Clculo 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.

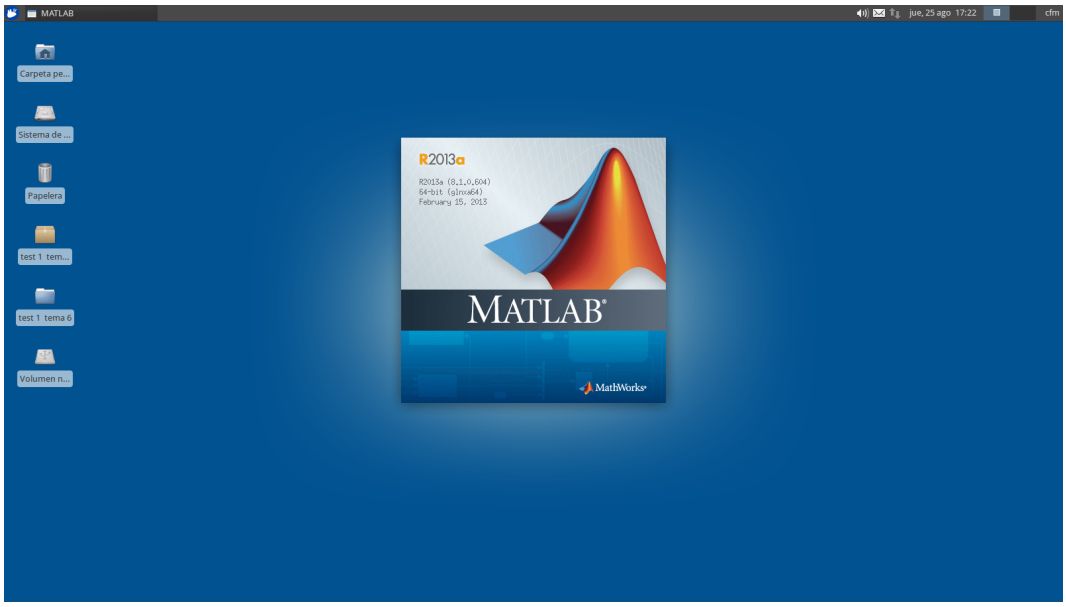

Figura 2: Logo de MATLAB.

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*.

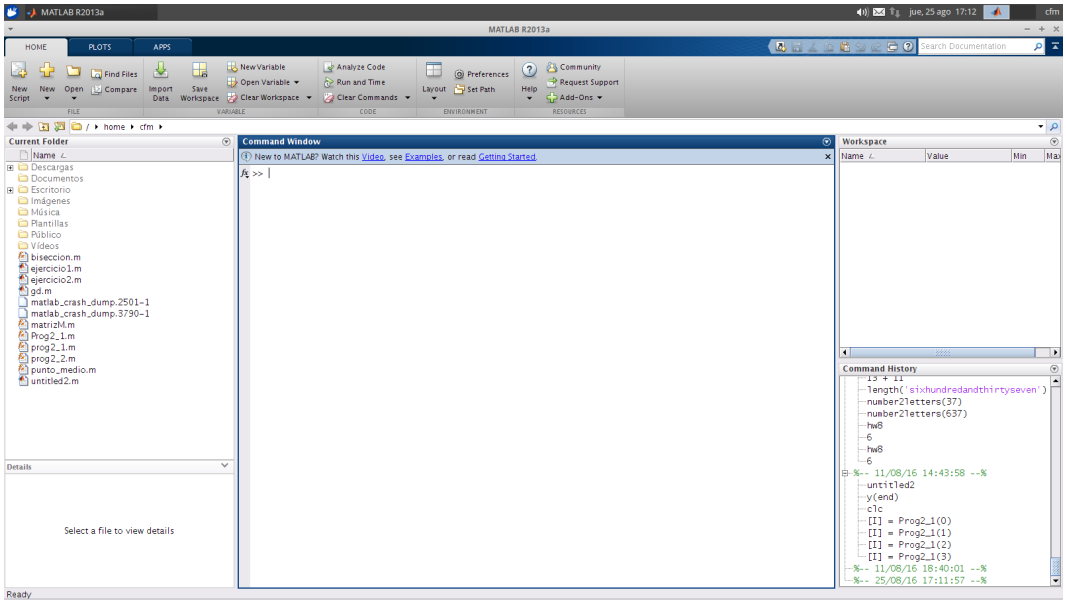

Figura 3: Interfaz gráfica (layout) de MATLAB.

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 instucción 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 muesta las variables que actualmente estan 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 boton 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.

```
>> a = 10.5+3.1;   % no se muestra resultado de asignaci n
>> b = 1/5         % se muestra resultado de asignaci n

>> a + b
>> a*b^3           % Primero se calcula potencia c bica de b
>> (a*b)^3         % Primero se multiplican a y b

>> 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`.

```
    >> a               % ver el valor de a con formato por defecto
    >> format long     % cambiar formato a long
    >> a
    >> format rat      % cambiar formato a rat
    >> a
    >> format short    % cambiar formato a short
    >> 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).

==	igual a
>	mayor que
<	menor que
>=	mayor o igual que
<=	menor o igual que
~=	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.

