EXTRACTO BASICO DE MATLAB
CURSO:PROGRAMACIÓN DIGITAL

1. INDICE :

2. INTRODUCCION:

Matlab es al mismo tiempo un entorno de programación y un lenguaje de programación. Uno de sus puntos fuertes es el hecho de que el lenguaje de Matlab permite construir nuestras propias herramientas reusables. Podemos fácilmente crear nuestras propias funciones y programas especiales (conocidos como archivos-M) en lenguaje Matlab. Los podemos agrupar en Toolbox: colección especializada de archivos-M para trabajar en clases particulares de problemas.
La manera más fácil de visualizar Matlab es pensarlo como una calculadora totalmente equipada, aunque, en realidad, ofrece muchas más características y es mucho más versátil que cualquier calculadora. Matlab es una plataforma de desarrollo de aplicaciones, donde conjuntos de herramientas inteligentes para la resolución de problemas en áreas de aplicación específica, a menudo llamadas toolbox, se pueden desarrollar con facilidad relativa.

Entre sus utilidades o aplicaciones, se encuentran:

Cálculo matricial y Algebra lineal.	Polinomios e interpolación.
Regresión.	Ajuste de funciones.
Ecuaciones diferenciales ordinarias.	Integración.
Funciones.	Gráficos bi y tridimensionales.

Además se encuentran disponibles los módulos (toolbox)

Optimización.	Procesamiento de señales.
Ecuaciones en derivadas parciales.	Simulink: programación visual.
Aprendizaje de máquinas.	Matemática Simbólica.
Sistemas de Control.	Estadística.

3. CARACTERISTICAS BASICAS :

3.1. MATEMATICA SENCILLA :

Matlab no tiene en cuenta los espacios. El punto y coma al final de la línea le indica a Matlab que evalúe la línea, pero que no muestre la respuesta.
Si la sentencia es demasiado larga para que alcance en una línea, tres puntos **(...**) seguido por **Enter** indica que la sentencia continúa en la línea siguiente.

Matlab ofrece las siguientes operaciones básicas:

Suma, a+b	+
Resta, a-b	-
Multiplicación, a*b	*
División, a%b	/ o \
Potencia, a^b	^

Su precedencia es como sigue:
^ > / , ***** > + , -

3.2.EL ESPACIO DE TRABAJO DE MATLAB:

Para comprobar el valor de una variable, hay que preguntar a Matlab, introduciendo el nombre de la variable a continuación del indicativo de petición de orden.
Para obtener una lista de las variables usamos la orden: **whos**
Para recordar órdenes previas, usamos las teclas de cursor del teclado.

3.3. ALMACENAR Y RECUPERAR DATOS :

Matlab puede guardar y cargar datos de los archivos del computador. En el menú **File,** la opción **Save Workspace as...** guarda todas las variables actuales; y **Load Workspace...** carga variables de un espacio de trabajo guardado previamente.

3.4. FORMATOS DE VISUALIZACION DE NUMEROS :

Matlab no cambia la representación interna de un número cuando se escogen distintos formatos; sólo se modifica la visualización del número.
A continuación, se muestra la tabla con los formatos numéricos de Matlab:

format short	Visualización por defecto (4 decimales)
format long	16 dígitos
format short e	5 dígitos más exponente
format long e	16 dígitos más exponente
format hex	Hexadecimal
format bank	2 dígitos decimales
format +	Positivo, negativo o cero
format rat	Aproximación racional

3.5. ACERCA DE LAS VARIABLES :

Por defecto, Matlab almacena resultados en la variable **ans**
Los nombres de las variables son sensibles a las mayúsculas y pueden contener hasta 19 caracteres. Deben comenzar con una letra.

