

Cálculo Numérico (521230) - Laboratorio 1 INTRODUCCION A MATLAB I

MATLAB es una abreviatura para matrix laboratory (laboratorio de matrices). Éste es un programa (y a la vez un lenguaje de programación) especialmente diseñado para la solución numérica de problemas matemáticos como, por ejemplo, sistemas de ecuaciones lineales, sistemas de ecuaciones no lineales, problemas de valores iniciales, etc.

La página web del programa es <http://es.mathworks.com/index.html>. En ella se encuentran ejemplos de aplicación del programa para la solución de distintos tipos de problemas reales, videos y documentación.

1. Generalidades

Observación: Si bien las guías de estos laboratorios hacen referencia a la versión de MATLAB R2013a, para el S.O. *Linux* instalado en los laboratorios LC 301 y LC 304 de la Facultad de Ciencias Físicas y Matemáticas, lo que se verá aquí, puede ser aplicable a versiones más recientes de MATLAB , versiones un poco mas antiguas o versiones en otras plataformas como *Windows* o *Mac*.

Para comenzar MATLAB , en la esquina superior izquierda: Inicio > Desarrollo > Matlab R2013a, como lo muestra la figura 1.

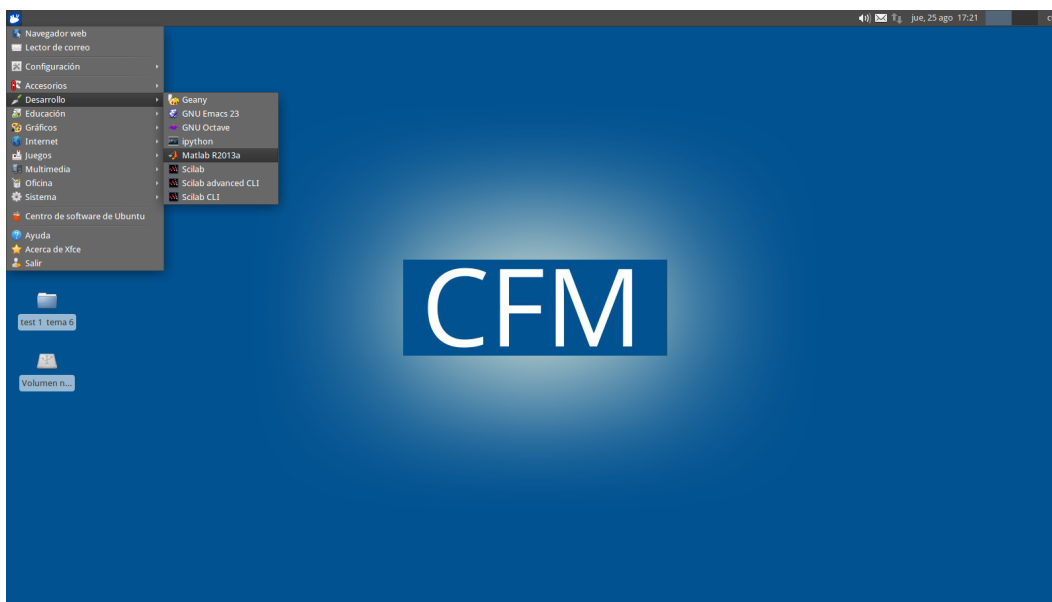


Figura 1: Menú para iniciar MATLAB .

Aparecerá el logo de MATLAB mientras inicia el programa, como se muestra en la figura 2.

