Inducción a MATLAB
Álgebra Lineal

Escuela de Matemáticas

Facultad de Ciencias
Universidad Nacional de Colombia
Sede Medellín

¿Qué es MATLAB?

MATLAB (acrónimo de **MAT**rix **LAB**oratory, "laboratorio de matrices") es un *ambiente computacional numérico* y es un *lenguaje de programación*.

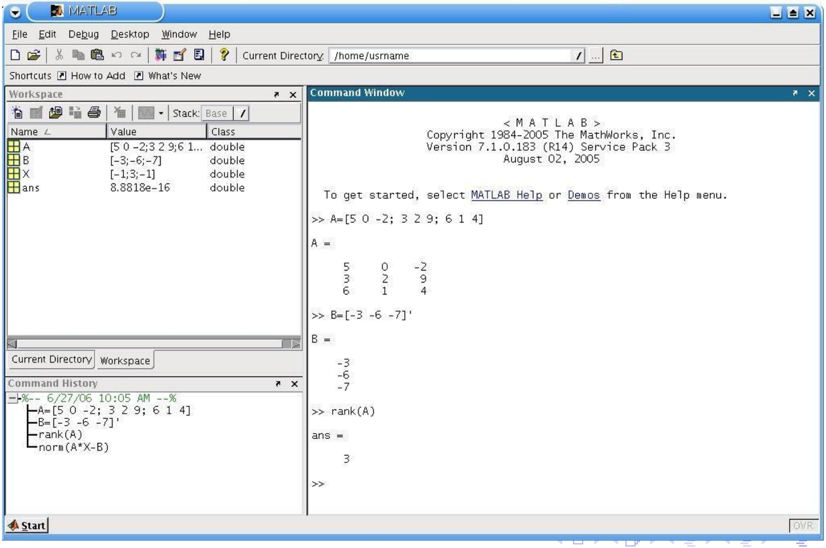
- Proporciona varias formas convenientes para crear y manipular vectores, matrices y arreglos multidimensionales.
- Permite resolver problemas que se modelan mediante matrices:
  - Sistemas de ecuaciones lineales ($A\mathbf{x} = \mathbf{b}$),
  - Factorización de matrices,
  - Cálculo de valores y vectores propios,
  - Sistemas de ecuaciones diferenciales lineales, etc.

1984 MathWorks lanzó al mercado MATLAB 1.0.

2004 MathWorks afirmó que MATLAB fue usado por más de un millón de personas en la industria y el mundo académico.

Abriendo MATLAB

Al iniciar el programa, aparece en pantalla la siguiente ventana:



Formatos de salida

- La instrucción

```
format
```

controla el formato de salida de los valores numéricos presentados en la Ventana de Comandos (Command Window).

Formatos de salida

- La instrucción

  `format`

  controla el formato de salida de los valores numéricos presentados en la Ventana de Comandos (Command Window).

- Hay tres posibilidades para esta instrucción:

Formatos de salida

- La instrucción

`format`

controla el formato de salida de los valores numéricos presentados en la Ventana de Comandos (Command Window).

- Hay tres posibilidades para esta instrucción:
  - `format short` (muestra 5 dígitos decimales).

Formatos de salida

- La instrucción

  `format`

  controla el formato de salida de los valores numéricos presentados en la Ventana de Comandos (Command Window).

- Hay tres posibilidades para esta instrucción:
  - `format short` (muestra 5 dígitos decimales).
  - `format long` (muestra 15 dígitos decimales).

Formatos de salida

- La instrucción

  `format`

  controla el formato de salida de los valores numéricos presentados en la Ventana de Comandos (Command Window).

- Hay tres posibilidades para esta instrucción:
  - `format short` (muestra 5 dígitos decimales).
  - `format long` (muestra 15 dígitos decimales).
  - `format rat` (muestra un cociente de enteros).

Formatos de salida

- La instrucción

  `format`

  controla el formato de salida de los valores numéricos presentados en la Ventana de Comandos (`Command Window`).

