(3 \times 2). \quad \text{Sol } t = 3$$

$$s + t + 1. \quad \text{Sol: } ans = 4$$

2. Para  $x = 0$ ,  $x = 1/2$ ,  $x = 2^2$  y  $x = 1$ , calcula el valor del cociente  $x / (x^2 - 1)$ .

Sol: `ans = 0`, `ans = 0.6777`, `ans = 0.2666`, `ans = Inf`

## 3. VECTORES Y MATRICES

Los vectores y las matrices son el mismo elemento para Matlab, los vectores se tratan de matrices de una fila. Para generar matrices la forma más sencilla es

escribiendo explícitamente los elementos de la matriz. Hay que tener en cuenta las siguientes reglas:

- Los elementos de la matriz hay que introducirlos fila a fila.
- Los elementos de cada fila deben estar separados por comas o espacios en blanco.
- Para indicar el final de una fila se debe escribir ‘;’
- La lista de todos los elementos debe estar encerrada entre corchetes, []

Debe observarse que el número de elementos en cada fila debe ser el mismo; en caso contrario, Matlab produciría un mensaje de error.

Por ejemplo, para introducir la matriz

$$A = \begin{pmatrix} 16 & 3 & 2 & 13 \\ 5 & 10 & 11 & 8 \\ 9 & 6 & 7 & 12 \\ 4 & 15 & 14 & 1 \end{pmatrix}$$

el código sería

```
>> A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
```

La respuesta de Matlab en este caso sería:

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

En el caso de que la instrucción sea muy larga y no entre en la misma línea se puede separar la instrucción en varias líneas. En ese caso hay que terminar la línea con puntos suspensivos. Por ejemplo:

```
>> A = [16 3 2 13; ...  
        5 10 11 8; ...  
        9 6 7 12; ...  
        4 15 14 1]
```

produce la misma matriz que antes.

En ambos casos la matriz es asignada a la variable A. Debe observarse que con Matlab no hay que declarar las variables. Se podría haber introducido la matriz sin asignársela de forma específica a una variable. Por ejemplo,

```
>> [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
```

en este caso la respuesta de Matlab es:

```
ans =
```

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

### 3.1 Casos particulares

#### Definición de vectores fila.

Los vectores fila son útiles para crear elementos espaciados que pueden ser utilizados para recorrer elementos concretos de una matriz o para la ejecución de bucles. Para la definición de los vectores existen tres posibilidades:

$v=[a:h:b]$  → Define un vector “fila” cuyo primer elemento es a y los demás elementos aumentan de h en h sin superar b.

```
>> y=0:pi/4:pi
```

```
y =
```

```
0 0.7854 1.5708 2.3562 3.1416
```

Es posible utilizar incrementos negativos.

```
>> z=6:-.5:3
```

```
z =
```

```
6.0000 5.5000 5.0000 4.5000 4.0000 3.5000 3.0000
```

$v=[a:b]$  → Define un vector “fila” cuyo primer elemento es a y los demás elementos aumentan de 1 en 1 sin superar b.

```
>> x=1:5
```

```
x =
```

```
1 2 3 4 5
```

$v=\text{linspace}(a,b,n)$  → Define un vector “fila” de n componentes, cuyo primer elemento es a y cuyo último elemento es b, con diferencia constante entre componentes consecutivas. El número de componentes por defecto es 100

```
>> k = linspace(-sqrt(3),pi,5)
```

```
k =
```

```
-1.7321 -0.5136 0.7048 1.9232 3.1416
```



### Ejemplos:

```
>> u=linspace(-4,7,6)
>> v=[-4:2:7], w=[-4:7]
>> v=-4:2:7, w=-4:7 % se puede escribir sin los corchetes
```

### Matrices especiales

Para el caso de matrices características existen funciones específicas de Matlab que permiten simplificar el código:

`zeros(m,n)` genera una matriz de ceros de dimensión  $m \times n$ .

`ones(m,n)` genera una matriz de unos de dimensión  $m \times n$ .

`eye(m,n)` genera una matriz de dimensión  $m \times n$ , cuya diagonal principal son unos, y el resto de los elementos ceros. Cuando las matrices son cuadradas de orden  $n$ , se puede sustituir el argumento  $(m,n)$  por  $(n)$ .