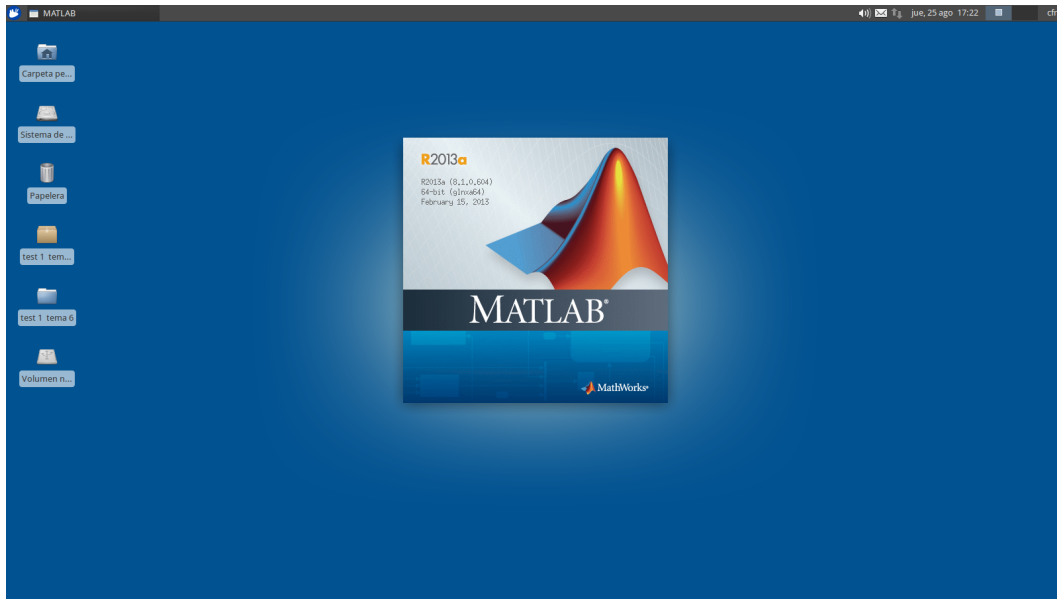


Figura 2: Logo de MATLAB .

Una vez iniciado MATLAB lo que verá será la interfaz gráfica (layout) por defecto mostrada en la figura 3. En caso de que no aparezca una interfaz como la de la figura 3, haga click en el botón *Layout* (aparece en el menú de la barra superior, aproximadamente en la mitad) y posteriormente en la opción *Default*.

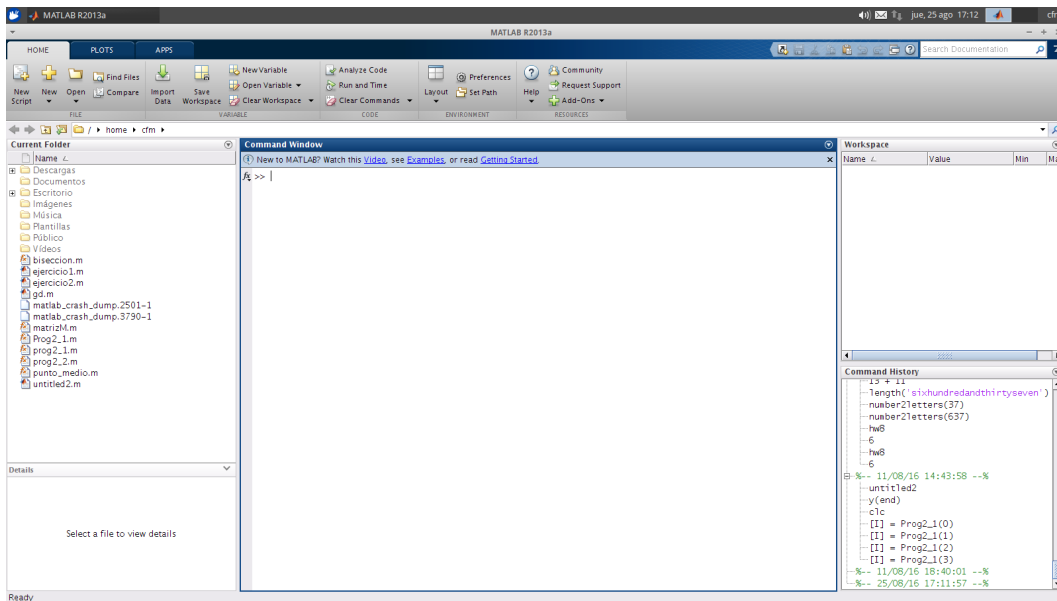


Figura 3: Interfaz gráfica (layout) de MATLAB .