- Hay tres posibilidades para esta instrucción:
  - `format short` (muestra 5 dígitos decimales).
  - `format long` (muestra 15 dígitos decimales).
  - `format rat` (muestra un cociente de enteros).
- **Ejemplo**. Calculemos $\sqrt{2}$ en formato corto y largo.

Formatos de salida

- La instrucción

  `format`

  controla el formato de salida de los valores numéricos presentados en la Ventana de Comandos (Command Window).

- Hay tres posibilidades para esta instrucción:
  - `format short` (muestra 5 dígitos decimales).
  - `format long` (muestra 15 dígitos decimales).
  - `format rat` (muestra un cociente de enteros).

- **Ejemplo**. Calculemos $\sqrt{2}$ en formato corto y largo.

```
>> format short
>> sqrt(2)
```

Formatos de salida

- La instrucción

  `format`

  controla el formato de salida de los valores numéricos presentados en la Ventana de Comandos (Command Window).

- Hay tres posibilidades para esta instrucción:
  - `format short` (muestra 5 dígitos decimales).
  - `format long` (muestra 15 dígitos decimales).
  - `format rat` (muestra un cociente de enteros).

- **Ejemplo**. Calculemos $\sqrt{2}$ en formato corto y largo.

```
>> format short
>> sqrt(2)
 ans   =
       1.4142
```

Formatos de salida

- La instrucción

  `format`

  controla el formato de salida de los valores numéricos presentados en la Ventana de Comandos (Command Window).

- Hay tres posibilidades para esta instrucción:
  - `format short` (muestra 5 dígitos decimales).
  - `format long` (muestra 15 dígitos decimales).
  - `format rat` (muestra un cociente de enteros).

- **Ejemplo**. Calculemos $\sqrt{2}$ en formato corto y largo.

```
>> format short
>> sqrt(2)
 ans   =
       1.4142

>> format long
>> sqrt(2)
```

Formatos de salida

- La instrucción

  `format`

  controla el formato de salida de los valores numéricos presentados en la Ventana de Comandos (Command Window).

- Hay tres posibilidades para esta instrucción:
  - `format short` (muestra 5 dígitos decimales).
  - `format long` (muestra 15 dígitos decimales).
  - `format rat` (muestra un cociente de enteros).

- **Ejemplo**. Calculemos $\sqrt{2}$ en formato corto y largo.

```
>> format short
>> sqrt(2)
 ans   =
       1.4142

>> format long
>> sqrt(2)
 ans   =
       1.41421356237310
```

Ingresando vectores

Alt + 91 = [Alt + 93 =]   Alt + 39 = '

Ingresando vectores

$$\text{Alt} + 91 = [\qquad \text{Alt} + 93 =] \qquad \text{Alt} + 39 = \text{'}$$

- Para entrar $\mathbf{u} \in \mathbb{R}^n$ como *vector fila*, se digita cada componente entre **corchetes**, separadas por un espacio en blanco:

Ingresando vectores

$\text{Alt} + 91 = [\qquad \text{Alt} + 93 =]\qquad \text{Alt} + 39 = '$

- Para entrar $\mathbf{u} \in \mathbb{R}^n$ como *vector fila*, se digita cada componente entre **corchetes**, separadas por un espacio en blanco:

```
>> u = [1 3 -8]
```

Ingresando vectores

$$\text{Alt} + 91 = [\qquad \text{Alt} + 93 =] \qquad \text{Alt} + 39 = '$$

- Para entrar $\mathbf{u} \in \mathbb{R}^n$ como *vector fila*, se digita cada componente entre **corchetes**, separadas por un espacio en blanco:

```
>> u = [1 3 -8]
```

o bien separadas por **comas**:

Ingresando vectores

$\mathsf{Alt} + 91 = [$ $\qquad$ $\mathsf{Alt} + 93 =]$ $\qquad$ $\mathsf{Alt} + 39 = '$