`rand(m,n)` Crea una matriz de tamaño  $m \times n$  con elementos aleatorios distribuidos uniformemente entre 0 y 1

`randn(m,n)` Crea una matriz de tamaño  $m \times n$  con elementos aleatorios elegidos con una distribución normal de media 0 y varianza y desviación standard 1

### Ejemplos:

```
>> zeros(2,3)
>> zeros(2)
>> eye(2,3)
>> eye(3)
```

### Ejercicios

Definir los vectores siguientes:

- |  |                            |
|--|----------------------------|
| 1. $v = (\sqrt{2}, \pi, e)$ .                    | Sol = $[\sqrt{2}, \pi, e]$ |
| 2. $x = (0.15, 0.30, 0.45, \dots, 1.65, 1.80)$ . | Sol = 0.15:0.15:1.80       |
| 3. $y = (3, 4, 5, \dots, 46, 47)$ .              | Sol = 3:47                 |
| 4. $z = (100, 100, 1, 2, 3, \dots, 99, 100)$ .   | Sol = [100, 100, 1:100]    |

Definir las siguientes matrices

1. Matriz de 3x3 de unos Sol = eye(3)
2. Matriz aleatoria de 4x2 Sol = rand(4,2)
3. Matriz de 5x3 con números que vayan del 1 al 15 Sol=[1:5;5:10;11:15]

### 3.2 Manipulación de vectores y matrices

Para poder manipular los elementos de una matriz hay que tener en cuenta que dada una matriz A, se accede a través de los elementos por fila y por columna. Así para poder acceder al elemento de la fila i y la columna j, se denota por A(i,j). Tomando como ejemplo la matriz A el elemento en la posición (2, 4) es 8. Escribiendo

```
>> A(2,4)
```

Matlab devuelve:

```
ans =  
      8
```

Se puede modificar el elemento de cualquier posición sin más que cambiar su valor. La respuesta de Matlab es toda la matriz con el nuevo valor en dicha posición.

```
>> A(2,4)= 3*sqrt(A(1,4))-1/log10(A(3,4))  
A =  
    16.0000    3.0000    2.0000   13.0000  
    5.0000   10.0000   11.0000    9.8900  
    9.0000    6.0000    7.0000   12.0000  
    4.0000   15.0000   14.0000    1.0000
```

En una misma línea de comandos se pueden poner varias órdenes separadas bien sea por comas (en cuyo caso los resultados correspondientes aparecerán sucesivamente en la pantalla) o bien por ';' para que los resultados no aparezcan en la pantalla.

Si pedimos el valor del elemento en la posición (4, 5), Matlab devuelve un mensaje de error porque el tamaño de A es 4 x 4:

```
>> A(4,5)  
??? Index exceeds matrix dimensions.
```

Se puede usar `end` para indicar el último elemento de una matriz, respecto de una dimensión dada. Así:

```
>> a=4:10
```

```

a =
     4  5  6  7  8  9 10
>> a(end)
ans =
     10
>> A(end,end)
ans =
      1
>> A(2,end)
ans =
     15

```

Estos son casos de indexación para elementos aislados de una matriz. En el caso de que queramos extraer bien sea una fila o columna o una submatriz, los elementos se indican por medio de notación vectorial. Para obtener el subvector de *a* formado por los últimos 5 primeros elementos escribiríamos:

```

>> a(1:5)
ans =
     4  5  6  7  8

```

y el formado por los últimos 4:

```

>> a(end-3:end)
ans =
     7  8  9 10

```

Para obtener la submatriz formada por los elementos que ocupan las posiciones donde se interceptan las filas 1 y 3 y las columnas 3 y 4 escribiríamos:

```

>> C=A([1 3],[3 4])
C =
     2 13
     7 12

```

Para obtener la submatriz formada por las dos primeras se utiliza el comando dos puntos:

```

>> A([1 2], :)
ans =
    16  3  2 13
     5 10 11  8

```

Utilizando el comando end:

```
>> A(end-2:end, end-1:end)
ans =
    11     8
     7    12
    14     1
```

Si a una matriz le especificamos un solo subíndice, MATLAB cuenta los elementos por columnas y nos devuelve el elemento correspondiente:

```
>> A(7)
ans =
     6
>> A(12)
ans =
    14
```

Así para obtener una matriz B con las filas primera y segunda, y las columnas segunda y cuarta de A permutadas, haríamos lo siguiente:

```
>> B=A([2 1 3 4],[1 4 3 2])
B =
     5     8    11    10
    16    13     2     3
     9    12     7     6
     4     1    14    15
```

Dada una matriz A podemos añadirle filas y columnas sin más que especificar el vector con los elementos que se quiere añadir. Por ejemplo, si a nuestra matriz A le queremos añadir la fila [1 3 5 7] haríamos lo siguiente:

```
>> C =[A;[1 3 5 7]]
C =
    16     3     2    13
     5    10    11     8
     9     6     7    12
     4    15    14     1
     1     3     5     7
```

Si lo que queremos es añadir una columna actuaríamos por transposición

```
>> [A [1:2:7]']
ans =
    16     3     2    13     1
     5    10    11     8     3
     9     6     7    12     5
     4    15    14     1     7
```