Puede notar que aparecen 5 ventanas en esta interfaz:

1. *Current Folder*: Esta ventana muestra el contenido de la carpeta donde trabaja MATLAB . La dirección de esta carpeta se puede apreciar inmediatamente arriba, en la barra de navegación. La carpeta por defecto en la cual trabaja matlab es `/home/cfm`, aunque esta puede ser cambiada en la misma barra de navegación o haciendo click en alguna de las carpetas que muestra esta ventana.

2. *Details*: Esta ventana muestra propiedades de los archivos que se seleccionan en *Current Folder*, similar a las ventanas de Windows.
3. *Command Window* (Ventana de Comandos): La ventana principal de MATLAB . En esta ventana se ejecutan todos los comandos, operaciones, asignaciones y programas de MATLAB . Para ejecutar una instrucción, se escribe esta instrucción y se presiona la tecla *Enter*. Si en la ventana de comandos de MATLAB escribimos una instrucción seguido por `;`, MATLAB no mostrará el resultado de la operación realizada.
4. *Workspace*: Esta ventana muestra las variables que actualmente están guardadas en la memoria temporal de MATLAB . En cualquier momento del trabajo con MATLAB , escribiendo el comando `whos` en la ventana de comandos podremos ver las variables en memoria y con el comando `clear all` o `clear <nombre_variable>` podremos eliminarlas de memoria, aunque también se puede eliminar haciendo click con el botón derecho del mouse en la variable y seleccionando la opción *Delete*.
5. *Command History*: Muestra el historial de los últimos comandos ejecutados (correcta o incorrectamente) en la ventana de comandos. Se puede volver a ingresar un comando haciendo doble click sobre este, o bien en la ventana de comandos apretando reiteradas veces la flecha hacia arriba del teclado hasta encontrar el comando, y luego presionar la tecla *Enter*.

Para salir de MATLAB se puede ingresar `exit` o `quit` en la ventana de comandos, o bien haciendo click en la \times que aparece en la esquina superior derecha.

Se puede acceder a la ayuda en MATLAB ingresando `helpdesk` o `helpwin` o `help <nombre_comando>` en la ventana de comandos o haciendo click en el botón *Help* en la barra de menú.

En las secciones que continúan se verán los comandos necesarios para comenzar a utilizar MATLAB para la solución de los problemas matemáticos que veremos durante el semestre. Los comandos encerrados en recuadros deben ser escritos en la ventana de comandos. Las palabras que siguen a un signo `%` son sólo comentarios para explicar un comando particular y no tienen que ser escritas en la ventana de comandos de MATLAB .

TRABAJO CON NÚMEROS REALES

En MATLAB es posible realizar todas las operaciones aritméticas entre números reales. En el cuadro 1 se muestran los comandos para realizarlas.

+	adición
-	sustracción
*	producto
/	división
^	potencia

Tabla 1: Comandos para operaciones aritméticas

Cuando, como en el recuadro que sigue, realizamos una operación en MATLAB y no asignamos el resultado a ninguna variable, la respuesta se asigna a una variable especial llamada `ans`.

1

```
>> 10.5 + 3.1      % suma de dos n meros reales, el resultado se asigna a
ans
```

También podemos, con el comando `=` asignar números reales a variables.
 El comando `whos` muestra las variables almacenadas en memoria.

```

1 >> a = 10.5+3.1; % no se muestra resultado de asignación
2 >> b = 1/5      % se muestra resultado de asignación
3
4 >> a + b
5 >> a*b^3        % Primero se calcula potencia c bica de b
6 >> (a*b)^3      % Primero se multiplican a y b
7
8 >> whos         % s lo a, b y ans deber an estar en memoria
    
```