- Para entrar $\mathbf{u} \in \mathbb{R}^n$ como *vector fila*, se digita cada componente entre **corchetes**, separadas por un espacio en blanco:

```
>> u = [1 3 -8]
```

o bien separadas por **comas**:

```
>> v = [5, 4, 7]
```

Ingresando vectores

$\mathsf{Alt} + 91 = [$ $\mathsf{Alt} + 93 =]$ $\mathsf{Alt} + 39 = '$

- Para entrar $\mathbf{u} \in \mathbb{R}^n$ como *vector fila*, se digita cada componente entre **corchetes**, separadas por un espacio en blanco:

```
>> u = [1 3 -8]
```

  o bien separadas por **comas**:

```
>> v = [5, 4, 7]
```

- Si se separan mediante **punto y coma**, se crea un *vector columna*:

Ingresando vectores

Alt + 91 = [Alt + 93 =] Alt + 39 = '

- Para entrar $\mathbf{u} \in \mathbb{R}^n$ como *vector fila*, se digita cada componente entre **corchetes**, separadas por un espacio en blanco:

```
>> u = [1 3 -8]
```

o bien separadas por **comas**:

```
>> v = [5, 4, 7]
```

- Si se separan mediante **punto y coma**, se crea un *vector columna*:

```
>> x = [1; 3; -8; 0; 5]
```

Ingresando vectores

$\mathsf{Alt} + 91 = [\qquad \mathsf{Alt} + 93 =]\qquad \mathsf{Alt} + 39 = '$

- Para entrar $\mathbf{u} \in \mathbb{R}^n$ como *vector fila*, se digita cada componente entre **corchetes**, separadas por un espacio en blanco:

```
>> u = [1 3 -8]
```

  o bien separadas por **comas**:

```
>> v = [5, 4, 7]
```

- Si se separan mediante **punto y coma**, se crea un *vector columna*:

```
>> x = [1; 3; -8; 0; 5]
```

  Otra forma: ingresélo primero como fila y luego transponga

Ingresando vectores

Alt + 91 = [Alt + 93 =] Alt + 39 = '

- Para entrar $\mathbf{u} \in \mathbb{R}^n$ como *vector fila*, se digita cada componente entre **corchetes**, separadas por un espacio en blanco:

```
>> u = [1 3 -8]
```

o bien separadas por **comas**:

```
>> v = [5, 4, 7]
```

- Si se separan mediante **punto y coma**, se crea un *vector columna*:

```
>> x = [1; 3; -8; 0; 5]
```

Otra forma: ingreséla primero como fila y luego transponga

```
>> w = [2 7 -8 1/3 12]'
```

Ingresando vectores

Alt + 91 = [Alt + 93 =] Alt + 39 = '

- Para entrar $\mathbf{u} \in \mathbb{R}^n$ como *vector fila*, se digita cada componente entre **corchetes**, separadas por un espacio en blanco:

```
>> u = [1 3 -8]
```

o bien separadas por **comas**:

```
>> v = [5, 4, 7]
```

- Si se separan mediante **punto y coma**, se crea un *vector columna*:

```
>> x = [1; 3; -8; 0; 5]
```

Otra forma: ingreséla primero como fila y luego transponga

```
>> w = [2 7 -8 1/3 12]'
```

Ingresando matrices

- Entrar una matriz es tan fácil como crear un vector.
  - Separamos cada fila usando **punto y coma** (;).
  - Todas las filas deben tener el mismo número de componentes.

Ingresando matrices

- Entrar una matriz es tan fácil como crear un vector.
  - Separamos cada fila usando **punto y coma** (;).
  - Todas las filas deben tener el mismo número de componentes.