Esta forma de crear matrices a partir de otras más pequeñas nos permite ir creando matrices sobre la marcha empezando desde una matriz vacía.

```
>> B=[]
B =
[]
>> c1=1:3:7; c2=2:3:8; c3=3:3:9; B1=[B c1' c2' c3'],
B2=[B;c1;c2;c3]
B1 = 1 2 3 4 5 6 7 8 9
B2 =
    1     4     7
    2     5     8
    3     6     9
```

También podemos eliminar filas y/o columnas con ayuda de la “matriz vacía”. Simplemente igualamos a [] la submatriz que queramos eliminar:

```
>> B2(:,2)=[]
B2 =
    1     7
    2     8
    3     9
```

nos devuelve la submatriz de B2 que se obtiene al suprimir la segunda columna.

### Ejercicio

Considerar la siguiente matriz:

$$A = \begin{pmatrix} 11 & 12 & 13 & 14 \\ 21 & 22 & 23 & 24 \\ 31 & 32 & 33 & 34 \\ 41 & 42 & 43 & 44 \end{pmatrix}$$

Se pide:

a) Introducir la matriz A.

Sol = A=[11:14;21:24;31:34;41:44]

b) Obtener los valores de la primera columna.

Sol = A(:,1)

c) Obtener los valores de la segunda fila.

Sol = A(2,:)

d) Obtener los valores de la segunda y la tercera columna.

Sol = A(:,2:3)

e) Obtener la diagonal de A.

Sol = A([1:5:15])

### 3.3 Funciones básicas sobre matrices

`length(v)` muestra el número de componentes del vector `v`.

`size(a)` muestra el número de filas y columnas de la matriz `a`.

`min(a)` devuelve el mínimo valor de un vector, si es una matriz devuelve un vector fila con el mínimo de cada columna. `min(a, [], dim)` devuelve el menor de los elementos a lo largo de la dimensión `dim`. Por ejemplo `min(A, [], 2)` es un vector columna con el mínimo de cada fila.

`max(a)` tiene el mismo comportamiento que la función `min`.

`sum(a)` devuelve la suma de los elementos de un array. En el caso de que sea una matriz devuelve la suma por columnas. `sum(a, dim)` devuelve la suma a lo largo de la dimensión especificada.

`diag(a)` devuelve los elementos de la diagonal de `a`, en el caso de que `a` sea un vector crea una matriz con los elementos del vector en la diagonal.

#### Ejercicio

Generar dos matrices aleatorias de 3x3 cada una y realizar el producto elemento a elemento de las matrices y el producto matricial y obtener el elemento máximo de cada resultado.

Sol = A=rand(3), B=rand(3), A\*B, A.\*B

# Problemas

## 1. Matrices I

Almacenar la variable *Matriz1* la siguiente matriz

$$A = \begin{pmatrix} 1 & -3 & 5 \\ 9 & 3 & 1 \\ 2 & -1 & 4 \end{pmatrix}$$

- Calcular su transpuesta y guardarla en *Matriz2*
- Calcular el producto elemento a elemento entre *Matriz1* y *Matriz2*
- Calcular la suma de *Matriz1* y el producto anterior y almacenarlo en *Matriz3*
- Calcular el producto matricial entre *Matriz1* y *Matriz2* y almacenarlo en *prodM1M2*
- Calcular el producto matricial entre *Matriz2* y *Matriz1* y almacenarlo en *prodM2M1*
- Cambiar el valor del elemento central de *Matriz1* a 9
- Guardar en una matriz llamada esquinas de tamaño 2x2 los elementos de las esquinas de *Matriz1*
- Guardar en un vector fila *diagonalM1* los elementos de la diagonal principal de *Matriz1*
- Guardar en un vector columna *diagonalM2* los elementos de la primera fila de la *Matriz1*
- Calcular el producto elemento a elemento de *diagonalM1* y *diagonalM2*
- Calcular el producto matricial de *diagonalM1* y *diagonalM2*

## 2. Matrices II

Definir la siguiente matriz

$$A = \begin{pmatrix} 2 & 2 & 1 \\ 3 & 4 & 0 \\ 1 & 5 & 4 \end{pmatrix}$$

- Calcular el máximo de cada fila y el máximo de la matriz
- Calcular el mínimo de cada columna y el mínimo de la matriz
- Calcular el sumatorio de los elementos de la primera columna y de la tercera

### 3. Generación de vectores

- Generar un vector de 10 elementos que cubra el intervalo  $[-\pi, e]$ .
- Generar sin utilizar ninguna función un vector de 500 puntos para el intervalo  $[0, 10]$ .