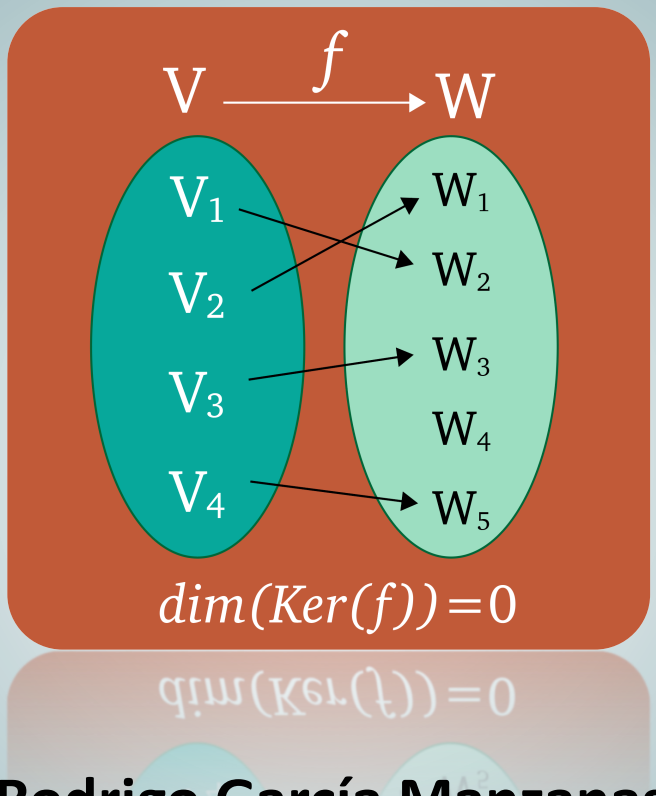
# Álgebra

## Práctica 1. Introducción a MATLAB




**Rodrigo García Manzanas**
**Neila Campos González**
**Ana Casanueva Vicente**

Departamento de Matemática Aplicada y Ciencias de la Computación

Este tema se publica bajo Licencia:
Creative Commons BY-NC-SA 4.0

*Grado en Ingeniería Química*

## G320: Álgebra

# Práctica 1: Introducción a MATLAB

*Rodrigo García Manzanas (rodrigo.manzanas@unican.es)*

### Objetivos

- Familiarizarse con el entorno MATLAB
- Aprender a definir matrices y vectores

### Introducción

El nombre de MATLAB es un acrónimo de MATrix LABoratory. Hoy en día MATLAB es un programa muy potente con un entorno agradable, que incluye herramientas de visualización gráfica, así como un lenguaje de alto nivel. La interfaz principal de MATLAB se divide en varias partes:

- Las órdenes a ejecutar se escriben en el *Command Window* (ventana de comandos)
- En el editor (ventana *Editor*) es donde escribimos código que guardaremos en scripts con extensión *.m*, para su posterior ejecución desde el Command Window. El color que aparece en la barra lateral a la derecha del script nos indica si la sintáxis es correcta (verde), es correcta pero podría llegar a dar problemas (naranja) o es incorrecta (rojo). Los comentarios (texto no ejecutable) se introducen con el caracter *%*
- La ventana *Workspace* (espacio de trabajo) proporciona información sobre las variables que están almacenadas en memoria.

Hay ciertas reglas para definir variables en MATLAB:

- El nombre de una variable puede tener como máximo 63 caracteres que pueden ser letras, números y el guión bajo.
- El primer carácter tiene que ser una letra. Por ejemplo, *matriz2* es un nombre válido, *2matriz* no lo es.
- Las mayúsculas y las minúsculas tienen valor distintivo. La variable *X* es distinta de la variable *x*.
- Dentro de un nombre de variable no puede haber espacios en blanco. *modulo1* es un nombre de variable válido, *modulo 1* no.
- Existen nombres que deben evitarse porque tienen significado propio en Matlab: *ans*, *pi*, *Inf*, *i*, . . .