- **Ejemplo**. Para entrar la matriz A de tamaño 3×5

$$A = \begin{bmatrix} 1 & -1 & 5 & 8 & 0 \\ 8 & 5 & 7 & -3 & 9 \\ 7 & 6 & -3 & 1 & 0 \end{bmatrix}$$

Ingresando matrices

- Entrar una matriz es tan fácil como crear un vector.
  - Separamos cada fila usando **punto y coma** (;).
  - Todas las filas deben tener el mismo número de componentes.

- **Ejemplo**. Para entrar la matriz A de tamaño 3×5

$$A = \begin{bmatrix} 1 & -1 & 5 & 8 & 0 \\ 8 & 5 & 7 & -3 & 9 \\ 7 & 6 & -3 & 1 & 0 \end{bmatrix}$$

usamos la instrucción

```
>> A = [ 1 -1 5 8 0; 8 5 7 -3 9; 7 6 -3 1 0 ]
```

Ingresando matrices

- Entrar una matriz es tan fácil como crear un vector.
  - Separamos cada fila usando **punto y coma** (;).
  - Todas las filas deben tener el mismo número de componentes.

- **Ejemplo**. Para entrar la matriz A de tamaño 3×5

$$A = \begin{bmatrix} 1 & -1 & 5 & 8 & 0 \\ 8 & 5 & 7 & -3 & 9 \\ 7 & 6 & -3 & 1 & 0 \end{bmatrix}$$

usamos la instrucción

```
>> A = [ 1 -1 5 8 0; 8 5 7 -3 9; 7 6 -3 1 0 ]
```

que tiene como resultado:

```
A =
     1    -1     5     8     0
     8     5     7    -3     9
     7     6    -3     1     0
```

Operando con vectores

Instrucción	**Acción**
`>> u + v`	calcula la suma vectorial $\mathbf{u} + \mathbf{v}$.
`>> c*v`	calcula el producto escalar $c\mathbf{v}$.
`>> norm(v)`	calcula la norma $\|\mathbf{v}\|$ de $\mathbf{v}$.
`>> v/norm(v)`	normaliza $\mathbf{v}$.
`>> dot(u,v)`	calcula el producto punto $\mathbf{u} \cdot \mathbf{v}$.
`>> v'`	genera $\mathbf{v}^T$.
`>> acos(c)`	calcula $\cos^{-1}(c)$.

Operando con matrices

Instrucción	Acción
`>> A + B`	calcula la suma matricial $A + B$.
`>> c*A`	calcula el producto escalar cA.
`>> A*B`	calcula el producto matricial AB, donde A es $m \times n$ y B es $n \times p$.
`>> A*x`	calcula el producto $A\mathbf{x}$, donde $\mathbf{x}$ es un vector columna $n \times 1$.
`>> u*A`	calcula el producto $\mathbf{u}A$, donde $\mathbf{u}$ es un vector fila $1 \times m$.

Matrices especiales

`>> eye(n)` genera la matriz identidad I_n de tamaño $n \times n$.

Matrices especiales

`>> eye(n)` genera la matriz identidad I_n de tamaño $n \times n$.

`>> zeros(m,n)` genera la matriz cero O de tamaño $m \times n$.

Matrices especiales

`>> eye(n)`	genera la matriz identidad I_n de tamaño $n \times n$.
`>> zeros(m,n)`	genera la matriz cero O de tamaño $m \times n$.

Ejemplo.

```
>> eye(4)            ans =

                          1   0   0   0
                          0   1   0   0
                          0   0   1   0
                          0   0   0   1


>> zeros(2,5)        ans =

                          0   0   0   0   0
                          0   0   0   0   0


>> zeros(2)          ans =

                          0   0
                          0   0
```

Matrices asociadas a A

`>> A'` genera la transpuesta A^T de A

Matrices asociadas a A

`>> A'` genera la transpuesta A^T de A

`>> A^k` calcula A^k, si A es una matriz cuadrada $n \times n$. Alt + 94 = ^

Matrices asociadas a A

`>> A'` genera la transpuesta A^T de A

`>> A^k` calcula A^k, si A es una matriz <u>cuadrada</u> $n \times n$. Alt + 94 = ^