El comando `format` permite cambiar la forma en que el valor de una variable se muestra en pantalla. Tenga siempre presente que con este comando sólo se cambia la forma en que se muestra el valor de la variable, no se cambia el valor de la variable. En MATLAB los números reales se muestran por defecto en formato `short`.

```

1 >> a             % ver el valor de a con formato por defecto
2 >> format long   % cambiar formato a long
3 >> a
4 >> format rat     % cambiar formato a rat
5 >> a
6 >> format short   % cambiar formato a short
7 >> a
    
```

En la tabla 2 se encuentran los comandos que se utilizan en MATLAB para comparar dos números reales. El resultado de la comparación será 1 (si la proposición es verdadera) o 0 (si la proposición es falsa).

<code>==</code>	igual a
<code>></code>	mayor que
<code><</code>	menor que
<code>>=</code>	mayor o igual que
<code><=</code>	menor o igual que
<code>~=</code>	distinto de

Tabla 2: Comparaciones entre números reales en MATLAB

Los comandos `&&`, `||` y `~` representan los operadores lógicos conjunción (\wedge), disyunción (\vee) y negación (\sim) respectivamente.

<code>&&</code>	\wedge
<code> </code>	\vee
<code>~</code>	\sim

Tabla 3: Conectivos lógicos en MATLAB

```

1 >> c = 3/4           % asignamos 3/4 a la variable c
2 >> b == c           % b igual a c
3 >> (a > b) && (b > c) % (a mayor que b) y (b mayor que c)
4 >> ~(a <= b || a <= c) % negaci n de:
5 % (a menor o igual que b) o (a menor o igual que c)

```

Muchas de las funciones conocidas ya estan incorporadas en MATLAB . Observe que la constante `pi` contiene un valor aproximado de π .

```

1 >> abs(-3/5)         % valor absoluto de un n mero real
2 >> sqrt(9)           % raiz cuadrada de un numero real positivo
3 >> format long
4 >> pi
5 >> format rat
6 >> pi                % pi se trabaja con una aproximacion racional
7 >> format short
8
9 >> cos(pi/2)         % es cero el resultado?
10 >> sin(pi/4)
11 >> d = sqrt(2)      % ra z cuadrada de un n mero real
12
13 >> log2(d)           % logaritmo base 2 de un n mero real
14
15 >> x = 1/6
16 >> y = exp(x)        % funci n exponencial
17 >> format rat
18 >> log(y)            % logaritmo natural de un n mero real
19 >> format short

```

En los computadores personales se utiliza la aritmtica de punto flotante para almacenar los nmeros reales. En MATLAB cada nmero real se almacena en formato *double*, es decir, usando 64 bits consecutivos de memoria.

Las constantes `realmax` y `realmin` contienen el mayor nmero real positivo y el menor nmero real positivo normalizado que pueden almacenarse de esta forma. Note que `realmax` es $(1 - 2^{-53}) \cdot 2^{1024}$ y `realmin` es exactamente 2^{-1022} .

```

1 >> realmax
2 >> 2^1023
3 >> realmin
4 >> 2^(-1023)

```

Si el valor absoluto del resultado de una operacin aritmtica es mayor que `realmax`, ocurrir *overflow*. Nos daremos cuenta de que esto ha ocurrido porque MATLAB dar `Inf` o `-Inf` como resultados de la operacin, en lugar de un nmero real. El resultado en MATLAB ser `Inf` si el resultado real de la operacin aritmtica es un nmero positivo mayor que `realmax`, se obtendr un `-Inf` cuando el resultado real de la operacin aritmtica sea un nmero negativo menor que `-realmax`.

```

1  x = 2^512
2  x^2           % resultado real es 2^(1024), mayor que realmax matlab
    devuelve Inf
3  x = -2^512
4  x^2           % resultado real es -2^(1024), menor que -realmax matlab
    devuelve -Inf