A tener en cuenta:

- Al abrir sesión seleccionaremos el directorio de trabajo, donde almacenaremos los scripts
- Si se escribe *;* al final de una linea de código, dicha línea es ejecutada, pero no se muestra por pantalla el resultado de la ejecución
- Los comandos **más útiles** de MATLAB *help, doc y lookfor*, que sirven para obtener ayuda sobre el resto de comandos/funciones. La documentación online de MATLAB (accesible mediante *doc*) es realmente buena

```
help diag
```

```
diag - Create diagonal matrix or get diagonal elements of matrix

    This MATLAB function returns a square diagonal matrix with the elements of
    vector v on the main diagonal.

    D = diag(v)
    D = diag(v,k)
    x = diag(A)
    x = diag(A,k)

    See also blkdiag, isdiag, istril, istriu, spdiags, tril, triu

    Reference page for diag
    Other functions named diag
```

```
lookfor diagonal
```

```
balance                         - Diagonal scaling to improve eigenvalue accuracy.
cdf2rdf                         - Complex diagonal form to real block diagonal form.
isdiag                          - Determine whether a matrix is diagonal.
rsf2csf                         - Real block diagonal form to complex diagonal form.
trace                           - Sum of diagonal elements.
sqrtm_tbt                       - Square root of 2x2 matrix from block diagonal of Schur form.
blkdiag                         - Block diagonal concatenation of matrix input arguments.
diag                            - Diagonal matrices and diagonals of a matrix.
lesp                            - Tridiagonal matrix with real, sensitive eigenvalues.
poisson                         - Block tridiagonal matrix from Poisson's equation (sparse).
toeppen                         - Pentadiagonal Toeplitz matrix (sparse).
tridiag                         - Tridiagonal matrix (sparse).
spdiags                         - Sparse matrix formed from diagonals.
diagrep                         - Replicates model along the diagonal.
diagrep                         - Replicates model along the diagonal.
diagrep                         - Replicates model along the diagonal.
diag                            - Turns Nx1 FRD into NxN diagonal, and NxN FRD into Nx1
diagrep                         - Replicates model along the diagonal.
diagrep                         - Replicates model along the diagonal.
blkdiag                         - Block-diagonal concatenation of input/output models.
bdschur                         - Block-diagonal Schur factorization.
diag                            - Diagonal matrices or diagonals of matrix
diag                            - Create or extract symbolic diagonals.
hFindDiagElementsInLocalPart    - Return the local linear indices of the diagonal
hFindDiagElementsInLocalPart    - Return the local linear indices of the diagonal
diag                            - Diagonal matrices and diagonals of a codistributed matrix
isdiag                          - Determine whether a codistributed matrix is diagonal.
spdiags                         - Sparse matrix formed from diagonals.
diagSolve                       - Diagonal solver for codistributed matrices
reciprocalConditionNumberFromDiagonal - Calculate recirpocal condition number (rCond) for triangular or di
cdf2rdf                         - Complex diagonal form to real block diagonal form.
diag                            - Diagonal matrices and diagonals of a gpuArray matrix
```

```
isdiag                          - Determine whether a gpuArray matrix is diagonal.
spdiags                         - Sparse matrix formed from diagonals.
tridieig                        - Find a few eigenvalues of a tridiagonal matrix.
tridisolve                      - Solve A*x = b where A is a square, symmetric tridiagonal matrix.
bkbrk                           - Part(s) of an almost block-diagonal matrix.
slvblk                          - Solve almost block-diagonal linear system.
ssbal                           - Balancing of state-space model using diagonal similarity.
blkbuild                        - Builds a block-diagonal state-space structure from a block
getHistogramAxes                - This function adds a set of axes along the diagonals for the histograms.
```

### Operaciones básicas

La forma de operar con MATLAB es igual que con una calculadora de bolsillo, usando los símbolos +, -, *, /, ^

- Suma y resta

```
2+4
```

```
ans =
     6
```

```
2-4
```

```
ans =
    -2
```

- Multiplicación y división

```
2*pi
```

```
ans =
   710/113
```

```
pi/10
```

```
ans =
    71/226
```

- Potenciación

```
3^2
```

```
ans =
     9
```

- Otros operadores importantes

```
sqrt(4)  % raíz cuadrada
```

```
ans =
     2
```

```
log(1)  % logaritmo
```

```
ans =
     0
```

```
exp(1)  % exponencial
```

```
ans =
    1457/536
```

```
sin(pi/2)  % seno
```

```
ans =
       1
```

```
cos(pi/2)  % coseno
```

```
ans =
       1/16331239353195370
```

```
tan(pi/2)  % tangente
```

```
ans =
16331239353195370
```

- Formatos

```
format short  % o simplemente format
1/3
```

```
ans = 0.3333
```

```
format long
1/3
```

```
ans =
   0.333333333333333
```

```
format rat
1/3
```

```
ans =
       1/3
```