`>> inv(A)` calcula la inversa A^{-1} de A si A es <u>invertible</u>

Matrices asociadas a A

`>> A'`	genera la transpuesta A^T de A
`>> A^k`	calcula A^k, si A es una matriz <u>cuadrada</u> $n \times n$.
`>> inv(A)`	calcula la inversa A^{-1} de A si A es <u>invertible</u>

Alt + 94 = ^

Ejemplo.

```
>> B = [2 1 0 1; 1 1 -1 0; 2 3 -2 -1; 3 0 -1 0]
```

Matrices asociadas a A

`>> A'`	genera la transpuesta A^T de A
`>> A^k`	calcula A^k, si A es una matriz cuadrada $n \times n$.
`>> inv(A)`	calcula la inversa A^{-1} de A si A es invertible

Alt $+$ 94 $=$ ^

Ejemplo.

```
>> B = [2 1 0 1; 1 1 -1 0; 2 3 -2 -1; 3 0 -1 0]

>> inv(B)        ans =
                        1/6   -2/3    1/6    1/3
                        1/3   -1/3    1/3   -1/3
                        1/2   -2      1/2    0
                        1/3    5/3   -2/3   -1/3

>> B^6           ans =
                        711    215   -109    232
                        171     39      3     96
                        207     35     35    160
                        522    130    -14    260
```

Eliminación Gauss-Jordan

- Generadas dos matrices A y B de tamaño $m \times n$ y $m \times r$, la instrucción

```
>> M = [A B]
```

  crea la *matriz aumentada* $[A \mid B]$ de tamaño $m \times (n + r)$.

Eliminación Gauss-Jordan

- Generadas dos matrices A y B de tamaño $m \times n$ y $m \times r$, la instrucción

```
>> M = [A B]
```

  crea la *matriz aumentada* $[A \mid B]$ de tamaño $m \times (n + r)$.

- `>> R = rref(A)` calcula la *forma escalonada reducida* de A.

Eliminación Gauss-Jordan

- Generadas dos matrices A y B de tamaño $m \times n$ y $m \times r$, la instrucción

```
>> M = [A B]
```

  crea la *matriz aumentada* $[A \mid B]$ de tamaño $m \times (n + r)$.

- `>> R = rref(A)` calcula la *forma escalonada reducida* de A.

- Así, para solucionar el sistema lineal $A\mathbf{x} = \mathbf{b}$ mediante *eliminación de Gauss-Jordan*,

Eliminación Gauss-Jordan

- Generadas dos matrices A y B de tamaño $m \times n$ y $m \times r$, la instrucción

```
>> M = [A B]
```

  crea la *matriz aumentada* $[A \mid B]$ de tamaño $m \times (n + r)$.

- `>> R = rref(A)` calcula la *forma escalonada reducida* de A.

- Así, para solucionar el sistema lineal $A\mathbf{x} = \mathbf{b}$ mediante *eliminación de Gauss-Jordan*,

  1. se crea la matriz aumentada $M = [A \mid \mathbf{b}]$:

```
>> M = [A b]
```

Eliminación Gauss-Jordan

- Generadas dos matrices A y B de tamaño $m \times n$ y $m \times r$, la instrucción

```
>> M = [A B]
```

  crea la *matriz aumentada* $[A \mid B]$ de tamaño $m \times (n + r)$.

- `>> R = rref(A)` calcula la *forma escalonada reducida* de A.

- Así, para solucionar el sistema lineal $A\mathbf{x} = \mathbf{b}$ mediante *eliminación de Gauss-Jordan*,
  1. se crea la matriz aumentada $M = [A \mid \mathbf{b}]$:

     ```
     >> M = [A b]
     ```

  2. y se lleva a su forma escalonada reducida:

     ```
     U = rref(M)
     ```

Eliminación Gauss-Jordan

- Generadas dos matrices A y B de tamaño $m \times n$ y $m \times r$, la instrucción

```
>> M = [A B]
```

  crea la *matriz aumentada* $[A \mid B]$ de tamaño $m \times (n + r)$.