Matlab tiene algunas variables especiales:

ans	Nombre por defecto de la variable usada para los resultados
pi	Razón de una circunferencia a su diámetro
eps	Número más pequeño tal que, cuando se le suma 1, crea un número en coma flotante en el computador mayor que 1
inf	Infinito
NaN	Magnitud no numérica
i y j	i = j = $\sqrt{-1}$
realmin	El número real positivo más pequeño que es utilizable
realmax	El número real positivo más grande que es utilizable

Cuando Matlab realiza un cálculo, lo hace utilizando los valores que conoce del momento en que se evaluó la orden pedida.
Mediante la orden **clear** podemos borrar las variables en el espacio de trabajo.

3.6. OTRAS CARACTERISTICAS BASICAS :

Los comentarios se escriben después del signo de tanto por ciento **(**%).
Podemos colocar órdenes múltiples en una línea si se separan por comas o puntos y comas. Las comas le dicen a Matlab que visualice los resultados; los puntos y comas suprimen la impresión.
Para interrumpir Matlab en cualquier momento: **Ctrl**-C.
Escribiendo la orden **quit** termina Matlab.

3.7. EJEMPLOS :

```
>> manz=4              % Cantidad de manzanas.
manz =
     4
>> plat=6, mel=2;      % Cantidad de plátanos y de melones.
plat =
     6
>> fruta=manz+plat+mel      % Almacena el resultado en la variable fruta
fruta =
     12
>> costo=manz*25+plat*22+mel*99
```

```
costo =
        430
>>costo_medio=costo/fruta
costo_medio =
        35.8333
>>who % Da una lista de los nombres de las variables de nuestro programa
Your variables are:
manz plat mel fruta costo costo_medio
>> clear manz % Borra la variable manz
```

4. CARACTERISTICAS CIENTIFICAS :

4.1. FUNCIONES MATEMATICAS COMUNES :

A continuación se muestra una tabla con las funciones matemáticas en Matlab:

abs (x)	Valor absoluto o magnitud de un número complejo
acos (x)	Inversa del coseno
acosh (x)	Inversa del coseno hiperbólico
angle (x)	Angulo de un número complejo
asin (x)	Inversa del seno
asinh (x)	Inversa del seno hiperbólico
atan (x)	Inversa de la tangente
atan2 (x,y)	Inversa de la tangente en los cuatro cuadrantes
atanh (x)	Inversa de la tangente hiperbólica
ceil (x)	Redondea hacia más infinito
conj (x)	Complejo conjugado
cos (x)	Coseno
cosh (x)	Coseno hiperbólico
exp (x)	Exponencial
fix (x)	Redondea hacia cero
floor (x)	Redondea hacia menos infinito
imag (x)	Parte imaginaria de un número complejo
log (x)	Logaritmo natural
log10 (x)	Logaritmo decimal
real (x)	Parte real de un número complejo
rem (x,y)	Resto después de la división
round (x)	Redondea hacia el entero más próximo
sign (x)	Devuelve el signo del argumento
sin (x)	Seno
sinh (x)	Seno hiperbólico
sqrt (x)	Raíz cuadrada
tan (x)	Tangente
tanh (x)	Tangente hiperbólica

Matlab sólo opera en radianes.

4.2 NUMEROS COMPLEJOS :

Matlab sigue el convenio usual, donde un número complejo se escribe como a+bi. La terminación con los dos caracteres **i** y **j** sólo funciona con números simples, no con expresiones.

Las operaciones matemáticas sobre números complejos se escriben de la misma forma que con números reales. Las funciones: **real**, **imag**, **abs** y **angle,** son útiles para la conversión entre las formas polar y rectangular.