&&	$\wedge$
||	$\vee$
~	$\sim$

Tabla 3: Conectivos lógicos en MATLAB

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

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

```
>> abs(-3/5)       % valor absoluto de un n mero real
>> sqrt(9)         % raiz cuadrada de un numero real positivo
>> format long
>> pi
>> format rat
>> pi              % pi se trabaja con una aproximacion racional
>> format short

>> cos(pi/2)       %  es cero el resultado?
>> sin(pi/4)
>> d = sqrt(2)     % ra z cuadrada de un n mero real

>> log2(d)         % logaritmo base 2 de un n mero real

>> x = 1/6
>> y = exp(x)      % funci n exponencial
>> format rat
>> log(y)          % logaritmo natural de un n mero real
>> format short
```

En los computadores personales se utiliza la aritmética de punto flotante para almacenar los números reales. En MATLAB cada número real se almacena en formato *double*, es decir, usando 64 bits consecutivos de memoria.
Las constantes `realmax` y `realmin` contienen el mayor número real positivo y el menor número 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}.

```
>> realmax
>> 2^1023
>> realmin
>> 2^(-1023)
```

Si el valor absoluto del resultado de una operación aritmética es mayor que `realmax`, ocurrirá *overflow*. Nos daremos cuenta de que esto ha ocurrido porque MATLAB dará `Inf` o `-Inf` como resultados de la operación, en lugar de un número real. El resultado en MATLAB será `Inf` si el resultado real de la operación aritmética es un número positivo mayor que `realmax`, se obtendrá un `-Inf` cuando el resultado real de la operación aritmética sea un número negativo menor que $-$`realmax`.

```
>> x = 2^512
>> x^2            % resultado real es 2^(1024), mayor que
   realmax
                  % matlab devuelve Inf
>> x = -2^342
>> x^3            % resultado real es -2^(1026), 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.

```
>> x = 2^(-512)
>> x^2
```

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

```
>> 1/0            % resultado es Inf
>> 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`.

```
>> eps
>> (1+eps)-1      % es distinto de cero
>> (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 $\boldsymbol{x}$ ($\boldsymbol{x} \in \mathbb{R}^{1\times n}$) en MATLAB se crea escribiendo cada una de sus componentes, separadas por comas o espacios, entre `[ ]`. Un vector columna $\boldsymbol{x}$ ($\boldsymbol{x} \in \mathbb{R}^{n\times 1}$) en MATLAB se crea escribiendo sus componentes, separadas por punto y coma, entre `[ ]`.

```
>> x = [1 2 3 4 5]                  % vector fila
>> y = [1; 1/2; 1/3; 1/4; 1/5]      % vector columna
>> z = 1:5                          % genera un vector igual a x
>> 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).

```
>> length(x)     % numero de componentes de x
>> 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`.

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

```
>> (w >= y) & (y >= u)        % resultado es un vector
>> 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 `( )`.

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

```
>> t = 1:.01:3             % 1 es 1er elemento de t, 3 es el  ltimo
% diferencia entre 2 elementos consecutivos
% es 0.01.
>> 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).

```
>> s1 = x + z          % suma de vectores

>> 2*t                 % multiplicaci n por escalar

>> x*y                 % producto entre 2 vectores
>> y*x
```

Las operaciones aritméticas también pueden realizarse componente a componente.

```
>> p = u.*y            % producto componente a componente
>> d = u./y            % divisi n componente a componente
>> x.^2                % cada componente al cuadrado
```

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

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

A pesar de que para todo $\boldsymbol{x} \in \mathbb{R}^n$ se cumple $\|\boldsymbol{x}\|_2 = \sqrt{\boldsymbol{x}^{\mathrm{t}}\boldsymbol{x}}$, debemos siempre calcular la norma 2 de un vector usando la función `norm` de MATLAB y no mediante `sqrt(x'*x)`. Si, por ejemplo, $\boldsymbol{x} = 2^{512}\begin{pmatrix}3\\4\end{pmatrix}$, su norma 2 es $5 \cdot 2^{512}$ que es menor que `realmax`. Sin embargo, $\boldsymbol{x}^{\mathrm{t}}\boldsymbol{x} = 9 \cdot 2^{1024} + 16 \cdot 2^{1024}$. 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.

```
>> x = 2^(512)[3;4]
>> norm(x)
>> 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,

```
>> x = linspace(0,2*pi,100) % 100 valores entre 0 y 2*pi
>> y = sin(x)               % evaluar la funci n en x
>> plot(x,y)                % graficar
```

Trabajo con matrices en Matlab

Para crear una matriz 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 `[ ]`.

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

También podemos comparar matrices de igual dimensión

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

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.

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

`[gobble=2,frame=single]`

```
    >> [m,n] = size(A)          % averiguar dimensi n de A
    >> A(1,2)                   % elemento en posici n (1,2) de A
    >> A(:,2)                   % 2da columna de A
    >> A(3,:) = [1 0]           % modifica 3ra fila de A
    >> A([1 3],2)               % filas 1 y 3, columna 2 de A
    >> A = [A [1;1;1]]          % a adiendo columna a A
    >> At = A'                  % transpuesta de A
```

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

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

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

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

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

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

```
>> m=3; n=2;                  % notar que se pueden asignar 2
   variables en una misma linea
>> A = zeros(m,n)             % matriz nula de m por n
>> B = zeros(n)               % matriz nula de n por n
>> C = ones(m,n)              % matriz con todas sus componentes
   iguales a 1, de m por n
>> I = eye(n)                 % matriz identidad de n por n
>> R = rand(m,n)              % matriz aleatoria de n por m cuyas
   componentes contienen valores entre 0 y 1
>> Ri = randi([-10 10],m,n)   % matriz aleatoria de n por m cuyas
   componentes contienen valores enteros entre -10 y 10
>> x = diag(I)                % vector que contiene la diagonal
   principal de I
>> D = diag(x)                % matriz diagonal de tama o length(x)
   cuya diagonal  principal contiene a los elementos del vector x
>> 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
>> 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
>> 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

```
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 concadenación de ellas. En este caso, podemos concadenarlas usando fácilmente usando el mismo operador que concadena matrices, por ejemplo la sentencia

```
cadena1='Me llamo';
cadena2='MATLAB';
quiensoy=[cadena1,cadena2];
```

creará una cadena que es la concadenación 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 {.