```

Si el valor absoluto del resultado de una operación aritmética es menor que **realmin**, ocurre *underflow*. Pero en MATLAB se resuelve este problema de manera desapercibida para el usuario. Note que, por ejemplo, a pesar de que el resultado de la siguiente operación es menor que **realmin**, MATLAB muestra un número real como resultado y no un valor especial como en el caso anterior.

```

1  x = 2^(-512)
2  x^2

```

Si realizamos alguna operación aritmética cuyo resultado no pueda ser determinado, el resultado será NaN (*not a number*).

```

1  >> 1/0           % resultado es Inf
2  >> 0/0           % resultado es NaN

```

En MATLAB la precisión del computador (menor número real positivo x que satisface que el resultado de la operación $1 + x$ es distinto de 1) se almacena en la constante **eps**.

```

1  >> eps
2  >> (1+eps)-1      % es distinto de cero
3  >> (1+eps/2)-1    % es cero

```

2. Trabajo con vectores en MATLAB

Como su nombre lo indica, MATLAB está especialmente diseñado para el trabajo con matrices.

Un vector fila \mathbf{x} ($\mathbf{x} \in \mathbb{R}^{1 \times n}$) en MATLAB se crea escribiendo cada una de sus componentes, separadas por comas o espacios, entre []. Un vector columna \mathbf{x} ($\mathbf{x} \in \mathbb{R}^{n \times 1}$) en MATLAB se crea escribiendo sus componentes, separadas por punto y coma, entre [].

```

1  >> x = [1 2 3 4 5]           % vector fila
2  >> y = [1; 1/2; 1/3; 1/4; 1/5] % vector columna
3  >> z = 1:5                   % genera un vector igual a x
4  >> u = ones(5,1)             % crea una matriz de 5 filas y 1 columna
    (un vector columna) cuyas entradas son todas iguales a 1

```

Con los comandos `length` y `size` podemos preguntar la dimensión de un vector. El resultado de `length(x)`, siendo `x` un vector, es el número de componentes de `x`. Con `size(x)` preguntamos el número de filas (primer valor de salida) y de columnas de `x` (segundo valor de salida).

```
1 >> length(x)      % numero de componentes de x
2 >> size(x)        % numero de filas y numero de columnas de x
```

Con los mismos comandos que se usan para comparar números reales (ver cuadro 2) podemos comparar dos vectores de igual dimensión, esta comparación se hace componente a componente. El resultado será un vector con la misma dimensión que los vectores que se comparan, sus componentes tomarán los valores 0 o 1.

```
1 x == z
2 x >= y          % operacion invalida tienen el mismo n mero de
                  componentes, pero x es vector fila e y es vector columna.
3 w = x'          % asignar a w la transpuesta de x w es vector
                  columna
4 w >= y
```

Al igual que con los números reales, también podemos formar proposiciones lógicas compuestas con ayuda de conectivos lógicos. Éstos pueden ser `&`, `|` y `~`, que no son exactamente iguales a los listados en el cuadro 3, y realizan las operaciones de conjunción, disyunción y negación de proposiciones lógicas con vectores componente a componente. Si, por ejemplo, `a` y `b` son vectores de la misma longitud, `a & b` es un vector cuya componente i -ésima será 1 si las componentes i -ésimas de `a` y `b` también lo son. De lo contrario, su componente i -ésima será 0.

Con `all(x)` podremos saber si todas las componentes de un vector son distintas de cero. Con `any(x)` es posible averiguar si al menos una de las componentes de un vector es distinta de cero.

```
1 >> (w >= y) & (y >= u)      % resultado es un vector
2 >> all(ans)                 % averiguar si la relaci n se cumple para
                              todas las componentes de los vectores w, y, u.
```

Una vez que hemos creado un vector podemos ver y/o modificar sus componentes con `()`.

```
1 >> y(2)                  % elemento en 2da posici n de y
2 >> y([1 3 5])            % componentes 1, 3, 5 de y
3 >> y(2:4) = 0.5          % modificando componentes 2 a 4 de y
4 >> y([1 5]) = [-1 1]     % modificando componentes 1 y 5 de y
```