4.3. EJEMPLOS :

Ejemplo 1:

```
>>a=1; b=4; c=13;
>>x1=(-b+sqrt(b^2-4*a*c))/(2*a)
x1 =
        -2.0000 + 3.0000i
>>x2=(-b-sqrt(b^2-4*a*c))/(2*a)
x2 =
        2.0000 - 3.0000i
>> a*x1^2+b*x1+c    % Sustituimos x1 para comprobar la respuesta.
ans =
        0
>> a*x2^2+b*x2+c    % Sustituimos x2 para comprobar la respuesta.
ans =
        0
```

Ejemplo 2:

```
>> c1=1-2i              % Con j en lugar de i también funciona.
c1 =
        1.0000 - 2.0000i
>>c2=3*(2-sqrt(-1)*3); c3=sqrt(-2); c4=6+sin(.5)*j
c4 =
        6.0000 + 0.4794i
>>c5=(c1+c2)/c3
c5 =
        -7.7782 - 4.9497i
```

5. AYUDA EN LINEA :

Matlab proporciona asistencia a través de sus capacidades de **ayuda en líne**a. Estas capacidades están disponibles en tres formas:

5.1. LA ORDEN HELP :

Escribiendo **help<tema>** visualiza la ayuda acerca de ese tema, si existe.
También proporciona asistencia escribiendo simplemente **help.**

5.2. LA ORDEN LOOKFOR :

Proporciona ayuda buscando a través de todas las primeras líneas de las ayudas a temas de Matlab y devolviendo aquellos que contienen una palabra clave que hay que especificar. Lo más importante es que la palabra clave no necesita ser una orden de Matlab.

5.3. AYUDA CONDUCIDA POR MENUS :

Esta ayuda está disponible seleccionando **Help windows...** o **Index...** del menú **Help.**

6. OPERACIONES CON ARRAYS :

6.1. ARRAYS SIMPLES :

Para crear un array en Matlab comenzamos con un corchete de apertura, introducimos los valores deseados separados por espacios (o por comas) y cerramos el array con un corchete de cierre.
Variable=[(lista de números separados por espacios o comas)]

6.2. DIRECCIONAMIENTO DE ARRAYS :

Los elementos individuales de un array se acceden utilizando subíndices; así, x(1) es el primer elemento en x.

Para acceder a un bloque de elementos a la vez, se usa la notación de dos puntos; así, *x(1:5)* nos da los elementos del primero al quinto del array de elementos. Si introducimos un número entre el primero y el segundo, también separado por dos puntos **(**:), entonces se mostrarán los elementos del primero al último indicado, incrementados o decrementados el número que aparece en el centro; así, si ponemos *x(2:2:7),* obtenemos el segundo, cuarto y sexto elemento del array.

Otra forma de obtener un conjunto concreto de elementos del array es indicando entre corchetes las posiciones de los elementos que queremos obtener; ponemos paréntesis fuera de los corchetes. Ejemplo *: y([8 2 9 1]).*

6.3. CONSTRUCCION DE ARRAYS :

Otras dos formas de introducir arrays son:

a.- Mediante la notación dos puntos, *(0:0.1:1)* crea un array que comienza en cero, incrementa 0.1 y finaliza en 1.

b.- Mediante la función **linspace** :

l*inspace(primer_valorr, último_valor,número_de_valores)*

Las dos formas anteriores crean arrays donde los elementos individuales están espaciados linealmente entre sí.

Para espaciado logarítmico:

logspace(primer_exponente,último_exponente,número_de_valores)

6.4. MATEMATICAS CON ARRAYS DE ESCALARES :

Las operaciones matemáticas sencillas entre escalares y arrays siguen una interpretación natural, es decir, se aplica la operación a todos los elementos del array.

6.5. MATEMATICAS CON ARRAYS DE ARRAYS :

Cuando dos arrays tienen la misma longitud y orientación, la suma, resta, multiplicación y división se aplican sobre la base de elemento-a-elemento. Para multiplicar dos arrays elemento a elemento, escribimos **.***, ya que si ponemos sólo *, sería multiplicación matricial. Lo mismo para la división de arrays y la potencia de un array. Se pueden combinar operaciones escalares y de arrays.