### Creación de vectores y matrices

Hay varias maneras de crear un vector fila

```
a = 1:2:10
```

```
a = 1×5
       1             3             5
```

```
a = [1 4 9]
```

```
a = 1×3
       1             4
```

```
a = [1, 4, 9]  % podemos dejar un espacio en blanco entre números o poner una coma
```

```
a = 1×3
       1             4
```

Para crear un vector columna

```
b = [-1; 2; 3]
```

```
b = 3×1
     -1
      2
      3
```

```
b = [1 2 3]'  % la comilla al final del vector le da la vuelta
```

```
b = 3×1
      1
      2
      3
```

Para crear una matriz combinamos la definición de vector fila y vector columna. Por ejemplo

```
M = [1 2 3; 4 5 6]  % matriz de 2 filas y 3 columnas
```

```
M = 2×3
      1            2
      4            5
```

```
A = [1 2 1; 2 4 3; 3 5 2]  % matriz de 3 filas y 3 columnas
```

```
A = 3×3
      1            2
      2            4
      3            5
```

Se pueden seleccionar elementos de una matriz indicando, entre paréntesis, la posición de la fila y la columna

```
A(1, 3)  % elemento de la primera fila y tercera columna
```

```
ans =
      1
```

Para extraer filas o columnas enteras se utiliza el símbolo *:*

```
A(2, :)  % fila 2
```

```
ans = 1×3
      2            4
```

```
A(:, 1)  % columna 1
```

```
ans = 3×1
      1
      2
      3
```

```
A(:, 1:3) % columnas 1, 2 y 3
```

```
ans = 3×3
      1            2
      2            4
      3            5
```

```
A(:, [1, 3]) % columnas 1 y 3
```

```
ans = 3×2
       1
       2
       3
```

Para crear una matriz de ceros

```
mceros = zeros(2,3)  % matriz con 2 filas y 3 columnas de ceros
```

```
mceros = 2×3
       0             0
       0             0
```

Para crear una matriz de unos

```
munos = ones(2,3)  % matriz con 2 filas y 3 columnas de unos
```

```
munos = 2×3
       1             1
       1             1
```

Para crear la matriz identidad

```
I5 = eye(5)  % matriz identidad de orden 5
```

```
I5 = 5×5
       1             0             0
       0             1             0
       0             0             1
       0             0             0
       0             0             0
```

Para crear una matriz de números aleatorios

```
maleatoria = rand(3,3)  % matriz 3x3 de números aleatorios (entre el 0 y el 1)
```

```
maleatoria = 3×3
     687/712        581/607         6
     589/3737       614/1265        4
    6271/6461      1142/1427       10
```

Para extraer la diagonal principal de una matriz

```
d = diag(maleatoria)
```

```
d = 3×1
     687/712
     614/126
    1065/116
```

Para transponer (intercambiar, ordenadamente, filas por columnas) una matriz

```
M'  % transpuesta de M
```

```
ans = 3×2
       1
       2
```

```
     3
```

Para saber cuál es el orden de una matriz

```
size(M)  % orden de M
```

```
ans = 1×2
     2
```

```
size(M')  % orden de la transpuesta de M
```

```
ans = 1×2
     3
```

## Ejercicios propuestos

### Ejercicio 1:

Sea la matriz $A = \begin{bmatrix} 1 & 3 & 2 & -1 \\ 0 & 0 & 2 & 5 \\ -2 & 0 & 0 & 3 \\ 1 & 1 & 1 & -1 \end{bmatrix}$

- Sustituye el elemento $a_{2,3}$ por un -3 y el $a_{1,2}$ por un 7, y llama *B* a la matriz resultante
- Extrae en un vector la primera fila de *B*, y en otro vector la segunda columna de *B*
- Extrae en una matriz *C* las filas 1 y 3 de *B* y en otra matriz *D* las columnas 2 y 4
- Añade la matriz identidad 4x4 ($I_{4\times4}$) a la derecha de *B*, y llama a la matriz resultante $B^*$. ¿Tienes algún problema?
- Añade la matriz identidad 4x4 ($I_{4\times4}$) debajo de *B*, y dale el nombre que quieras a la matriz resultante
- Crea un vector cuyos elementos sean los términos de la diagonal principal de *A*
- Crea una matriz 4x4 de números aleatorios entre 0 y 1 y súmasela a la matriz *A*

### Ejercicio 2:

Crea las siguientes matrices:

- Matriz 4x4 con todos sus elementos cero
- Matriz 4x4 con todos sus elementos iguales a 0.5
- Matriz 3x5 con todos sus elementos iguales a π