Los comandos `linspace(p,u,N)` y `p:incr:u` nos permiten crear vectores cuyas componentes son equidistantes entre sí. Con el primero de ellos se genera un vector fila de `N` componentes equidistantes entre sí, siendo `p` la primera de ellas y `u`, la última. Con el segundo se genera un vector fila cuyo primer elemento es `p`, el último es `u` y la diferencia entre dos elementos consecutivos es `incr`.

```
1 >> t = 1:.01:3           % 1 es 1er elemento de t, 3 es el ltimo
                              diferencia entre 2 elementos consecutivos es 0.01.
2 >> s = linspace(1,3,201) % s y t son iguales
```

Se pueden realizar operaciones aritméticas entre vectores (las dimensiones deben ser las correctas para cada operación).

```

1 >> s1 = x + z           % suma de vectores
2 >> 2*t                  % multiplicación por escalar
3 >> x*y                  % producto entre 2 vectores
4 >> y*x

```

Las operaciones aritméticas también pueden realizarse componente a componente.

```

1 >> p = u.*y             % producto componente a componente
2 >> d = u./y             % división componente a componente
3 >> x.^2                 % cada componente al cuadrado

```

Con la función `norm` podemos calcular normas de vectores.

```

1 >> norm(x)              % norma 2 de x
2 >> p = 1;
3 >> norm(x,p)            % norma p de x
4 >> norm(x,inf)          % norma infinito de x

```

A pesar de que para todo $\mathbf{x} \in \mathbb{R}^n$ se cumple $\|\mathbf{x}\|_2 = \sqrt{\mathbf{x}^t \mathbf{x}}$, debemos siempre calcular la norma 2 de un vector usando la función `norm` de MATLAB y no mediante `sqrt(x'*x)`. Si, por ejemplo, $\mathbf{x} = 2^{512} \begin{pmatrix} 3 \\ 4 \end{pmatrix}$, su norma 2 es $5 \cdot 2^{512}$ que es menor que `realmax`. Sin embargo, $\mathbf{x}^t \mathbf{x} = 9 \cdot 2^{1024} + 16 \cdot 2^{1024}$. Dado que los términos en esta suma son mayores que `realmax`, al calcular `sqrt(x'*x)` ocurrirá *overflow*. En general, en el trabajo con MATLAB, siempre es mejor usar las funciones provistas por el programa que usar expresiones que sean matemáticamente equivalentes.

```

1 >> x = 2^(512) [3;4]
2 >> norm(x)
3 >> sqrt(x'*x)

```

Las funciones sobre números reales disponibles en MATLAB también pueden aplicarse a vectores. Con el comando `plot` podemos graficar estas funciones.

Por ejemplo, para graficar $\sin(x)$ con x entre 0 y 2π ,

```

1 >> x = linspace(0,2*pi,100) % 100 valores entre 0 y 2*pi
2 >> y = sin(x)                % evaluar la función en x
3 >> plot(x,y)                 % graficar

```

TRABAJO CON MATRICES Y VECTORES EN MATLAB

Para crear una matriz o un vector desde sus componentes escribimos cada una de sus filas separadas por `;`. Los elementos dentro de cada fila se separan por comas o espacios y todo se encierra entre `[]`.

```

1 >> A = [1 -1;4 0;1 -1]
2 >> x = [1 -1];
3 >> B = [x;4 0;x]          % es igual a A

```

También podemos comparar matrices de igual dimensión


```
1 >> A == B
```

Junto con esto, para crear vectores existen las siguientes opciones

```
1 a=1:10 %Crea un vector de espaciado 1
2 b=1:0.1:2 %Crea un vector de espaciado 0.1
3 c=10:-1:1 %Crea un vector decreciente
4 d=linspace(1,10,10) %Crea un vector con 10 elementos equiespaciados
```