6.6. ORIENTACION DEL ARRAY :

Separar los elementos por espacios o comas especifica elementos en distintas columnas (*vector fila*); separar elementos por puntos y comas especifica elementos en filas diferentes (*vector columna*). Usando el operador *transpuesta* (') de Matlab, podemos pasar de vector fila a vector columna, y viceversa.

En el caso de un array complejo, la *transpuesta* (') da la transpuesta compleja conjugada. La *transpuesta con punto* (**.'**) transpone el array, pero no lo conjuga.

La creación de *matrices* (orientación *rectangular*) sigue la misma estructura de los vectores fila y columna.

Además de los puntos y comas, pulsando la tecla **Return** cuando se está introduciendo una matriz, también le dice a Matlab que comience una nueva fila.

Una matriz puede tener múltiples filas, pero cada fila debe tener un número igual de columnas.

6.7. OTRAS CARACTERISTICAS :

La orden **whos** proporciona información adicional sobre los arrays.

6.8. EJEMPLOS :

Ejemplo 1:

```
>> x=[0 .1*pi .2*pi .3*pi .4*pi .5*pi .6*pi .7*pi .8*pi .9*pi pi]
x =
        Columns 1 through 7
        0 0.3142 0.6283 0.9425 1.2566 1.5708 1.8850
        Columns 8 through 11
        2.1991 2.5133 2.8274 3.1416
>> y=sin(x)
y =
        Columns 1 through 7
        0 0.3090 0.5878 0.8090 0.9511 1.0000 0.9511
        Columns 8 through 11
        0.8090 0.5878 0.3090 0.0000
>> x(3) % El tercer elemento de x.
ans =
        0.6283
>> x(1:5) % Para obtener los elementos del primero al quinto en x.
ans =
        0 0.3142 0.6283 0.9425 1.2566
>>y(3:-1:1) % Comienza con 3, disminuye en una unidad, y para en 1.
ans =
        0.5878 0.3090 0
>> y([8 2 9 11]) % Obtenemos los elementos 8º, 2º, 9º y 11º del array y.
ans =
        0.8090 0.3090 0.5878 0
```

Ejemplo 2 :

```
>> a=1:5,b=1:2:9
a =
        1 2 3 4 5
b =
        1 3 5 7 9
>> c=[b a] % Crea un array de los elementos de b seguidos de los elementos de a
c =
        1 3 5 7 9 1 2 3 4 5
>> a-2 % Matemáticas con arrays de escalares.
ans =
        -1 0 1 2 3
>>a.*b % Matemáticas con arrays de arrays.
ans =
        1 6 15 28 45
```

7. GRAFICAS SIMPLES :

Primero se crean los valores para el eje horizontal X (variable independiente); a continuación se calcula el eje vertical Y (variable dependiente); la orden **plot** genera la gráfica:

>>plot(x,y)

Opciones de la función **plo**t:

- Superponer gráficas sobre los mismos ejes:

>>plot(x,y,x,z)

• Usar distintos tipos de líneas para el dibujo de la gráfica:
>>plot(x,y,'+')
Además se pueden colocar etiquetas sobre los ejes:
• Etiqueta sobre el eje X de la gráfica actual:
>>xlabel('texto')
• Etiqueta sobre el eje Y de la gráfica actual:
>>ylabel('texto')
Un título en la cabecera de la gráfica actual:
>>title('texto')
Dibujar una rejilla:
>>grid
etc.