- `>> R = rref(A)` calcula la *forma escalonada reducida* de A.

- Así, para solucionar el sistema lineal $A\mathbf{x} = \mathbf{b}$ mediante *eliminación de Gauss-Jordan*,

  1. se crea la matriz aumentada $M = [A \mid \mathbf{b}]$:

```
>> M = [A b]
```

  2. y se lleva a su forma escalonada reducida:

```
U = rref(M)
```

  - O se aplica una única instrucción

```
U = rref([A b])
```

¿Cómo hallar bases para los subespacios asociados a A?

- Para obtener bases para el *espacio fila* y el *espacio columna* de A,

¿Cómo hallar bases para los subespacios asociados a A?

- Para obtener bases para el *espacio fila* y el *espacio columna* de A, generamos la forma escalonada reducida de A :

```
>> R = rref(A)
```

¿Cómo hallar bases para los subespacios asociados a A?

- Para obtener bases para el *espacio fila* y el *espacio columna* de A, generamos la forma escalonada reducida de A :

```
>> R = rref(A)
```

  - Una **base para** ren (A) son las filas **no** nulas de R y

¿Cómo hallar bases para los subespacios asociados a A?

- Para obtener bases para el *espacio fila* y el *espacio columna* de A, generamos la forma escalonada reducida de A :

```
>> R = rref(A)
```

  - Una **base para** ren (A) son las filas **no** nulas de R y
  - una **base para** col (A) las columnas de A correspondientes a los unos principales de R.

¿Cómo hallar bases para los subespacios asociados a A?

- Para obtener bases para el *espacio fila* y el *espacio columna* de A, generamos la forma escalonada reducida de A :

```
>> R = rref(A)
```

  - Una **base para** ren (A) son las filas **no** nulas de R y
  - una **base para** col (A) las columnas de A correspondientes a los unos principales de R.

- La instrucción

```
>> N = null(A,'r')
```

¿Cómo hallar bases para los subespacios asociados a A?

- Para obtener bases para el *espacio fila* y el *espacio columna* de A, generamos la forma escalonada reducida de A :

```
>> R = rref(A)
```

  - Una **base para** ren (A) son las filas **no** nulas de R y
  - una **base para** col (A) las columnas de A correspondientes a los unos principales de R.

- La instrucción

```
>> N = null(A,'r')
```

  genera una matriz cuyas columnas son una base para el *espacio nulo* nul (A) de A.

Generando matrices aleatorias

En algunas ocasiones, es útil generar *al azar* matrices cuyos valores se distribuyen uniformemente en ciertos intervalos.

Generando matrices aleatorias

En algunas ocasiones, es útil generar *al azar* matrices cuyos valores se distribuyen uniformemente en ciertos intervalos.

	Instrucción	**Intervalo**	**Entradas**
$m \times n$:	`>> rand(m,n)`	$(0,1)$	*Reales*
	`>> 2*rand(m,n)-1`	$(-1,1)$	″
	`>> k*rand(m,n)`	$(0,k)$	″
	`>> k*(2*rand(m,n)-1)`	$(-k,k)$	″

Generando matrices aleatorias

En algunas ocasiones, es útil generar *al azar* matrices cuyos valores se distribuyen uniformemente en ciertos intervalos.

	Instrucción	Intervalo	Entradas
$m \times n$:	`>> rand(m,n)`	$(0, 1)$	*Reales*
	`>> 2*rand(m,n)-1`	$(-1, 1)$	″
	`>> k*rand(m,n)`	$(0, k)$	″
	`>> k*(2*rand(m,n)-1)`	$(-k, k)$	″
$n \times n$:	`>> round(k*rand(n))`	$(0, k)$	*Enteras*
	`>> round(k*(2*rand(n)-1))`	$(-k, k)$	″