Para ver y/o modificar las entradas de una matriz usamos (). Con el comando `size` podemos averiguar el número de filas y columnas de una matriz.

```
1 >> [m,n] = size(A) % averiguar dimensi n de A
2 >> A(1,2) % elemento en posici n (1,2) de A
3 >> A(:,2) % 2da columna de A
4 >> A(3,:) = [1 0] % modifica 3ra fila de A
5 >> A(end,end) % averiguar el elemento en la ultima
    fila y columna
6 >> A(1:2:end) % Averiguar los elementos en las
    posiciones impares
7 >> A([1 3],2) % filas 1 y 3, columna 2 de A
8 >> A = [A [1;1;1]] % aadiendo columna a A
9 >> At = A' % transpuesta de A
```

También pueden realizarse operaciones aritméticas entre matrices, incluyendo operaciones aritméticas componente a componente.

```
1 >> A + At/2 % primero se multiplica At por 1/2
2 >> (A+At)/2 % primero se suman las matrices
3 >> A*At % producto de matrices
4 >> A.*At % producto componente a componente
5 >> A./At % divisi n componente a componente observe la fila
    2 de la matriz resultante.
```

Supongamos que $A \in \mathbb{R}^{n \times n}$ es una matriz invertible (su determinante es distinto de cero). Dada una matriz $B \in \mathbb{R}^{n \times m}$, la matriz $X \in \mathbb{R}^{n \times m}$ tal que $AX = B$ es $X = A^{-1}B$. Si $m = 1$, $AX = B$ es un sistema de ecuaciones lineales. La matriz X se encuentra en MATLAB escribiendo $X = A \backslash B$.

```
1 >> A = [1 -1 1;2 1 2;3 1 1] % la matriz A es invertible
2 >> det(A) % su determinante no es cero
3 >> b = [1;1;1]
4 >> x = A\b % es la soluci n de Ax = b
5 >> norm(A*x-b) % comprobando que Ax = b
6 >> B = [2 1;1 2;1 1]
7 >> X = A\B % es tal que AX = B
8 >> norm(A*X-B) % con este comando se puede calcular la
    norma 2 de una matriz
9 >> norm(A*X-B,1) % y la norma 1
10 >> norm(A*X-B,inf) % y la norma infinito
```

Si, en lugar de $A \setminus B$, escribimos B/A MATLAB devuelve un mensaje de error pues las matrices A y B de este ejemplo no permiten realizar esta operación. Al escribir B/A en MATLAB se pretende calcular BA^{-1} .

MATLAB tiene comandos para el trabajo y la construcción de matrices especiales.

```

1  >> m=3; n=2;                                % notar que se pueden asignar 2 variables
    en una misma linea
2  >> A = zeros(m,n)                            % matriz nula de m por n
3  >> B = zeros(n)                              % matriz nula de n por n
4  >> C = ones(m,n)                            % matriz con todas sus componentes iguales
    a 1, de m por n
5  >> I = eye(n)                                % matriz identidad de n por n
6  >> R = rand(m,n)                            % matriz aleatoria de n por m cuyas
    componentes contienen valores entre 0 y 1
7  >> Ri = randi([-10 10],m,n) % matriz aleatoria de n por m cuyas
    componentes contienen valores enteros entre -10 y 10
8  >> x = diag(I)                              % vector que contiene la diagonal principal
    de I
9  >> D = diag(x)                              % matriz diagonal de tama o length(x) cuya
    diagonal principal contiene a los elementos del vector x
10 >> D1 = diag(x,1)                           % matriz de tama o length(x)+1 cuyos
    elementos sobre la diagonal principal son los de x y el resto son
    nulos
11 >> D2 = diag(x,-1)                          % matriz de tama o length(x)+1 cuyos
    elementos bajo la diagonal principal son los de x y el resto son
    nulos
12 >> M = [A' I ; B C']                       % matriz formada por bloques note que los
    bloques de cada 'fila' tienen la misma cantidad de filas

```

Queda como ejercicio la experimentación de estos comandos con distintas variables o combinando comandos (por ejemplo, ver qué ocurre si ingresa `ones(m)` o `N=randi([0 100],10)` y luego `diag(diag(N))`). También queda como ejercicio ver que ocurre al hacer doble click en una matriz enlistada en el *Workspace*.