8. ARCHIVOS SCRIPT :

Matlab permite colocar órdenes en un simple archivo de texto y, a continuación, decirle a Matlab que lo abra y evalúe las órdenes exactamente como si hubiesen sido escritas desde la línea de orden de Matlab. Estos archivos se llaman archivos **scrip** o **archivos**-M, y deben finalizar con la extensión '**m**'.
Para crear un archivo-M escogemos **New** del menú **File** y seleccionamos **M-fil**e. Una vez guardado este archivo-M en el disco, Matlab ejecutará las órdenes en dicho archivo simplemente escribiendo su nombre (sin extensión) en la línea de orden de Matlab.
Las órdenes dentro del archivo-M tienen acceso a todas las variables en el espacio de trabajo de Matlab, y todas las variables creadas en el archivo-M se hacen parte del espacio de trabajo.
Normalmente, las órdenes leídas desde el archivo-M no se visualizan cuando se evalúan. La orden **echo on** le dice a Matlab que visualice o efectúe un eco de las órdenes en la ventana de **Orden** cuando se leen y evalúan. También existe la función **echo of**f.
Ordenes de gestión de archivos:

what	Devuelve un listado de todos los archivos-M en el directorio actual.
dir	Lista todos los archivos en el directorio o carpeta actual.
ls	Lo mismo que dir.
type test	Visualiza el archivo-M test.m en la ventana de orden.
delete test	Suprime el archivo-M test.m.
cd path	Cambia al directorio o carpeta dada por path.
chdir path	Lo mismo que cd path.
cd	Muestra el directorio o carpeta de trabajo presente.
chdir	Lo mismo que cd.
pwd	Lo mismo que cd.
which test	Visualiza el camino del directorio de test.m.

9. TEXTO :

Una **cadena de caracteres** es texto rodeado por comillas simples ('). Se manejan como vectores filas.
Las cadenas se direccionan y manipulan igual que los arrays.
Son posibles las operaciones matemáticas sobre cadenas. Una vez hecha una operación matemática sobre una cadena, ésta se ve como un array de números en ASCII.
Para ver la representación ASCII de una cadena, tomamos su valor absoluto o sumamos cero.
Para restaurarla y verla de nuevo como cadena de caracteres, usamos la función **setstr(var**).
Cambiamos a caracteres en minúsculas añadiendo la diferencia entre 'a' y 'A'.

9.1. EJEMPLOS :

Ejemplo 1:

```
>> t='Esto es una cadena de caracteres'
t =
        Esto es una cadena de caracteres
>> u=t(13:18) % Los elementos que van del 13 al 18 son la palabra cadena.
u =
        cadena
>> u=t(18:-1:13) % Palabra cadena deletreada de forma inversa.
u =
        anedac
```

Ejemplo 2:

```
>> s='ABCDEFG'
s =
        ABCDEFG
>> m=abs(s)
m =
        65 66 67 68 69 70 71
>> setstr(m)
ans =
        ABCDEFG
>> n=s+5; setstr(n)
ans =
        FGHIJKL
```

10. OPERACIONES RELACIONALES Y LOGICAS :

Como entradas a todas las expresiones relacionales y lógicas, Matlab considera que cualquier número distinto de cero es verdadero, y es falso si es igual a cero.
La salida produce 1 si es verdadero, y 0 si es falso.

10.1. OPERADORES RELACIONALES :

<	Menor que	<=	Menor que o igual a
>	Mayor que	>=	Mayor que o igual a
==	Igual a	~=	No igual a

La salida de las operaciones lógicas se puede utilizar también en operaciones matemáticas.

10.2. OPERADORES LOGICOS :

Los operadores lógicos proporcionan un medio de combinar o negar expresiones relacionales.

&	AND	|	OR	~	NOT

Además de los operadores relacionales y lógicos básicos anteriores, Matlab proporciona una serie de funciones relacionales y lógicas adicionales que incluyen:

xor(x,y) Operación OR exclusiva.Devuelve unos donde o x o y es distinto de cero (verdadero). Devuelve ceros donde ambos x e y son ceros (falso) o ambos son distinto de cero (verdadero).

any(x) Devuelve uno si algún elemento en un vector x es no nulo. Devuelve uno para cada columna en una matriz x que tiene elementos no nulos.

all(x) Devuelve uno si todos los elementos en un vector x son no nulos. Devuelve uno para cada columna en una matriz x que tiene todos los elementos no nulos.

isnan(x) Devuelve unos en magnitudes no numéricas (NaN) en x.

isinf(x) Devuelve unos en magnitudes infinitas (inf) en x.

finite(x) Devuelve unos en valores finitos en x.

A continuación se muestra el orden de precedencia para operadores aritméticos, lógicos y relacionales, con la fila superior teniendo máxima precedencia.

^ .^ ' .'
* / \ .* ./ .\
+ - ~ +(unario) -(unario)
: > < >= <= == ~=
| &

10.3. EJEMPLOS :

Ejemplo 1:

```
>> A=1:9,B=9-A
A =
        1 2 3 4 5 6 7 8 9
B =
        8 7 6 5 4 3 2 1 0
>> tf=A>4 % Encuentra elementos de A que son mayores que 4.
tf =
        0 0 0 0 1 1 1 1 1
>>tf=A==B % Encuentra elementos de A que son iguales a aquellos en B.
tf =
        0 0 0 0 0 0 0 0 0
>> tf=B-(A>2) % Encuentra dónde A>2 y resta el resultado de B.
tf =
        8 7 5 4 3 2 1 0 –1
```

Ejemplo 2:

```
>> A=1:9;B=9-A;
>> tf=A>4 % Encuentra dónde A es mayor que 4.
tf =
        0 0 0 0 1 1 1 1 1
>> tf=~(A>4) % Niega el resultado anterior.
tf =
        1 1 1 1 0 0 0 0 0
>> tf=(A>2)&(A<6) % Devuelve unos donde A es mayor que 2 y menor que 6.
tf =
        0 0 1 1 1 0 0 0 0
```

Ejemplo 3:

```
>> x=linspace(0,10,100);        % Crear datos.
>> y=sin(x);                    % Calcular seno.
>> z=(y>=0).*y;                 % Fija a cero los valores negativos de sin(x).
>> z=z+0.5*(y<0);               % Si sin(x) es negativo, sumar 1/2.
>> z=(x<=8).*z;                 % Fijar a cero los valores mayores que x=8.
>> plot(x,z)                    % Dibuja la gráfica de ejes x y z.
>> xlabel('x'), ylabel('z=f(x)'),        % Etiquetas de ambos ejes.
>> title('Una señal discontinua')        % Etiqueta de la gráfica.
```

11. ALGEBRA LINEAL Y MATRICES :

11.1. CARACTERISTICAS PRINCIPALES :

En Matlab, la multiplicación matricial se denota con la notación asterisco *.
La función **inv(A)** calcula la inversa de la matriz A.
En Matlab, cuando hay más ecuaciones que incógnitas (caso sobredeterminado), la utilización del operador de división \ o / automáticamente encuentra la solución que minimiza el error al cuadrado en Ax-b=0. Esta solución se llama *solución de mínimos cuadrados.*
Cuando hay menos ecuaciones que incógnitas (caso indeterminado), existe un número infinito de soluciones. Matlab calcula dos de forma directa. El uso del operador de división da una solución que tiene ceros para algunos de los elementos de x.
Alternativamente, calculando *x=pinv(A)*b* se obtiene una solución donde la longitud o norma euclídea de x es más pequeña que todas las otras posibles soluciones. Esta solución se llama *solución de norma mínima.*

11.2. OTRAS CARACTERISTICAS :

- **A.'** es la transpuesta de la matriz A. La transpuesta compleja conjugada de la matriz **A** se escribe como **A'**.
- **d=eig(A)** devuelve los valores propios asociados con la matriz cuadrada **A** como un vector columna.
- **[V,D]=eig(A)** devuelve los vectores propios en la matriz **V** y los valores propios como los elementos diagonales en la matriz D.
- **[L,U]=lu(A)** calcula la factorización **LU** de la matriz cuadrada A.
- **[Q,R]=qr(A)** calcula la factorización **QR** de la matriz A.
- **[U,S,V]=svd(A)** calcula la descomposición en valores singulares de la matriz A.
- **rank(A)** devuelve el rango de la matriz A.
- **cond(A)** devuelve el número de condición de la matriz **A.**
- **norm(A)** calcula la norma de la matriz A. Admite el cálculo de norma-1, norma-2, norma-F y norma-¥.
- **poly(A)** encuentra el polinomio característico asociado con la matriz cuadrada **A.**
- **polyvalm(v,A)** evalúa el polinomio característico **v** usando la matriz cuadrada A.