3. Cadenas y celdas

Si bien MATLAB está diseñado desde sus comienzos para trabajar con matrices, con el tiempo se ha extendido a otro tipo de variables. En particular serán utilizadas las cadenas (del inglés “string”). Las cadenas son sucesiones de caracteres, tal cual como los que esta leyendo ahora. Para ingresar cadenas a MATLAB se utilizan los operadores de comillas simples `'`. Por ejemplo la sentencia

```

1  string='Hola mundo';

```

grabará en la memoria local una variable llamada `string` cuyo valor es la cadena `Hola mundo`. Un trabajo común con las cadenas es la concatenación de ellas. En este caso, podemos concatenarlas usando fácilmente usando el mismo operador que concatena matrices, por ejemplo la sentencia

```

1  cadena1='Me llamo';
2  cadena2='MATLAB';
3  quiensoy=[cadena1,cadena2];

```

creará una cadena que es la concatenación de dos.

Las celdas o variables tipo `cell` son arreglos indexados, como las matrices, pero que permiten contener más cosas que números, en efecto pueden contener cualquier tipo de variable de Matlab. Para declarar celdas, se ocupa el operador `{`, por ejemplo

```
1 P='Esto es una cadena';
2 A=ones(1,20);
3 C={P,A};
```

crea una celda `C` donde la primera entrada es la cadena `P` y la segunda entrada es la matriz `A`. Para recuperar una entrada de una celda, se debe recorrer su índice con el operador `{}`.

4. Ejercicios

1. Escriba `help length` en la ventana de comandos y escriba en el recuadro siguiente qué devuelve MATLAB al llamar `length(M)`, siendo `M` una matriz cualquiera.

2. Ingrese a MATLAB los siguientes vectores y matrices

$$\begin{array}{llll}
 a) A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} & c) C = \begin{bmatrix} 1 & -1 & 1 \end{bmatrix} & e) D = \begin{bmatrix} 2 \\ -2 \\ 0 \end{bmatrix} & f) B + C \\
 b) B = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix} & d) E = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix} & & g) C \cdot D \\
 & & & h) D \cdot C
 \end{array}$$

3. Para este ejercicio considere

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 2 & 5 \\ 1 & 0 & 1 \end{bmatrix} \quad B = \frac{1}{2} \begin{bmatrix} -2 & 2 & -4 \\ -1 & 2 & -7 \\ 2 & -2 & 6 \end{bmatrix}$$

- a) Ingrese a MATLAB las matrices `A` y `B`.
 - b) Calcule en MATLAB los productos matriciales `AB` y `BA`.
 - c) ¿Qué observa de los resultados anteriores?
 - d) Calcule las normas 1, 2 e infinito de `A` Y `B`.
4. Usando las funciones de MATLAB para crear matrices, ingrese la matriz de orden 92×108 de la forma

$$\begin{bmatrix} 3 & 4 & 5 & 0 & 0 & \cdots & 0 \\ 0 & 3 & 4 & 5 & 0 & \cdots & 0 \\ 0 & 0 & 3 & 4 & 5 & \cdots & 0 \\ 0 & 0 & 0 & 3 & 4 & \ddots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \end{bmatrix}.$$

5. Cree una celda que contenga como elementos a las matrices identidad de órdenes 2,3,4,y 5.

6. Considere las siguientes instrucciones en lenguaje de MATLAB :

```
1 s = 'Esto es una cadena';  
2 n = 1*s  
3 c = char(b)
```

a) ¿Qué tipo de variable son **s**, **n** y **c**?

b) ¿Que retorna **s(end:-1:1)**?