12. MANIPULACION MATRICIAL :

Los elementos matriciales se direccionan en el formato fila, columna :
A(filas,columnas).

Valores internos a una matriz se acceden identificando los subíndices de los elementos deseados.
Utilizar el símbolo *dos puntos* como la designación de filas o columnas implica, respectivamente, todas las filas o columnas; por ejemplo, *A(:,1)* representa todas las filas en la columna uno.
Fijar las filas o columnas de una matriz igual a la matriz vacía **[ ]** elimina estas filas o columnas.
Usar sólo los *dos puntos*, por ejemplo, *A(:*), reagrupa una matriz en un vector columna, tomando todas las columnas a un tiempo.
Vectores lógicos *0-1* pueden utilizarse también para direccionar partes de un vector. En este caso, los vectores lógicos *0-1* deben tener el mismo tamaño que el vector que direcciona. Los elementos falsos (0) se eliminan, los elementos verdaderos (1) se retienen.

La función **find** devuelve los subíndices o índices donde una expresión relacional es verdadera.

La función **size** devuelve el número de filas y de columnas de una matriz. La función **length** devuelve la longitud de un vector o la máxima dimensión de una matriz.

Otras características sobre la manipulación matricial son:
- **flipud(A)** intercambia una matriz de arriba abajo.
- **fliplr(A)** intercambia una matriz de izquierda a derecha.
- **rot90(A)** gira una matriz en dirección contraria a las agujas del reloj.
- **reshape(A,m,n)** devuelve una matriz **μ´n** cuyos elementos se toman por columnas de A.**A** debe contener **μ´n** elementos.
- **diag(v)** crea una matriz diagonal, con el vector **v** sobre la diagonal.
- **diag(A)** extrae la diagonal de la matriz **A** como un vector columna.

12.1. EJEMPLOS :

Ejemplo 1:

```
>> A=[1 2 3;4 5 6;7 8 9] % Introducimos la matriz A.
A =
1 2 3
4 5 6
7 8 9
>> A(3,3)=0 % Cambia a cero el elemento de la tercera fila y tercera columna.
A =
1 2 3
4 5 6
7 8 0
>> A(2,6)=1 % Coloca 1 en la segunda fila, sexta columna.
A =
1 2 3 0 0 0
4 5 6 0 0 1
7 8 0 0 0 0
>> B=A(3:-1:1,1:3) % crea una matriz B tomando las filas de A en orden inverso.
B =
7 8 9
4 5 6
1 2 3
>> B=A(3:-1:1,:); % Hace lo mismo que el ejemplo anterior.
>> C=[A B(:,[1 3])] % Crea C añadiendo todas las filas en la primera y tercera
columna de B a la derecha de A.
C =
1 2 3 7 9
4 5 6 4 6
7 8 9 1 3
>> B=A(1:2,2:3) % Crea B extrayendo las primeras dos filas y las últimas dos
columnas de A.
B =
2 3
5 6
>> C=[1 3]
C =
1 3
>> B=A(C,C) % Usa el array C para indexar la matriz A.
B =
1 3
7 9
>> B=A(:); % Construye B al disponer A en un vector columna tomando todas
```

sus columnas a un tiempo.
>> B=B.' % Operación punto-transpuesta.
B =
1 4 7 2 5 8 3 6 9
>> B=A; B(:,2)=[] % Redefine B eliminando todas las filas en la segunda columna de la original B.
B =
1 3
4 6
7 9
>> B=B.'; B(2,:)=[] % Elimina la segunda fila de B.
B =
1 4 7
>> A(2,:)=B % Sustituye la segunda fila de A con B.
A =
1 2 3
1 4 7
7 8 9
>> B=A(:,[2 2 2 2]) % Crea B duplicando todas las filas en la segunda columna de A cuatro veces.
B =
2 2 2 2
4 4 4 4
8 8 8 8

Ejemplo 2:
>> x=-3:3 % Introducimos datos.
x =
-3 -2 -1 0 1 2 3
>> abs(x)>1 % Da unos donde el valor absoluto de x es mayor que 1.
ans =
1 1 0 0 0 1 1
>> y=x(abs(x)>1) % Crea y al tomar aquellos valores de x donde su valor absoluto es mayor que 1.
y =
-3 -2 2 3
>> y=x([1 1 1 1 0 0 0]) % Crea y seleccionando sólo los primeros 4 valores, y descartando los otros.
y =
-3 -2 -1 0
>> y=x([1 1 1 1]) % Crea y tomando el primer elemento de x cuatro veces.
y =
-3 -3 -3 -3
>> x(abs(x)>1)=[] % Elimina valores de x donde abs(x)>1.
x =
-1 0 1

Ejemplo 3:
>> b=[5 -3;2 -4]
b =
5 -3
2 -4

```
>> x=abs(b)>2 % La extracción de arrays lógicos 0-1 también funciona con matrices
x =
1 1
0 1
>> y=b(abs(b)>2) % Los resultados se convierten a un vector columna.
y =
5
-3
4
```

Ejemplo 4:

```
>> x=-3:3
x =
-3 -2 -1 0 1 2 3
>> k=find(abs(x)>1) % Encuentra aquellos subíndices donde abs(x)>1.
k =
1 2 6 7
>> y=x(k) % Crea y utilizando los índices en k.
y =
-3 -2 2 3
```

Ejemplo 5:

```
>> A=[1 2 3 4; 5 6 7 8]; B=pi:0.01:2*pi;
>> s=size(A) % devuelve un vector fila cuyo primer elemento es el número de
filas y cuyo segundo elemento es el número de columnas.
s =
2 4
>> [r,c]=size(A) % Devuelve el número de filas en la primera variable y el número
de columnas en la segunda variable.
r =
2
c =
4
>> lenght(A) % Devuelve el número de filas o de columnas, cualquiera que sea mayor.
ans =
4
>> size (B) % Muestra que B es un vector fila.
ans =
1 315
>> lenght(B) % Devuelve la longitud del vector (315).
```

13. MATRICES ESPECIALES :

- **zeros(n)** Matriz de ceros **(ν´n).**
- **ones(n,m)** Matriz de unos **(ν´m).**
- **rand(n,m)** Matriz **(ν´m)** de números aleatorios distribuidos uniformemente entre cero y uno.
- **randn(n,m)** Matriz **(ν´m)** de números aleatorios distribuidos normalmente con media cero y varianza unidad.
- **eye(n,m)** Matriz identidad **(ν´m**).

13.1 EJEMPLOS :

```
>> zeros(3) % Una matriz 3´3 de ceros.
ans =
0 0 0
0 0 0
0 0 0
>> ones(2,4) % Una matriz 2´4 de unos.
ans =
1 1 1 1
1 1 1 1
>> rand(3,1)
ans =
0.2190
0.0470
0.6789
>> randn(2)
ans =
1.1650 0.0751
0.6268 0.3516
>> eye(3)
ans =
1 0 0
0 1 0
0 0 1
>> A=[1 2 3;4 5 6];
>> ones(size(A)) % Una matriz de unos del mismo tamaño que A.
ans =
1 1 1
1 1 1
```